С. О. Переяславська, В. М. Жукова,

О.О. Смагіна

JAVA ПРОГРАМУВАННЯ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА”

С. О. Переяславська, В. М. Жукова, О.О. Смагіна

JAVA ПРОГРАМУВАННЯ

Методичні рекомендації до лабораторних робіт
для студентів спеціальності

123 – „Комп’ютерна інженерія”

Старобільськ

ДЗ „ЛНУ імені Тараса Шевченка”

2018

УДК 004.42(072)
ББК 32.973-018
П27

Р е ц е н з е н т и :

Могильний Г. А. – кандидат технічних наук, доцент, директор навчально-наукового інституту фізики, математики та інформаційних технологій Луганського національного університету імені Тараса Шевченка.

Фоменко А. В.. – кандидат педагогічних наук, доцент кафедри програмного забезпечення інституту комп'ютерних наук та інформаційних технологій Національного університету «Львівська політехніка».

П27 **Переяславська С.О.** Java програмування : метод. рек. до лаб. робіт для студ. спец. 123 – „Комп'ютерна інженерія" / С. О. Переяславська, В. М. Жукова, О.О. Смагіна; Держ. закл. „Луган. нац. ун-т імені Тараса Шевченка". – Старобільськ : ДЗ „ЛНУ імені Тараса Шевченка", 2018. – 119 с.

Методичні рекомендації структуровано відповідно до розділів навчальної програми курсу "Java - програмування" для спеціальності 123 – „Комп'ютерна інженерія" кафедри інформаційних технологій та систем ДЗ ЛНУ імені Тараса Шевченка. Посібник охоплює основні групи питань з розробки програм мовою Java, починаючи з базових основ об'єктно-орієнтовного програмування, так і більш складних тем, пов'язаних з управлінням багатопотоковості, проектуванням графічного інтерфейсу користувача, використанням класів-колекцій для зберігання і обробки даних. Як інтегроване середовище розробки використовується IDE NetBeans.

Навчальний посібник призначений для студентів фізико-математичного та технічного профілю, учителів-предметників ліцеїв, коледжів, гімназій, слухачів курсів підвищення кваліфікації, а також для самоосвіти.

УДК 004.42(072)
ББК 32.973-018

Рекомендовано до друку Вченою радою
Луганського національного університету імені Тараса Шевченка
(протокол № 9 від 30 березня 2018 р.)

ЗМІСТ

ВСТУП

На сьогоднішній день Java є однією з найбільш поширених і популярних мов програмування. Java перетворилася з універсальної мови в цілу платформу і систему, яка об'єднує різні технології, що використовуються для розв'язання цілої низки завдань: від створення десктопних додатків до написання веб-порталів і сервісів. Крім того, мова Java активно застосовується як інструмент розробки програмного забезпечення для різних пристроїв: звичайних ПК, планшетів, смартфонів, мобільних телефонів і навіть побутової техніки. Це обумовлює актуальність предмету вивчення, якому присвячено методичні рекомендації.

Вивчення курсу „Java-програмування" є невід'ємною частиною у загальному процесі навчання студентів спеціальності "Комп'ютерна інженерія„. Головна мета полягає у формуванні базових знань з технології програмування мовою Java, умінь створення сучасних програмних продуктів з використанням об'єктно-орієнтованої парадигми і розв'язання різних інженерних завдань за допомогою комп'ютера. Знання, отримані при вивченні цієї дисципліни, забезпечать професійну підготовку фахівців в галузі інформаційних технологій.

Однією із основних навчальних форм є лабораторні роботи, які відіграють провідну роль у формуванні навичок та застосуванні набутих знань з Java-програмування. Лабораторні заняття логічно продовжують вивчення тематик, розпочатих на лекціях. Усі форми лабораторних занять призначені для відпрацювання практичних дій.

Методичні рекомендації з лабораторного практикуму структуровані відповідно до навчальної програми курсу „Java-програмування" і складаються з дванадцяти лабораторних робіт, що охоплюють широкий спектр питань з розробки Java-додатків. Тематика робіт спрямована на вивчення як базових основ об'єктно-орієнтовного програмування (розробка ієрархії класів та інтерфейсів, роботи з потоками введення-виведення тощо) так і більш складних тем, пов'язаних з управлінням багатопотоковості, проектуванням графічного інтерфейсу користувача, використанням класів-колекцій для зберігання і обробки даних. Як інтегроване середовище розробки

використовується IDE NetBeans, яке є найбільш адаптованим для навчальних цілей.

Методичні рекомендації містять назву, мету, основні теоретичні відомості та завдання для індивідуального виконання. Для кращого розуміння теми, яка розглядається, кожна лабораторна робота містить приклади програм, результати їх виконання. Наприкінці кожна робота має питання для самоперевірки отриманих знань. Зміст лабораторних занять передбачає роботу в комп'ютерних класах та самостійну роботу студентів.

Методичні рекомендації можуть бути корисними для студентів інших спеціальностей у якості посібника для самостійного опанування технологій Java-програмування.

ВИМОГИ ДО ВИКОНАННЯ Й ЗАХИСТУ ЛАБОРАТОРНИХ РОБІТ

Лабораторні роботи виконуються кожним студентом самостійно. Перед початком виконання лабораторної роботи студент повинен ознайомитися з теоретичним матеріалом, використовуючи джерела, наведені у списку рекомендованої літератури, чи будь-які інші; усвідомити завдання та порядок проведення роботи.

Під час виконання лабораторної роботи студент повинен поетапно виконати індивідуальне завдання відповідно до варіанту. Результатом проведеної роботи можуть бути розроблені програмні додатки мовою програмування Java в середовищі IDE NetBeans.

Після виконання лабораторної роботи студент повинен оформити звіт, який має містити наступне: ПІБ студента, № групи, курсу, назву дисципліни, номер та назву теми лабораторної роботи, завдання до роботи (відповідно до номеру варіанту), розв'язання поставленого завдання (короткий опис, алгоритм, програмний код, скріншоти, відповіді на контрольні питання тощо). До звіту додаються файли проекту у вигляді архіву.

Лабораторна робота подається до захисту безпосередньо після її виконання. При захисті роботи студент демонструє результати виконаної роботи та відповідає на контрольні запитання за темою лабораторної роботи.

Кожна лабораторна робота оцінюється за бальною системою, яка встановлена робочою програмою курсу. Оцінюється якість підготовки, повнота виконання роботи, вміст збережених файлів, зміст відповідей та оформлення звітів. Оцінці "відмінно" відповідає виконання більше 90% пунктів роботи, "добре" – більше 75%, "задовільно" – більше 50%.

МОДУЛЬ 1. ОСНОВИ МОВИ ПРОГРАМУВАННЯ JAVA

ЛАБОРАТОРНА РОБОТА №1

Тема: Установка віртуальної машини Java й інтегрованого середовища розробки NetBeans.

Мета: навчитися встановлювати основне програмне забезпечення для розробки Java-програм.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Загальні поняття

Для того, щоб використовувати програмний продукт, створений мовою програмування Java або розробляти свої власні програми, необхідно завантажити й встановити на комп'ютер віртуальну машину Java - Java Virtual Machine.

Java Virtual Machine (скорочено Java VM, JVM) - віртуальна машина Java, яка є основною частиною системи Java Runtime Environment (JRE). Віртуальна машина Java інтерпретує і виконує байт-код Java, попередньо створений компілятором Java з початкового тексту Java-програми.

Є два продукти для завантаження на комп'ютер:

• JRE - Java Runtime Environment, який надає віртуальну машину і API. Застосовується у разі, якщо потрібно використовувати готові програми на Java.

• JDK - Java Development Kit, який надає віртуальну машину, API і засоби розробки програм на Java. Застосовується, якщо передбачається розробка власних Java-програм.

Для програмування на Java існує декілька середовищ розробки (Integrated Development Environment - IDE). Найбільш популярними є NetBeans, Eclipse, Intellij IDEA.

Платформу Java можна встановлювати як самостійно, так і разом з інтегрованим середовищем розробки NetBeans.

У інтегрованому середовищі розробки NetBeans можна створювати програми на мові програмування Java, а також на інших популярних мовах програмування (Ruby, C / C ++, PHP). Це середовище підтримується компанією Oracle, та розроблено

на мові програмування Java. Тому для його роботи потрібні віртуальна машина і комплект утиліт для розробника на Java (Java Standard Developer Kit - JDK). Для зручності існують установчі комплекти, які містять відразу JDK і IDE NetBeans.

Проекти NetBeans. Пакети. Рівні видимості класів

Ідеологія Java передбачає роботу в комп'ютерних мережах і можливість завантаження у будь який момент необхідних ресурсів програми. Для забезпечення такої роботи Java-програми розробляються у вигляді великого числа незалежних класів. Однак такий спосіб розробки призводить до надзвичайно високої фрагментації програми. Навіть невеликі проекти часто складаються з десятків класів, а реальні проекти - із сотень. При цьому кожному загальнодоступному класу (public) відповідає свій файл, що має таке ж ім'я. Для того щоб впоратися з такою кількістю файлів, в Java передбачений спеціальний засіб угруповання класів - пакет (package). Пакети забезпечують незалежні простори імен (namespaces), а також обмеження доступу до класів.

Класи завжди задаються в будь-якому пакеті. Пакети можуть бути вкладеними з довільним рівнем (обмежується тільки операційною системою). Кожному пакету відповідає папка з вихідними кодами відповідних класів.

Для того щоб помістити клас в пакет, потрібно задекларувати ім'я пакета на початку файла, в якому оголошено клас, за допомогою команди:

package Имя_Пакета;

Крім того, необхідно помістити вихідний код класу в папку, що відповідає пакету.

Якщо декларація імені пакета відсутня, вважається, що клас належить пакету з ім'ям default.

Вкладеним пакетам відповідають складові імена. Наприклад, якщо ми маємо пакет з ім'ям pkg1, в який вкладено пакет з ім'ям pkg2, в який вкладено пакет з ім'ям pkg3, то оголошення класу з ім'ям MyClass1, що знаходиться в пакеті pkg3, буде наступним:

```
package pkg1.pkg2.pkg3;
class MyClass1 {
....}
```

При створенні проекту в середовищі NetBeans розміщення класу в пакеті відбувається автоматично.

При декларації класу можна вказувати, що він є загальнодоступним, за допомогою модифікатора доступу public:

```
public class MyClass2 {
....
}
```

У цьому випадку можливий доступ до даного класу з інших пакетів.

У файлі .java можна розташовувати тільки один загальнодоступний клас і довільне число класів з пакетним рівнем видимості.

Для того щоб відрізняти імена класів від назв пакетів, в Java прийнято назви пакетів писати тільки малими літерами, імена класів починати з великої літери, а імена полів даних (в тому числі імена об'єктних змінних) і методів починати з малої літери.

Порядок виконання роботи

1. Установка платформи Java та інтегрованого середовища розробки NetBeans.

Спочатку перевіримо, чи встановлена Java платформа на комп'ютері. Якщо програмний додаток встановлено, то у списку, який відкривається за командою *Панель управління, Програми та компоненти,* відображено існуюча версія Java.

2. Інсталяція програмного додатку

Зауваження! Рекомендується робити інсталяцію ПЗ з правами адміністратора.

Скачайте необхідний установник з офіційного сайту http://www.oracle.com/technetwork/java/javase/downloads

На сайті можна скачати як окрему версію JDK, так і версію разом з IDE NetBeans (рис. 1).

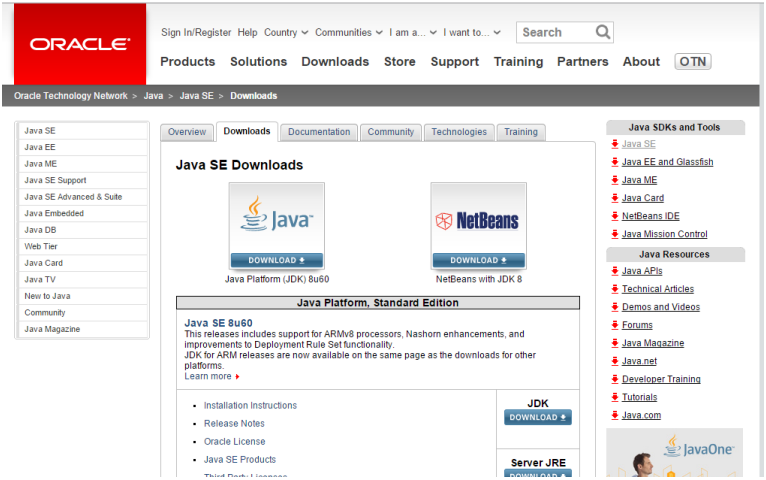

Рис. 1. Вікно завантаження JDK та NetBeans

Якщо на комп'ютері попередньо не встановлено JDK, його можна завантажити разом з середовищем NetBeans. Для цього треба вибрати кнопку, що розташована справа у вікні. Відкриється вікно, що на рис. 2.

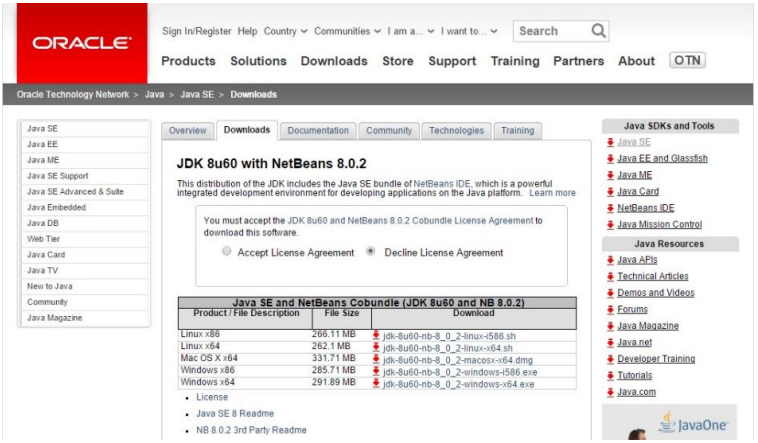

Рис. 2. Вікно інсталяції JDK та NetBeans

Далі необхідно прийняти умови угоди та вибрати установник Java NetBeans для відповідної операційної системи. Після завантаження дистрибутива перейти до інсталяції програмного забезпечення. Спочатку встановляться компоненти Java, потім NetBeans.

Зауваження! *Потрібно прописати шлях до файлів JDK в змінних середовищах операційної системи (в змінній Paht вказати шляхдоступу до дерікторії bin).*

3. Створення в NetBeans найпростішого додатка Java

Створимо за допомогою середовища NetBeans додаток Java. Для цього запустимо інтегроване середовище розробки NetBeans - далі IDE, і виберемо в головному меню *File / New Project* ... У діалозі виберемо *General / Java Application / Next* (рис. 3).

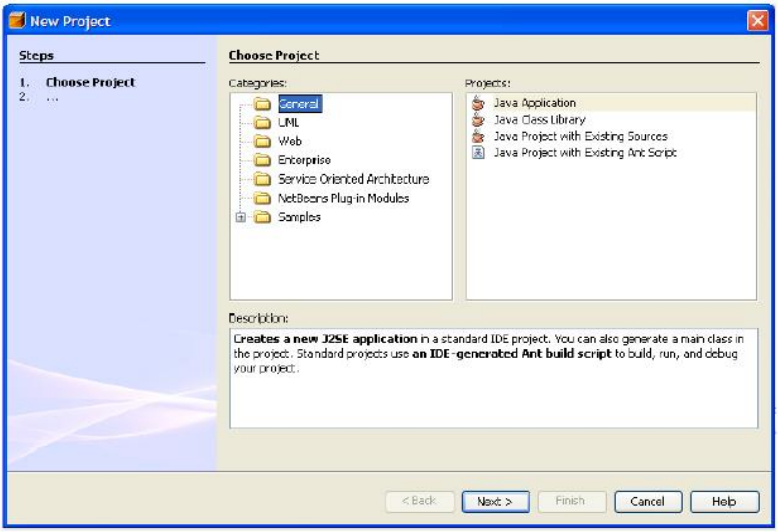

Рис. 3. Створення проекту Java

У наступному вікні (рис. 4) задається ім’я проекту, головного класу. Після чого можна натискати кнопку *Finish*.

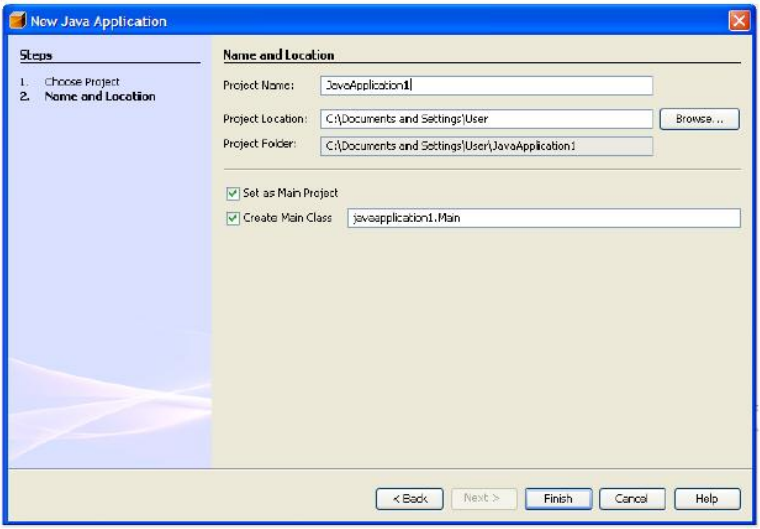

Рис. 4. Створення проекту Java (продовження)

Відкривається вікно проекту Java (рис. 5).

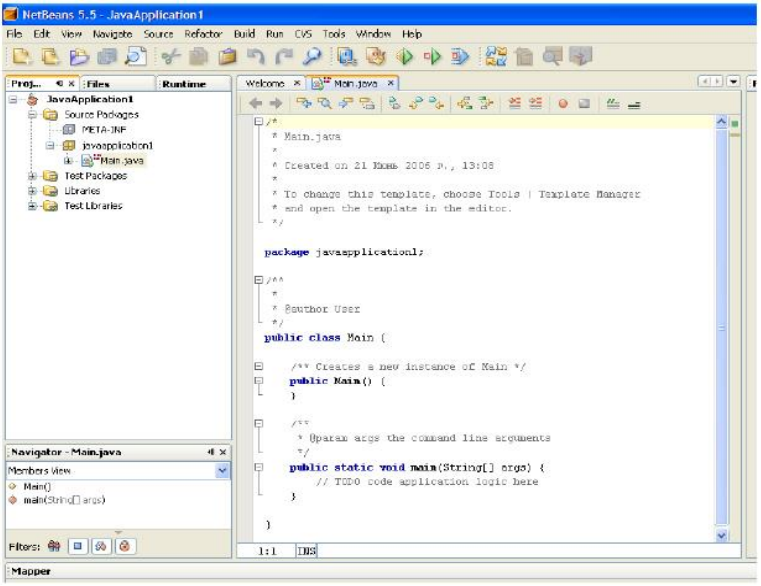

Рис. 5. Проект Java

У лівому верхньому вікні "Projects" показується дерево проектів. У нашому випадку це дерево для проекту JavaApplication1. Це вікно може бути використано для одночасного показу довільного числа проектів. За замовчуванням всі дерева згорнуті.

У правому вікні "Source" розміщений вихідний код проекту.

У лівому нижньому вікні "Navigator" показується список імен членів класу додатка - імена змінних і підпрограм. Подвійне клацання по імені призводить до того, що у вікні редактора вихідного коду відбувається перехід на те місце, де задана відповідна змінна або підпрограма.

Введемо у вікні редактора наступний код:

```
package javaapplication1;

public class JavaApplication1 {

    public static void main(String[] args) {
// TODO code application logic here
    }
}
```

Розглянемо цей код. Перший рядок коду після коментаря – це назва пакету, далі розміщено оголошення класу *JavaApplication1*, який є головним класом додатка.

Всі класи і об'єкти додатка викликаються і управляються з методу main, який виглядає наступним чином:

```
    public static void main (String[] args) {
    }
```

Метод *main* є головним методом додатка. Він автоматично викликається при запуску програми. Параметром *args* цього методу є масив рядків, що має тип String []. Це параметри командного рядка, які передаються до додатоку під час його запуску. Слово String означає "Рядок", а квадратні дужки використовуються для позначення того, що це масив рядків. Для

здійснення методом будь-якої діяльності слід дописати свій власний код.

Напишемо класичний приклад - вивід повідомлення "Привіт!". Для цього замість коментаря *// TODO code application logic here* напишемо рядок виведення тексту:

System.out.println ("Привіт!");

Метод println призначений для виведення тексту в режимі консолі. Після закінчення виконання методу *main* додаток завершує свою роботу.

4. Компіляція файлів проекту і запуск програми

У сучасних середовищах розробки використовується два режими компіляції - compile ("скомпілювати") і build ("побудувати"). У режимі "compile" відбувається компіляція тільки тих файлів проекту, які були змінені в процесі редагування після останньої компіляції. А в режимі "build" перекомпілюються заново всі файли.

Для компіляції проекту слід вибрати в меню середовища розробки *Build / Build Main Project* (або клавіша <F11>, або на панелі інструментів відповідну іконку). При цьому будуть заново скомпільовані все класи проекту.

Пункт *Build / Clean and Build Main Project* (або комбінація клавіш <Shift><F11>) видаляє всі вихідні файли проекту (очищає папки build і dist), після чого заново компілюються всі класи проекту.

Для того щоб запустити скомпільований додаток з середовища розробки, слід вибрати в меню середовища *Run / Run Main Project* (або клавіша <F6>, або на панелі інструментів відповідну іконку). При запуску програма завжди автоматично компілюється (але не "будується"), так що після внесення змін зазвичай досить натиснути <F6>.

Зкомпілюйте та запустіть на виконання наш проект. Після запуску нашого проекту в вихідний консолі, яка знаходиться в нижній частині вікна проекту, з'явиться службова інформація про хід компіляції і запуску (рис. 6, 7):

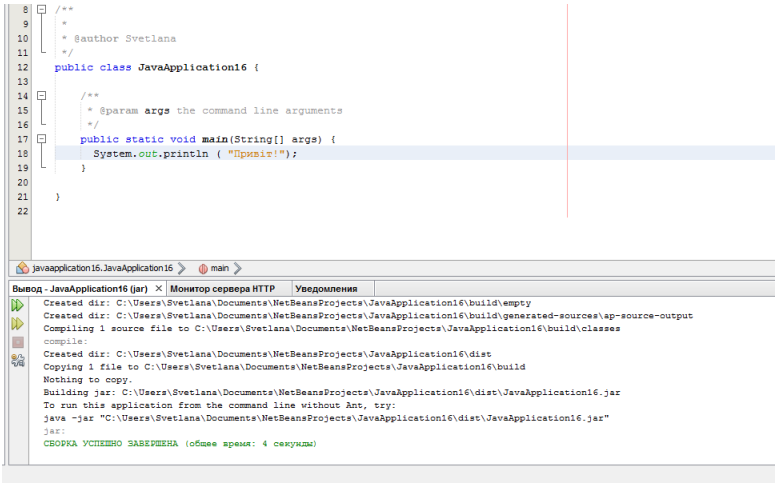

Рис. 6. Результат компіляції проекта

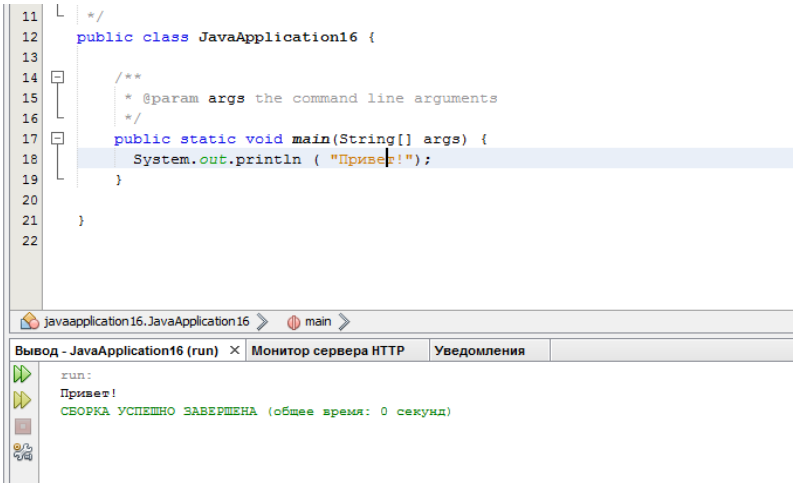

Рис. 7. Результат виконання проекту

Щоб запустити проект в режимі командного рядка, зайдіть в папку dist та наберіть в режимі командного рядка наступний текст:

java -jar "JavaApplication1.jar"

Завдання до лабораторної роботи

1. Виконайте згідно з наданим описом інсталяцію платформи Java (JDK) і інтегрованого середовища розробки NetBeans на свій ПК. Опишіть результати у звіті з розміщенням скріншотів.

2. Створіть свою першу програму на мові Java, в якій буде організований вивід на консоль вашого прізвища, ім'я, по батькові в один рядок, нижче назва спеціальності і курсу .

3. У звіті опишіть структуру створеного проекту (вміст папок), прокоментуйте рядки програми.

4. У звіт включіть код програми, скріншоти.

5. Надайте у звіті відповіді на контрольні питання.До звіту додаються файли проекту.

Контрольні питання

1. Які основні властивості об'єктно-орієнтованої мови притамання Java?

2. Яке призначення мають папки створеного проекту: build, dist, nbproject, src, test?

3. Які інші інтегровані середовища розробки, окрім NetBeans, можна застосовувати для розробки проектів?

ЛАБОРАТОРНА РОБОТА №2

Тема: Команди керування та масиви в Java

Мета: отримати навики побудови алгоритмів розгалужених процесів, навчитися складати алгоритми та програми процесів з обробки масивів.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Загальні поняття

В Java застосовуються наступні керуючи оператори.

Умовний оператор (if-then-else statement) у мові Java записується так:

if (логічний вираз) оператор1 else оператор2

Діє він наступним чином. Спочатку обчислюється логічний вираз. Якщо результат true, то діє оператор1 і на цьому дія умовного оператора завершується (оператор2 не діє, далі буде виконуватися наступний за if оператор). Якщо результат false, то діє оператор2, при цьому оператор1 взагалі не виконується. Умовний оператор може мати скорочену форму (if-then statement):

if (логічний вираз) оператор1

Синтаксис мови не дозволяє записувати декілька операторів ні у гілці then, ні у гілці else. При необхідності створюється блок операторів у фігурних дужках. "Code Conventions" рекомендує завжди використовувати фігурні дужки і розташовувати оператор у декількох рядках з відтступами, як у наступному прикладі:

```
if (a < x) {
x = a + b; } else {
x = a - b;
}
```

Оператор циклу while. Перший оператор циклу — оператор while виглядає так:

while (логічний вираз) оператор

Спочатку обчислюється логічний вираз. Якщо його значення true, то виконується оператор, що утворює цикл. Якщо у цикл потрібно додати декілька операторів, то необхідно створити блок операторів {}. Потім знову обчислюється логічний вираз і діє оператор, і так доти, поки логічний вираз не отримає значення false. Якщо логічний вираз з самого початку дорівнює false, то оператор не буде виконуватися жодного разу. Попередня перевірка забезпечує безпеку виконання циклу, дозволяє уникнути переповнення, ділення на нуль та інші неприємності.

Оператор do-while. Другий оператор цикла — оператор do-while — має вигляд:

do оператор while (логічний вираз)

Спочатку виконується оператор, а потім відбувається обчислення логічного виразу. Цикл виконується, поки логічний

вираз залишається рівним true. У циклі do-while перевіряються умови продовження, а не завершення циклу. Суттєва різниця між цими двома операторами циклу тільки у тому, що у циклі do-while оператор обов'язково виконується хоча б один раз.

Оператор for. Третій оператор циклу — оператор for — виглядає наступним чином:

for (список_Виразів_1; логічний_Вираз;
список_Виразів_2) оператор

Перед виконанням циклу обчислюється список виразів 1. Це нуль або декілька виразів, перерахованих через кому. Вони обчислюються зліва направо, і у наступному виразі уже можна використовувати результат попереднього виразу. Як правило, тут задаються початкові значення змінних циклу. Потім обчислюється логічний вираз. Якщо він true, то діє оператор, потім обчислюються зліва направо вирази із списку виразів 2. Далі знову перевіряється логічний вираз. Якщо він істинний, то виконується оператор і список виразів 2 і т. д. Як тільки логічний вираз стане дорівнювати false, виконання циклу завершується.

Замість списку виразів 1 може стояти одне визначення змінних з початковим значенням. Такі змінні діють лише у межах цього циклу. Будь-яка частина оператора for може бути відсутня: цикл може бути пустим, вираз в заголовку теж, при цьому крапки з комою зберігаються. Хоча в операторі for закладені великі можливості, використовується він, головним чином, для перерахувань, коли відомо їхня кількість.

Оператор варіанта switch. Оператор варіанта організовує розгалуження за декількома напрямами. Кожна гілка позначається константою або константним виразом будь-якого цілого типу (крім long) і вибирається, якщо значення певного виразу співпадає з цією константою. Вся конструкція виглядає наступним чином:

```
switch (Вираз){
case Вираз1: оператор1 case константВир2: оператор2
. . . . .
case ВиразN: операторN
default: операторDef
}
```

Вираз в дужках *Вираз* може бути типу byte, short, int, char, але не long. Цілі числа або цілочисельні вирази, складені із констант, *Вираз1, ... ВиразN* теж не повинні мати тип long.

Оператор варіанта виконується таким чином. Усі константні вирази обчислюються заздалегідь, на етапі компіляції, і повинні мати різні значення. Спочатку обчислюється цілочисельний вираз *Вираз*. Якщо він співпадає з однією із констант, то виконується оператор, позначений цією константою. Потім виконуються усі наступні оператори, включаючи і *операторDef,* і робота оператора switch завершується.

Якщо ж жодна константа не дорівнює значенню виразу, то виконується *операторDef* і всі наступні оператори. Тому гілка default повинна записуватися останньою. Гілка default може бути відсутня, тоді у цій ситуації оператор варіанта взагалі нічого не робить.

Таким чином, константи у варіантах case грають роль тільки міток, точок входу в оператор варіанта, а далі виконуються всі інші оператори згідно з порядком їх запису. Після виконання одного варіанта оператор switch продовжує виконувати всі інші варіанти.

Частіше за все необхідно "пройти" тільки одну гілку операторів. У такому випадку використовується оператор break, який відразу зупиняє виконання оператора switch. Може знадобитися виконувати один і той самий оператор у різних гілках case. У цьому випадку ставимо декілька міток case підряд.

Приклад.

```
switch(dayOfWeek){
case 1: case 2: case 3: case 4:case 5:
System.out.println("Week-day"); break;
case 6: case 7:
System.out.println("Week-end"); break;
 default:
System.out.println("Unknown day");
}
```

Масиви. Масив — це сукупність змінних одного типу, що зберігаються у суміжних комірках оперативної пам’яті.

Масиви у мові Java відносяться до посилальних типів Опис відбувається у три етапи.

Перший етап — *оголошення* (declaration). На цьому етапі визначається тільки змінна типу посилання (reference) на масив та тип масиву, наприклад,

double[] a, b;

Тут визначені дві змінні — вказівники a і b на масиви типу double. Можна поставити квадратні дужки і безпосередньо після імені. Це зручно робити серед визначень звичайних змінних:

int m = 0, ar[], k = -1;

Тут визначені дві змінні цілого типу m і k, і оголошено вказівник на цілочисельний масив ar.

Другий етап — *визначення* (installation). На цьому етапі вказується кількість елементів масиву (довжина), виділяється місце для масиву в оперативній пам'яті, змінна-вказівник одержує адресу масиву. Всі ці дії виконуються ще однією операцією мови Java — операцією *new mun*, що виділяє місце в оперативній пам'яті для об'єкта, вказаного в операції типу, і повертає в якості результату адресу цього місця. Наприклад,

a = new double[5];
b = new double[100];
ar = new int[50];

Індекси масивів завжди починаються з 0. Масив a складається із п'яти змінних a[0], a[1],…, a[4]. Елемента a[5] у масиві немає. Індекси можна задавати будь-якими цілочисельними виразами, крім типу long, наприклад, a[i+j], a[i%5], a[++i]. Виконуюча система Java слідкує за тим, щоб значення цих виразів не виходили за межі довжини масиву.

Третій етап — *ініціалізація* (initialization). На цьому етапі елементи масиву отримують початкові значення. Наприклад,

a[0] = 0.01; a[1] = -3.4; a[2] = 2.89; a[3] = 4.5; a[4] = -6.7;
for (int i = 0; i < 100; i++) b[i] = 1.0 /i;
*for (int i = 0; i < 50; i++) ar[i] = 2 * i + 1;*

Перші два етапи можна поєднати:

double a[] = new double[5], double b[] = new double[100];
int i = 0, int ar[] = new int[50], k = -1;

Можна зразу задати і початкові значення, записавши їх у фігурних дужках через кому у вигляді констант або константних

виразів. При цьому, навіть, необов'язково вказувати кількість елементів масиву, вона дорівнюватиме кількості початкових значень;

double[] a = {0.01, -3.4, 2.89, 4.5, -6.7};

Можна поєднати зазначені етапи:

double a[] = new double[] {0.1, 0.2, -0.3, 0.45, -0.02};

Отже, масиви у Java завжди визначаються динамічно, хоча посилання на них задаються статично.

Крім посилання на масив, для кожного масиву автоматично визначається ціла константа з одним і тим самим ім'ям length. Вона дорівнює довжині масиву. Так, після наших визначень, константа a.length дорівнює 5, константа b.length дорівнює 100, a ar.length дорівнює 50.

Останній елемент масиву a можна записати наступним чином: a [a. length - 1], передостанній — a [a. length - 2] тощо.

Елементи масиву — це звичайні змінні свого типу, з ними можна виконувати всі операції, що припустимі для цього типу. Масив символів у Java не є рядком!

Багатовимірні масиви. Елементами масиву у Java можуть бути масиви. Можна оголосити:

char[] [] c;

що є еквівалентним:

chart[] c[];

або

char c[][];

Потім визначаємо зовнішній масив:

c = new char[3][];

С — масив, який складається з трьох елементів-масивів.

Тепер визначимо його елементи-масиви:

c[0] = new char[2];
c[1] = new char[4];
c[2] = new char[3];

Після цих визначень змінна c.length дорівнює 3, c[0].length дорівнює 2,c[l].length дорівнює 4 і c[2].length дорівнює 3. Нарешті, задаємо початкові значення c [0] [0] = 'a', [0][1] = 'r',c[1][0] = 'h',c[1][1] = 'a',c[1][2] = 'y' тощо.

Опис можна скоротити:

int[] [] d = new int[3] [4];
А початкові значення задати наступним чином:
int[] []d = {{I, 2, 3}, {4, 5, 6}};

Порядок виконання роботи

1. Ознайомтеся з теоретичним матеріалом.
2. Виконайте індивідуальне завдання відповідно до вашого варіанту.
3. Підготуйте звіт, в якому повинні бути алгоритм, код програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Дана матриця A (n,n), де n – непарне число, а елементи матриці не повторюються. Знайти найбільший елемент серед всіх, що стоять на головній і побічній діагоналях і поміняти його місцями з елементом, що стоїть на перетині цих діагоналей.

2. Знайти максимальний елемент серед тих рядків матриці A (n,m), елементи яких впорядковані за зростанням.

3. Знайти мінімальний елемент серед тих стовпців матриці A (n,m), елементи яких впорядковані за убиванням.

4. З матриці A (n,m) отримати нову матрицю, викресливши з вихідної матриці рядок і стовпець, на перетині яких знаходиться перший максимальний елемент.

5. Заповнити матрицю A (n,n) по спіралі масивом натуральних чисел 1, 2, ..., nxn, починаючи з елемента, що знаходиться у верхньому лівому кутку.

6. Задано натуральне число k і двовимірний масив дійсних чисел A (n,m). Помножити на введене натуральне число k всі елементи матриці, що розташовані у рядках з парними номерами та стовпцях з непарними номерами, але індекси цих елементів повинні бути менші за індекси першого максимального елементу матриці.

7. Характеристикою рядка цілочисельної матриці A (n,m) назвемо суму її додатних парних елементів. Переставляючи рядки заданої матриці, розташувати їх відповідно до зростання їх характеристик.

8. Характеристикою стовпця цілочисельної матриці A (n,m) назвемо суму модулів його від'ємних непарних елементів. Переставляючи стовпці заданої матриці, розташувати їх відповідно до зростання їх характеристик.

9. У матриці A (n,n) елементи кожного рядка верхнього трикутника впорядкувати по зростанню, нижнього - за спаданням, діагональні - залишити без змін.

10. Для заданої матриці A (n,n) знайти максимум серед сум елементів діагоналей, паралельних головній діагоналі, і мінімум серед сум модулів елементів діагоналей, паралельних побічної діагоналі.

Контрольні питання

1. Чим ітераційний цикл відрізняється від умовних?
2. Коректний чи ні наступний код? Якщо ні, то дайте пояснення, в яких місцях і чому будуть виникати помилки:

```
int b[]=new int[5];
for (int i=1; i<=b.length(); i++) {
b[i]=Math.sqrt(i);
}
```

ЛАБОРАТОРНА РОБОТА №3

Тема: Обробка рядків. Використання регулярних виразів в Java-додатках

Мета: придбання навичок обробки рядків та розробки рядових шаблонів у мови Java.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Робота з рядками (класи String і StringBuffer)

Для зберігання і обробки рядків в Java є два класи: String – для незмінних рядків, і StringBuffer - для рядків, що можуть змінюватися. Обидва класи поширюють клас Object. Вони

знаходяться в пакеті java.lang, тому їх не треба підключати за допомогою оператора import.

Створення та ініціалізація об'єкта класу String

Ініціалізація об'єкта класу String може виконуватися за допомогою присвоювання змінної класу String строкової змінної або строкового літерала, наприклад:

String str = "Рядок 1";

або при створенні об'єкта за допомогою оператора new з використанням одного з *конструкторів* класу String, формати яких наведено нижче:

String ()- порожній рядок;

String (String рядок-1) – новий рядок, який є копією рядка-1.

String (char [] масив) – рядок, створений з елементів масиву.

String (char [] масив, int початковий-індекс, int довжина) – рядок, створений з фрагменту масиву символів, що починається з позиції *початковий-індекс* і заданої довжини;

String (byte [] масив)- рядок, створений з масиву байт, з використанням кодування за замовчуванням (для російської мови в Windows - кодування Windows-1251);

String (byte [] масив, int початковий-індекс, int довжина)- рядок, створений з фрагменту масиву байт, що починається з позиції *початковий-індекс* і заданої довжини.

Нижче наведено деякі приклади застосування конструкторів класу String.

Конструктор створення пустого рядка. В цьому випадку конструктору аргументи не передаються

String s = new String ();

Конструктор створення текстового рядка на основі символьного масиву.

char symbols [] = { 'a', 'b', 'c'};

String s = new String (symbols);

У цьому конструкторі, крім імені масиву, можна вказати індекс елемента масиву, починаючи з якого буде вилучатися рядок, а також довжину рядка в символах:

char symbols = { 'a', 'b', 'c', 'd', 'e', 'f'};

String s = new String (symbols, 2,3); // s = "cde"

Конструктор копіювання об'єкта.

```
String obj = new String ( "Текстовий рядок");
String s = new String (obj);
```

Деякі методи класу String

Значення об'єктів класу String після їх створення не можуть бути змінені. Це накладає деякі обмеження на методи роботи з рядками. Наприклад, щоб змінити вже існуючий текст, необхідно створювати новий об'єкт.

Довжина рядка може бути визначена за допомогою методу:

```
public int length ()
```

Єдина операція, яку можна використовувати для рядків, є операція зчеплення (конкатенація) двох або більше рядків - "+". Наприклад:

```
String str="Івану Петрову "+18+" років";
```

Вилучення символів з рядка

char charAt (int index) - повертає символ, що знаходиться за вказаним індексом у рядку. Результатом роботи методу буде символ типу char.

```
String str = "Останній символ у рядка - о";
int last = str.length()-1;//потрібно виконати операцію:
//довжина рядка - 1, тому що відлік починається з  0
char ch = str.charAt(last);
System.out.println(ch);
```

Вилучення підрядка з рядка

String substring (int beginIndex, int endIndex) або *substring (int beginIndex)* - повертає новий рядок, що є підрядком використовуваного рядка. У параметрах методу потрібно вказати індекс рядка, з якого починається підрядок, і індекс, яким закінчується. Також можливо вказувати тільки початковий індекс. У цьому випадку буде повернуто підрядок від початкового індексу і до кінця рядка.

```
String s = "www.mysite.com";
String name = s.substring (4, s.length () - 4);
System.out.println (name); // на консоль виведе "mysite"
String domain = s.substring (4);
```

```
System.out.println (domain); // на консоль виведе
//"mysite.com"
```

Пошук в рядку

boolean contains (CharSequence s) - перевіряє, чи містить рядок задану послідовність символів і повертає true або false.

```
String s = "www.mysite.com";
boolean isContain1 = s.contains ("mysite");
System.out.println (isContain1); // знайшов - виведе true
boolean isContain2 = s.contains ("mysite.ru");
System.out.println (isContain2); // не знайшов - виведе false
```

Порівняння рядків

boolean equals (Object anObject) - перевіряє ідентичність рядків. Повертає true тільки в тому випадку, якщо в рядках представлена однакова послідовність символів однієї величини.

```
String str = "ПРИКЛАД";
String str2 = "приклад";
// рядки не ідентичні
System.out.println (str.equals (str2)); // false
// рядки ідентичні після зміни регістру у першому рядку
System.out.println (str.toLowerCase (). Equals (str2)); // true
```

Метод toString()

Метод toString () визначено в класі Object, що знаходиться на вершині ієрархії класів Java (це загальний суперклас Java). Метод викликається за замовчуванням при перетворенні об'єкта в текстовий формат.

Приклад 1. Перевизначення методу toString()

```
class ComplNums{
// Поля класу:
double Re;
double Im;
// Конструктор:
    ComplNums(double x,double y){
    Re=x;
    Im=y;}
// Перевизначення методу toString():
    public String toString(){
    String result="",sign="",ImPart="",RePart="";
    if(Re!=0||(Re==0&&Im==0)) RePart+=Re;
```

```
        if((Im>0)&&(Re!=0)) sign+="+";
        if(Im!=0) ImPart+=Im+"i";
        result=RePart+sign+ImPart;
        return result;}
        }
class toStringDemo{
    public static void main(String[] args){
        for(int i=1;i<=3;i++){
            for(int j=1;j<=5;j+=2){
                ComplNums z=new ComplNums(i-2,j-3);
                // Автоматичний виклик методу toString():
                System.out.println(z); }
            } } }
```

У програмі створюється клас для реалізації комплексних чисел. У класі ComplNums є два поля Re і Im типу double. Це дійсна і уявна частини комплексного числа. Конструктор класу приймає два аргументи - значення полів Re і Im.

Крім цього, в класі перевизначений метод toString (). Метод, що описаний як public і не має аргументів, повертає об'єкт класу String. На метод toString () покладено обов'язок сформувати текстове представлення комплексного числа з заданими дійсної і уявною частинами. Результат методу попередньо записується в об'єктну змінну result класу String. Змінна result, в свою чергу, представляється як об'єднання трьох текстових рядків: RePart (текстове представлення дійсної частини комплексного числа), ImPart (текстове представлення уявної частини комплексного числа з урахуванням уявної одиниці i) і sign (знак між дійсною і уявною частинами комплексного числа).

Звертаємо увагу на спосіб відображення текстового представлення об'єкту z. Використовується команда System.out.println (z), тобто об'єкт вказується аргументом методу println (). Метод toString () викликається автоматично при спробі перетворити об'єкт z в текстовий формат (в об'єкт класу String).

Створення та ініціалізація об'єкта класу StringBuffer

Клас StringBuffer схожий на клас String, але рядки, створені за допомогою цього класу можна модифікувати, тобто їх вміст і довжина можуть змінюватися. При зміні рядка класу StringBuffer програма не створює новий строковий об'єкт, а

працює безпосередньо з вихідним рядком, тобто всі методи оперують безпосередньо з буфером, що містить рядок. Тому клас StringBuffer зазвичай використовується, коли рядок доводиться часто модифікувати зі зміною її довжини.

Конструктори

StringBuffer визначає чотири конструктора:

StringBuffer () - створення пустого рядка, буфер дорівнює 16 символам.

StringBuffer (int capacity) - створення пустого рядка, буфер дорівнює capacity символам.

StringBuffer (String str) - створення копії рядка на основі існуючої рядка, буфер дорівнює 16 символам.

StringBuffer (CharSequence chars) - створення рядка на основі набору символів, буфер дорівнює 16 символам.

Деякі методи StringBuffer

Метод *length ()* дозволяє отримати поточну довжину об'єкта:

```
StringBuffer strbuf = new StringBuffer ("Java");
System.out.println ("Довжина:" + strbuf.length ()); // 4
```

Метод *capacity ()* дозволяє отримати поточний обсяг виділеної пам'яті:

```
StringBuffer strbuf = new StringBuffer ("Java");
System.out.println ("Ємність:" + strbuf.capacity ());
// 16 + 4 = 20
```

StringBuffer ініціалізується рядком "Java", тому його ємність становить 4 + 16 = 20 символів.

Метод *setCharAt (int index, char ch)* встановлює нове значення символу з зазначенням індексу символу і його значенням:

```
StringBuffer strbuf = new StringBuffer ("Java");
strbuf.setCharAt (0, 'j'); // j
```

Додавання підрядка. У класі stringBuffer є десять перевантажених методів append (), що додають підрядок в кінець рядка. Ось формат типового методу:

```
append (String str)
```

Метод приєднує рядок str в кінець цього рядка.

```
StringBuffer strbuf;
strbuf = new StringBuffer ();
```

```
strbuf.append ("Центр");
strbuf.append ("« Турбо »");
System.out.println ("strbuf =" + strbuf);
// strbuf = Центр «Турбо»
```

Вставити підрядок

```
insert (int index, string str)
```

Метод вставляє рядок str до цього рядка перед символом з індексом index.

```
strbuf.insert (6, "комп'ютерного навчання");
System.out.println ( "strbuf =" + strbuf);
// strbuf = Центр комп'ютерного навчання  «Турбо»
// ще варіант
String sb = new StringBuffer ( "Центр «Турбо »").
Insert(6, "комп'ютерного навчання") .toString ();
System.out.println ( "sb =" + sb);
// sb = Центр комп'ютерного навчання «Турбо»
```

Видалити підрядок

```
delete (int begin, int end)
```

Метод видаляє з рядка символи починаючи з індексу begin включно до індексу end-1:

```
strbuf.delete (6, 29);
System.out.println ( "stbuf =" + strbuf);
// stbuf = Центр «Турбо»
```

Видалити символ

```
deleteCharAt (int ind)
```

Метод видаляє символ з вказаним індексом ind.

```
StringBuilder strCh = new StringBuilder ();
strCh.append ("ADC-DRF");
strCh.deleteCharAt (3);
System.out.println ("strCh =" + strCh);
// strCh = ADCDRF
```

Замінити підрядок

```
replace (int begin, int end, String str)
```

Метод видаляє символи з рядка, починаючи з індексу begin до індексу end-1 і вставляє замість них рядок str.

```
StringBuilder strCh = new StringBuilder ();
strCh.append ("ADC-DRF");
String str3 = "- літери англійського алфавіту";
```

```
strCh.replace (3, 7, str3);
System.out.println ("strCh =" + strCh);
// strCh = ADC- букви англійського алфавіту
```

Регулярні вирази

Регулярні вирази (Regular Expressions) дозволяють зіставляти текст зі зазначеним шаблоном, а також виконувати заміну тексту. Ці операції здійснюються за допомогою універсальних символів, які спеціальним чином інтерпретуються. При створенні рядка-шаблону регулярного виразу застосовуються метасимволи (^,$,.,|,&,+,*,\\d,\\s,\\w,\\D,\\S,\\W – основні з них)

Нижче наведено опис деяких комбінацій метасимволів, які мажна використовувати для регулярних виразів в мові програмування Java.

Таблиця 1. Комбінації метасимволів

Вираз	**Опис**
^	Відповідає початку рядка.
$	Відповідає кінцю рядка.
[...]	Відповідає будь-якому знаку в квадратних дужках.
[^...]	Відповідає будь-якому знаку за межами дужок.
(a) | (b)	Відповідає a або b.
\w	Відповідає словесним символам.
\s	Відповідає пробілу. Еквівалент [\ t \ n \ r \ f].
\S	Відповідає символу окрім пробілу.
\d	Збігається з цифрою. Еквівалент [0-9].
\D	Відповідає нечисловому символу.
\A	Відповідає початку рядка.

\z	Відповідає кінцю рядка.

Квантори регулярних виразів:

\+ позначає збіг один раз або більше;

* позначає збіг нуль або раз більше;

? позначає збіг нуль або один раз.

Java надає пакет java.util.regex для роботи з регулярними виразами. Всі функціональні можливості представлено двома класами (Matcher, Pattern), інтерфейсом (MatchResult) і винятком (PatternSyntaxException).

Клас *Pattern* є скомпільоване уявлення регулярного виразу. Клас не має публічних конструкторів, тому для створення об'єкта даного класу необхідно викликати статичний метод compile і передати в якості першого аргументу рядок з регулярним виразом:

```
Pattern pattern = Pattern.compile("^<([a-z]+)([^>]+)*(?:>(.*)<\/\1>|\s+\/>)$");
```

Matcher - клас, який реалізує механізм порівняння рядка з регулярним виразом, зберігає результати цього порівняння (використовуючи реалізацію методів інтерфейсу MatchResult). Не має публічних конструкторів, тому для створення об'єкта цього класу потрібно використовувати метод matcher класу Pattern:

```
// будемо шукати URL
String regexp = "^ (https?: \\ / \\ /)? ([a-z \\ d \\ .-] +) \\. ([Az \\.] {2,6}) ([\ \ / \\ w \\ .-] *) * \\ /? $ ";
String url = "http://habrahabr.ru/post/260767/";
Pattern pattern = Pattern.compile (regexp);
Matcher matcher = pattern.matcher (url);
```

Приклад валідації e-mail адреси

Регулярні вирази можуть бути дуже корисні при server- і client-side валідації даних. Розглянемо невеликий приклад. Припустимо, необхідно перевірити коректність e-mail адреси:

```
import java.util.regex. *;

public class TestRegexp {
```

```
    public static final Pattern pattern = Pattern.compile
  ("[a-zA-Z\\d\\u002E\\u005F]+@([a-zA-
Z]+\\u002E){1,2}((net)|(com)|(org))");
    public static void doMatch (String word) {
      Matcher matcher = pattern.matcher (word);
      System.out.println ( "Validation for" + word +
          (Matcher.matches ()? "Passed.": "Not passed."));
    }

    public static void main (String [] args) {
      doMatch ( "c0nst@money.simply.net");
      doMatch ( "Name.Sur_name@gmail.com");
      doMatch ( "useR33@somewhere.in.the.net");
    }
  }
```

Послідовність виду [a-zA-Z] вказує на безліч, в нашому випадку - це безліч латинських символів у верхньому і нижньому регістрах. Символ \ d вказує на безліч цифр. "\ U002E" і "\ u005F" - це символи точки і підкреслення відповідно. Знак плюс після деякої послідовності говорить про те, що вона повинна зустрінеться один або більше разів. { n } говорить про те, що деякий символ повинен зустрінеться n раз, а {n, m} - від n до m раз. "|" - уявлення логічного "або".

У нашому прикладі під Pattern будуть підходити ті e-mail адреси, які починаються з літери, містять літери, цифри, крапку і підкреслення до символу "@" і знаходяться в доменах com, net, org (не більше третього рівня).

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити програми до двох завдань: а) - з обробки рядків, б) – з використанням регулярних виразів (відповідно варіанту).

3. Підготувати звіт, в якому повинні бути коди програм з коментарем та скріншоти опрацювання програм, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

Варіант 1. а). За правилами машинопису після коми в тексті завжди ставиться символ пробілу. Скласти програму виправлення такого типу помилок (відсутність пробілу) в тексті. б). Дан текстовий рядок, що містить кілька слів. Вивести всі слова, що починаються з приголосних букв українського алфавіту.

Варіант 2. а) Дан рядок, що складається з декількох слів. Вивести на екран тільки ті слова, що починаються з великої літери. б). Знайти та підрахувати кількість всіх послідовностей символів «авто» в різних контекстах, як наприклад: «авто», «автомат», «автовокзал», «тавтологія», «мініавто» і т. ін.

Варіант 3. а). Дано два рядки. Вивести літери, що зустрічаються в обох рядка. б). Знайти у тексті всі слова «дім» та замінити їх на «будинок» (передбачити ситуацію, коли «дім» може бути складовим іншого слова – тоді заміну не робити).

Варіант 4. а). Розробити програму, що встановлює, чи є слово паліндром (слово, яке однаково читається зліва направо і справа наліво: дід, око, зараз і т.д. б). Скласти програму, що визначає, чи є заданий рядок IP адресою. Приклад правильних виразів: 127.0.0.1; 255.255.255.0. Приклад неправильних виразів: 1300.6.7.8, abc.def.gha.bcd.

Варіант 5. а). Дан текстовий рядок. Визначити частку (в відсотках) літери «а» в ньому. б). Дан рядок – номер телефону у форматі +38(xxx)уууууuу, де ххх - код мобільного оператора. Визначити правильність запису номера телефона. Приклад правильних телефонів: +38(050)1111111. Приклад неправильних телефонів: +38(050)ф111111.

Варіант 6. а). Дано слово. Вставити задану букву після першої літери «і». б). Дан рядок – номер телефону у форматі +38(xxx)уууууuу, де ххх - код українського мобільного оператора. Визначити, якому мобільному оператору відповідає номер телефону.

Варіант 7. а). Дано слово. Переставити його першу букву на місце останньої. б). Дан рядок, який містить числа, записані у різних системах числення (двійкова, десяткова, шістнадцяткова тощо). Визначити числа, що записані у двійковій системі числення, вивести їх на екран у двійковій та десятковій формі (наприклад: правильна відповідь: 100 (двійкова) – 4 (десяткова)).

Варіант 8. а). Дано слово. Видалити з нього другу за рахунком букву «о», якщо така є (наприклад: молоко → молко). б). Дано декілька рядків. Вивести на екран рідки, що містять цілі числа як окремі слова. Приклад рядків, що підходять: «Year is 2009.», «1 is a number». Приклад рядків, що не підходять: «Not2Bad», «No digits here», «3.5423».

Варіант 9. а). Дано речення. Замінити в ньому всі сполучники «і» на «та». б). Дан рядок, що містить число. Визначити, чи є цей рядок шістнадцятковим ідентифікатором кольору в HTML: #FFFFFF для білого, # 000000 для чорного, # FF0000 для червоного і т.д. Приклад правильних виразів: #FFFFFF, # FF3421, # 00ff00. Приклад неправильних виразів: 232323, f # fddee, # fd2.

Варіант 10. а). Дано слово. Додати до нього на початку і в кінці стільки зірочок, скільки літер в цьому слові. б). Дан рядок, який символізує пароль. Перевірити, чи надійно складений пароль. Пароль вважається надійним, якщо він складається з 8 або більше символів. Де символом може бути англійська літера, цифра і знак підкреслення. Пароль повинен містити хоча б одну велику літеру, одну маленьку букву і одну цифру. Приклад правильних виразів: C00l_Pass, SupperPas1. Приклад неправильних виразів: Cool_pass, C00l.

Контрольні питання

1. Які види параметрів можуть бути задані в конструкторах класу String?
2. Які методи порівняння рядків визначені в класі String?
3. Які методи класу String виконують пошук в рядках?
4. В розв’язанні яких задач доцільно використовувати регулярні вирази?

ЛАБОРАТОРНА РОБОТА №4
Тема: Класи та об’єкти в Java. Абстрактні класи та інтерфейси
Мета: навчитися опрацьовувати класи Java, абстрактні класи та інтерфейси з застосуванням принципів ООП.
Час виконання роботи: 4 години.
Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Класи

Java є об'єктно-орієнтованою мовою, тому такі поняття як "клас" і "об'єкт" грають ключову роль. Кожен об'єкт визначається деяким загальним шаблоном, який називається класом. Тобто об'єкт – це екземпляр класу. В рамках класу задається загальний шаблон, тобто структура, на основі якої потім створюються об'єкти. Дані, що відносяться до класу, називаються полями класу, а програмний код для їх обробки - методами класу.

Статичні і динамічні елементи класу. Якщо при визначенні елемента не використовується ключове слово static, то цей елемент за замовчуванням є динамічним (dynamic).

Динамічні методи і змінні завжди є елементами об'єкта класу, і доступ до них здійснюється через змінну-об'єкт.

ІмяОб'єкта.ІмяЗмінноїЕкземпляра
ІмяОб'єкта.ІмяМетодуЕкземпляра (<Параметри>)

Статичні методи і змінні пов'язані з класом, а не з екземпляром класу, і є елементами класу. Їх можна використовувати без створення об'єкта цього класу, доступ здійснюється через ім'я класу.

ІмяКласу.ІмяЗмінноїКласу
ІмяКласу.ІмяМетодуКласу (<Параметри>)

Статичні методи класу можуть працювати тільки зі статичними змінними класу. Ці елементи класу використовуються всіма об'єктами класу (для всіх об'єктів існує лише один екземпляр статичної змінної).

Приклад 1. Файл TestElements.java

```
class StaticAndDynamic {
    int i=0;            // змінна екземпляра
```

```
    static int j=0;      // змінна класу
    // статистичні методи
    static void setJ(int k)     {
        System.out.println("Static Method"); j=k;
    }
    static int getJ()   {
        System.out.println("Static Method"); return j;
    }
    // динамічні методи
    void setI(int k)    {
        System.out.println("Dynamic Method"); i=k;
    }
    int getI()          {
        System.out.println("Dynamic Method"); return i;
    }
    int summa()  {
    // в динамічних методах можуть використовуватися
    //статистичні змінні
        System.out.println("Dynamic Method"); return i+j;
    }
}
public class TestElements    {
    public static void main(String args[])     {
         int ii,jj;
        // застосування статистичної змінної
         StaticAndDynamic.j=6;     jj=StaticAndDynamic.j;
        System.out.println("Main, jj="+jj);
        // виклик статистичних методів
        StaticAndDynamic.setJ(4);
        jj=StaticAndDynamic.getJ();
        System.out.println("Main, jj="+jj);
        StaticAndDynamic obj=new StaticAndDynamic();
        // застосування динамічної змінної
        obj.i=3;     ii=obj.i;
        System.out.println("Main, ii="+ii);
        // виклик динамічних методів
        obj.setI(8);     ii=obj.getI();
        System.out.println("Main, ii="+ii);
```

```
        ii=obj.summa();
        System.out.println("Main, summa="+ii);
    }
}
```

Модифікатори доступу. Модифікатори доступу використовуються для управління доступом до елементів класу з інших частин програми (в інших класах).

Елемент, оголошений з ключовим словом public (відкритий), доступний у всіх класах, як в своєму пакеті, так і у всіх класах в будь-якому іншому пакеті. Цей модифікатор можна використовувати при оголошенні класу - тоді цей клас доступний для всіх класів інших пакетів. В кожному файлі повинен міститися лише один public-клас і ім'я файлу повинне відповідати імені такого класу.

Елемент, оголошений з модифікатором protected (захищений) в деякому класі A, доступний у всіх класах, що є підкласами класу A.

Модифікатор private (закритий) більш всього обмежує доступ до елемента. Він його робить невидимим за межами даного класу. Навіть підкласи даного класу не зможуть звертатися до елементу, оголошеному як private.

Якщо тип доступу до елементу не вказано (доступ за замовчуванням), то він доступний з усіх класів, що входять до даного пакету.

Приклад 2. Файл TestModificators.java

```
class A       {
    private int n;
    A() { k=2; n=11; }
    int summa() { return k+n; }
    public int getN() { return n; }
    public void setN(int nn) { n=nn; }
}

class TestModificators       {
    public static void main(String args[])     {
        A obj=new A();          // створення объекта класу A
        // отримати значення змінних
        int kk=obj.k;    System.out.println("k="+kk);
```

```
        int nn=obj.getN();      System.out.println("n="+nn);
        obj.k=10;       obj.setN(15);
        int s=obj.summa();      System.out.println("summa="+s);
    }
}
```

Спадкування класів. Спадкування (inheritance) спрощує практичне використання класів, так як дозволяє розширювати вже написанні та налагоджені класи, шляхом додавання до них нових властивостей і можливостей. Таким чином створюється те, що називається підкласом початкового класу. Клас, який при цьому успадковується (розширюється), називається суперкласом. При розширенні класу є можливість використання всіх написаних і налагоджених методів цього класу. Ця властивість є однією з головних переваг об'єкт-но-орієнтованого програмування.

Для реалізації спадкування вказується ключове слово extends. Наприклад, оголошення класу MyClass підкласом класу SuperClass, можна записати так:

```
class MyClass extends SuperClass {..}
```

Абстрактні класи

Крім звичайних класів в Java є абстрактні класи. Абстрактний клас схожий на звичайний клас. В абстрактному класі також можна визначити поля і методи, в той же час не можна створити об'єкт або екземпляр абстрактного класу. Абстрактні класи покликані надавати базовий функціонал для класів-спадкоємців. А похідні класи вже реалізують цей функціонал.

У цьому випадку в описі класу перед словом class повинен стояти abstact і при описі нереалізованих методів теж повинен використовуватися цей модифікатор. Наприклад:

```
public abstract class D {
int g1 (int s) {. . . }
public abstract g2 (String str); . . .}
```

У абстрактному класі D метод g1 - це звичайний метод, g2 - абстрактний, він містить тільки заголовок, але не містить реалізації. Як видно з прикладу, тіло абстрактного методу відсутнє.

Абстрактний клас не може використовуватися безпосередньо для породження об'єктів. Для цього необхідно породити інший клас на основі абстрактного класу, який буде базовим. В новому класі потрібно визначити всі абстрактні методи. Тоді можна буде створювати об'єкти.

З іншого боку не заборонено описувати змінні абстрактного класу. Просто їм потрібно присвоювати посилання на об'єкти неабстрактних класів.

Інтерфейси

Поняття інтерфейсу схоже на абстрактний клас. Інтерфейс - це повністю абстрактний клас, який не містить ніяких полів, крім констант (static final - поля).

Існує, однак, серйозна відмінність інтерфейсів від класів взагалі і від абстрактних класів, зокрема. Інтерфейси допускають множинне спадкування. Тобто один клас може задовольняти кільком інтерфейсам відразу.

Це пов'язано з тим, що інтерфейси не породжують проблем з множинним спадкуванням, оскільки вони не містять полів. Синтаксис:

```
public interface Stack {
int size = 100;
 int f (String s);
}
```

Це опис інтерфейсу Stack. Усередині дужок можуть перебувати тільки описи методів (без реалізації) і описи констант (static final - полів). В даному випадку інтерфейс Stack містить, зокрема метод f (...).

У наступному прикладі клас R реалізує інтерфейси Serializable і Stack.

```
public class R implements Serializable, Stack {
}
```

Усередині класу, що реалізовує деякий інтерфейс, повинні бути реалізовані всі методи, описані в цьому інтерфейсі. Оскільки Stack має метод f (...), то в класі R він повинен бути реалізований:

```
public class R implements Serializable, Stack {
public int f (String s) {...} ..
```

.}

В інтерфейсі f (...) описаний без модифікатора public, в класі R з public. Справа в тому, що всі методи інтерфейсу за замовчуванням вважаються public, так що цей модифікатор там можна опустити. А в класі R необхідно його використовувати явно. Ще одним загальним моментом інтерфейсів і абстрактних класів є те, що хоча і не можна створювати об'єкти інтерфейсів, але можна описувати змінні типу інтерфейсів.

Інтерфейси широко використовуються при написанні різних стандартів. Стандарт визначає, зокрема, набір якихось інтерфейсів.

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити першу програму з використанням успадкування суперкласу.
3. Розробити другу програму з використанням абстрактних класів або інтерфейсів (вказано в дужках у варіанті завдань до лабораторної роботи).
4. Підготуйте звіт, в якому повинні бути код програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файлі проекту.

Варіанти завдань до лабораторної роботи

1. Створити суперклас (абстрактний клас) Транспортний засіб і підкласи Автомобіль, Автобус, Маршрутне таксі. Задати кількість місць для пасажирів, ціни квитка. Порахувати вартість перевезення пасажирів за 1 рейс (з урахуванням повного завантаження транспорту) кожним транспортним засобом.

2. Створити суперклас (абстрактний клас) Вантажоперевізник і підкласи Літак, Поїзд, Автомобіль. Задати вантажопідйомність, ціну перевезення 1 умовної одиниці вантажу. Порахувати вартість перевезення вантажу за 1 рейс (з

урахуванням повного завантаження транспорту) кожним транспортним засобом.

3. Створити суперклас (абстрактний клас) Учень і підкласи Школяр і Студент. Задати масив предметів та оцінок. Розрахувати середній бал успішності.

4. Створити суперклас (абстрактний клас) Працівник фірми і підкласи Менеджер, Аналітик, Програміст, Тестувальник, Дизайнер. Задати кількість відпрацьованих днів та заробітну ставку за день. Вивести зарплату кожного співробітника.

5 Створити суперклас (абстрактний клас) Садове дерево і похідні класи Яблуня, Вишня, Груша та інші. Прийняти рішення про пересадку кожного дерева в залежності від віку дерева.

6. Реалізувати наступну схему: суперклас (інтерфейс) Лікар←клас Хірург←клас Нейрохірург.

7. Реалізувати наступну схему: суперклас (інтерфейс) Навчальний заклад, класи-спадкоємці Коледж та Університет.

8. Реалізувати наступну схему: суперклас (інтерфейс) Співробітник, класи-спадкоємці Інженер та Бригадир.

9. Реалізувати наступну схему: суперклас (інтерфейс) Корабель←клас Вантажний корабель←клас Танкер.

10. Реалізувати наступну схему: суперклас (інтерфейс) Книга, класи-спадкоємці Довідник та Енциклопедія.

Контрольні питання

1. Чи може клас не мати жодного конструктору?
2. Чи буде компілюватися наступний код, якщо обидва класи будуть оголошені у файлі Test.java?

```
//Test.java
public class Car{
   public String myCar = "Ferrari";
}
public class Test{
   public static void main(String ... args){
      Car myCar = new Car();
```

```
        System.out.println(myCar.myCar);
    }
}
```

ЛАБОРАТОРНА РОБОТА №5
Тема: Винятки. Обробка винятків
Мета: опанувати принципи обробки виняткових ситуацій у Java.
Час виконання роботи: 4 години.
Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Винятками або винятковими ситуаціями (станами) називаються помилки, що виникли в програмі під час її роботи.

Всі винятки в Java є об'єктами. Тому вони можуть породжуватися не тільки автоматично при виникненні виняткової ситуації, але й створюватися самим розробником.

Ієрархія класів винятків продемонстрована на рис. 8:

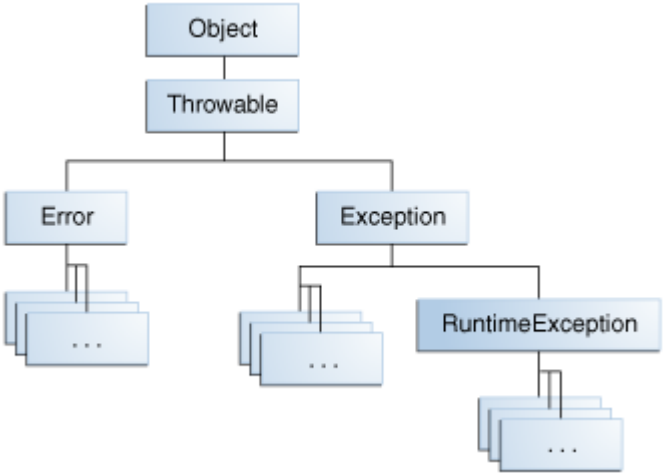

Рис. 8. Ієрархія винятків

Винятки поділяються на кілька класів, але всі вони мають спільний батьківський клас - Throwable. Його нащадками є підкласи Exception і Error.

Винятки (Exceptions) є результатом проблем в програмі, які в принципі можна вирішити і передбачити. Наприклад, такою проблемою може бути ділення на нуль.

Помилки (Errors) представляють собою більш серйозні проблеми, які, відповідно до специфікації Java, не слід намагатися обробляти у власній програмі, оскільки вони пов'язані з проблемами рівня JVM. Наприклад, виключення такого роду виникають, якщо закінчилася пам'ять, що доступна віртуальній машині.

У Java всі винятки діляться на три типи: контрольовані винятки (checked) і неконтрольовані винятки (unchecked), до яких відносяться помилки (Errors) і винятки часу виконання (RuntimeExceptions, нащадок класу Exception).

Контрольовані винятки - помилки, які можна і потрібно обробляти в програмі. До цього типу належать усі нащадки класу Exception (але не RuntimeException).

Обробку винятків може бути здійснювати за допомогою операторів try ... catch, або передавати зовнішньої частини програми. Наприклад, метод може передавати винятки, що виникли в ньому, вище за ієрархією викликів, при цьому цей метод його не обробляє.

Неконтрольовані винятки не вимагають обов'язкової обробки, проте, при бажанні, можна обробляти винятки класу RuntimeException.

Існує п'ять ключових слів, які використовуються у винятки: try, catch, throw, throws, finally. Порядок обробки винятків наступний.

Оператори програми, які ви хочете відслідковувати, розміщуються всередині блоку try. Якщо виняткова ситуація відбулося, то створюється виняток і передається далі. Ваш код може перехопити виняток за допомогою блоку catch і обробити його. Системні винятки автоматично передаються самою системою. Щоб передати виняток вручну, використовується throw. Будь-який виняток, створений і переданий всередині методу, має бути позначений в його інтерфейсі ключовим словом

throws. Будь-який код, який слід виконати обов'язково після завершення блоку try, поміщається у блок finally

Схематично код виглядає так:

```
try {
   // Блок коду, де відслідковуються помилки
}
catch (тип_виняток_1 exceptionObject) {
   // Обробляємо помилку
}
catch (тип_виняток_2 exceptionObject) {
   // Обробляємо помилку
}
finally {
   // Код, який потрібно виконати після завершення
//блоку try
}
```

Відкомпілюємо і запустимо таку програму:

```
class Main {
    public static void main (String [] args) {
        int a = 4;
        System.out.println (a / 0);
    }
}
```

У момент запуску на консоль буде виведено наступне повідомлення:

Exception in thread "main" java.lang.ArithmeticException: / by zero at Main.main (Main.java:4)

З повідомлення видно, клас винятку, що відбувся - ArithmeticException. Цей виняток можна обробити:

```
class Main {
    public static void main (String [] args) {
        int a = 4;
        try {
            System.out.println (a / 0);
        } catch (ArithmeticException e) {
            System.out.println ( "Відбулася неприпустима арифметична операція");
```

```
        }
    }
}
```

Тепер замість стандартного повідомлення про помилку буде виконуватися блок catch, параметром якого є об'єкт *e* відповідного класу винятку (самому об'єкту можна давати будь-яке ім'я, воно буде потрібно в тому випадку, якщо ми побажаємо знову примусово викинути цей виняток, наприклад, для того, щоб воно було перевірено якимось ще оброблювачем).

У блок try при цьому розміщується той фрагмент програми, де потенційно може виникнути виняток.

Одному try може відповідати відразу кілька блоків catch з різними класами винятків.

```
import java.util.Scanner;
class Main {
   public static void main (String [] args) {
      int [] m = {-1,0,1};
      int a = 1;
      Scanner sc = new Scanner (System.in);
      try {
         System.out.println ( "a =");
         a = sc.nextInt ();
         m [a-1] = 4 / a;
         System.out.println (m [a]);
      } catch (ArithmeticException e) {
         System.out.println ( "Відбулася неприпустима арифметична операція");
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println ( "Звернення по неприпустимого індексу масиву");
      } catch (Exception e) {
         System.out.println ( "Сталося ще якесь виключення");
      }
   }}
```

Клас Exception у прикладі буде перехоплювати будь-які винятки, оскільки цей клас батьківський для всіх інших

контрольованих винятків (в тому числі, і InputMismatchException).

Оскільки винятки побудовані на ієрархії класів і підкласів, то спочатку треба намагатися обробити більше приватні винятки і лише потім більш загальні. Тобто, якщо поставити першим (а не третім) блок з обробкою винятку класу Exception, ми б ніколи не побачили ніяких повідомлень про помилку, окрім «Сталося ще якесь виключення» (всі винятки перехоплюються відразу цим блоком і не доходять до інших) .

Необов'язковим додаванням до блокам try ... catch може бути блок finally. Ключове слово finally створює блок коду, який буде виконуватися після завершення блоку try / catch, але перед кодом, наступним за ним. Блок буде виконаний, незалежно від того, передано виняток чи ні. Оператор finally не обов'язковий, проте кожен оператор try вимагає наявності або catch, або finally.

Наприклад, при вході метод відкриває файл і закриває при виході. Щоб не пропустити закриття файлу через обробку виключення, був запропонований механізм finally.

Частину винятків може обробляти сама система. Але можна створити власні винятки за допомогою оператора throw. Код виглядає так:

throw экземпляр_Throwable

Потрібно створити екземпляр класу Throwable або його спадкоємців. Отримати об'єкт класу Throwable можна в операторі catch або стандартним способом через оператор new:

```
double c = Math.sqrt (a-b);
        if (Double.isNaN(c)){
            throw new ArithmeticException ("корінь з від'ємного числа ");
            }
```

У прикладі оголошено змінну с, якій привласнюється результат обчислення квадратного кореня, який може біти від’ємним числом. В цьому випадку система викличе виняток, а ми можемо прочитати повідомлення про помилку “корінь з від'ємного числа ” (можна використовувати інший виняток, наприклад, throw new Exception ("корінь з від'ємного числа ") ;).

У будь-якому випадку ми передали обробку помилки системі. У реальному додатку потрібно обробити помилку самостійно.

Потік виконання зупиняється безпосередньо після оператора throw і інші оператори не виконуються. При цьому здійснюється пошук найближчого блоку try / catch відповідного винятку типу.

Перепишемо приклад з обробкою помилки.

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
           System.out.println("a=");
           int a = sc.nextInt();
           System.out.println("b=");
           int b = sc.nextInt();
           double c = Math.sqrt (a-b);
           if (Double.isNaN(c)){
              throw new ArithmeticException("корінь з від'ємного числа ");
               }
           System.out.println(c);
        } catch (ArithmeticException e) {
           System.out.println(e);
        } catch (Exception e) {
           System.out.println("стався ще який-то виняток");
        }
    }
```

Якщо метод здатний порушувати винятки, які він сам не обробляє, то він повинен оголосити про таку поведінку, щоб методи, які його викликають, могли захистити себе від цих винятків. Для визначення списку таких винятків, які можуть порушуватися методом, використовується ключове слово throws. Синтаксис визначення методу повинен бути розширений:

тип ім'я_методу (список аргументів) throws список_винятків {}

Нижче наведено приклад програми, в якій метод procedure може порушувати виняток типу IllegalAccessException і в методі main надається код для обробки цього типу винятків:

```
class ThrowsDemo {
    static void procedure() throws IllegalAccessException {
    System.out.println(" inside procedure");
            throw new IllegalAccessException("demo");
            }
    public static void main(String args[]) {
            try {
                    procedure();
                    }
            catch (IllegalAccessException e) {
                    System.out.println("caught " + e);
                    }
            }
}
```

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити програму з обробки виняткових ситуацій (відповідно варіанту). Забезпечити коректну роботу програми при любих введених даних за допомогою механізму винятків.
3. Підготувати звіт, в якому повинні бути код програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Обчисліть вираз: b*b*(a-b)/a
2. Обчисліть вираз: $\sqrt{a - b}$
3. Обчисліть вираз: $a\sqrt{b}$
4. Обчисліть вираз: $a\dfrac{b - a}{b + a}$
5. Обчисліть вираз: $\sqrt{a} + \sqrt{b} + \sqrt{a + b}$
6. Обчисліть вираз: $\sqrt{a - b}$ /c

7. Розробити метод swapXY(), що може створювати виняткову ситуацію, коли до функції передається параметр null. При необхідності викликається виняток NullPointerException. Розробіть код обробки цієї виняткової ситуації.
8.Обчислити факторіал числа n, що вводиться з консолі. Обробити всі можливі помилки користувача при введенні цього числа.
9. Реалізуйте клас, призначений для введення цілих чисел з консолі, передбачивши обробку помилок не коректного введення за допомогою винятків.
10. Обчисліть значення функції y = f(x) для $x \in [a, b]$, y=ln(x-1). Обробити всі можливі помилки.

Контрольні питання

1. Чи є помилку у наступному прикладі:

```
try {
} finally {
}
```

2. Які типи винятків будуть оброблятися наступною конструкцією?

```
catch (Exception e) {
}
```

3. Яка помилка у наступному фрагменті коду?

```
try {
} catch (Exception e) {
} catch (ArithmeticException a) {
}
```

ЛАБОРАТОРНА РОБОТА №6

Тема: Операції з файлами. Файлові потоки введення-виведення в Java

Мета: набуття навичок
опрацювання основних класів і методів роботи з файлами.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Частина обчислювальної платформи, яка відповідає за об мін даними, називається
системою введення / виведення. У Java вона представлена пакет ом java.io (input / output). Ключовим моментом в роботі з цим пакетом є поняття потоку (Stream) введення-виведення.

Для роботи зі стандартними потоками введення-виведення в класі System є три статичних об'єкта: System.in, System.out і System.err. Потік System.in пов'язаний з клавіатурою, потік System.out і System.err – з консоллю програми Java.

При роботі з файлами також застосовується концепція потоку. Для введення- виведення інформації в файл (з файлу) створюється потік введення або виведення і підключається до відповідного файлу.

При цьому використовують класи файлових потоків FileInputStream (читання даних з файлу) і FileOutputStream (запис даних у файл). Конструктор класів FileInputStream і FileOutputStream створює виняток FileNotFoundException (файл не знайдено). Ім'я (повне) файлу, з яким зв'язується потік введення або виведення, у вигляді текстового рядка передається як аргумент конструктору.

Приклад 1.

```
package javaapplication25;

// Підключення пакету:
import java.io.*;

public class JavaApplication25 {

   public static void main(String[] args) throws IOException{
      // Цілочисельне поле:
        int a;
        try{
        // Потік файлового виведення (запис даних):
        FileOutputStream fout=new FileOutputStream
("C:/Users/Svetlana/Documents/mydata.txt");
        // Потік файлового введення (читання з файлу):
```

```
            FileInputStream fin=new FileInputStream
        ("C:|/Users/Svetlana/Documents/base.txt");
            // Читання 1 символу з файлу base.txt :
            a=fin.read();
            while(a!=-1){
            // Заміна пробілів символами «_»:
            if(a==(int)' ') a=(int)'_';
            // Запис 1 символу у  файл mydata.txt:
        fout.write(a);
        // Зчитування наступного символу із файлу:
        a=fin.read();}
        // Закриття потоку виведення обов'язково:
        fout.close();
        // Закриття потоку введення обов'язково:
        fin.close();
        }catch(FileNotFoundException  e){  // Обробка винятку
"файл не знайдено":
        System.out.println("Немає доступу до файлу: "+e);}
        // Повідомлення програми:
        System.out.println("Роботу програми завершено!");
           }
           }
```

Програма посимвольно зчитує інформацію з вихідного текстового файлу base.txt і записується в файл mydata.txt. При цьому всі пробіли замінюються символами підкреслення. Оскільки використовується при цьому метод read (), який зчитує символ і повертає його числовий код, вибір типу int для поля a є цілком виправданий.

На консоль по завершенню роботи програми виводиться повідомлення "Робота програми завершена!".

Крім байтових потоків введення- виведення (FileOutputStream, FileInputStream) є символьні потоки. На відміну від першого варіанту, де одиницею обміну є байт, тут буде символ. Ці потоки реалізуються класами FileWriter та FileReader. В цих класах присутні методи writer() та reader() відповідно.

Наприклад:

```
FileReader f = new FileReader("myfile.txt");
```

```
char[] buffer = new char[512];
f.read(buffer);
f.close();
```

Ще один клас призначений для роботи з файлами – File. Цей клас, визначений у пакеті java.io, не працює безпосередньо з потоками. Його завданням є управління інформацією про файли та каталоги:

```
// Створюємо об'єкт File для каталогу
File dir1 = new File ("C: // SomeDir");
// Створюємо об'єкти для файлів, які знаходяться
//в каталозі
File file1 = new File ("C: // SomeDir", "Hello.txt");
File file2 = new File (dir1, "Hello2.txt");
```

Приклад 2 .

```
package javaapplication25;
// Підключення пакета:
import java.io.File;
import java.io.IOException;

public class JavaApplication25 {

   public static void main (String [] args) throws
      IOException {
 // Визначаємо об'єкт для каталогу
      File myFile = new File ( "C: \\ Users \\ Svetlana \\
      Documents \\ somepicture.png");
      System.out.println ( "Файл:" + myFile.getName ());
      System.out.println ( "Батьківський каталог:" +
      myFile.getParent ());
      if (myFile.exists ())
         System.out.println ( "Файл існує");
      else
         System.out.println ( "Файл ще не створено");
         System.out.println ( "Розмір файлу:" +
      myFile.length ());
      if (myFile.canRead ())
```

```
    System.out.println ( "Файл доступний для
читання");
else
    System.out.println ( "Файл не доступний для
читання");

if (myFile.canWrite ())
    System.out.println ( "Файл доступний для
запису");
else
    System.out.println ( "Файл не доступний для
запису");

// Створимо новий файл
File newFile = new File ( "C: \\ Users \\ Svetlana \\
Documents \\ MyFile.txt");
try
{
    boolean created = newFile.createNewFile ();
    if (created)
        System.out.println ( "Файл було створено");
}
catch (IOException ex) {
        System.out.println (ex.getMessage ());
}   }   }
```

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Переробити приклад 1, щоб код використовував не байтові, а символьні потоки.
3. Розробити додаток відповідно до варіанту. При роботі з файлами здійснювати перевірку їх існування.
4. Підготувати звіт, в якому повинні бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Реалізувати програму, що записує в файл послідовність випадкових чисел у діапазоні, що вказаний за допомогою параметрів командного рядка. Ім’я файлу вказується також в командному рядку. У програмі передбачити перевірку існування файлу з заданим ім'ям, в разі його відсутності попередньо перед записом в файл створити його. Доповнити додаток контролем перевірки чисел на дублювання при записі їх у файл. Реалізувати виведення на консоль вмісту файлу.
2. Дано два файли дійсних чисел з іменами afile і bfile, які містять елементи прямокутних матриць M1 і M2 (за строками), причому початковий елемент кожного файлу містить кількість стовпців відповідної матриці. Створити файл з ім'ям cfile, що містить суму M1 + M2.
3. Дан файл дійсних чисел, що містить елементи квадратної матриці (за строками), причому початковий елемент файлу містить значення кількості стовпців матриці. Створити новий файл, що містить матрицю, транспоновану до вихідної.
4. Дан файл дійсних чисел, що містить елементи квадратної матриці (за строками), причому початковий елемент файлу містить значення кількості стовпців матриці. Створити новий файл, що містить k-ий стовпець вихідної матриці.
5. Дан файл F1, що містить рядки. Скопіювати в файл F2 тільки ті рядки з F1, які починаються з літери «А». Підрахувати кількість слів у F2.
6. Створити декілька об’єктів класу Книга з заданими полями (автор, назва, рік видання тощо). Записати значення полів у файл, потів прочитати їх та виконати фільтрацію за наступних умов: вивести список книг одного автора, вивести список книг, виданих у 2016 році.
7. Дан файл F1, що містить рядки. Скопіювати з файлу F1 в файл F2 рядки, починаючи з 4. Підрахувати кількість символів в останньому слові F2.
8. Дан файл дійсних чисел, що містить елементи квадратної матриці (за строками), причому початковий елемент файлу містить значення кількості стовпців матриці. Створити новий файл тієї ж структури, що містить k-й рядок вихідної матриці.

9. Дан файл F1, що містить рядки. Скопіювати в файл F2 тільки парні рядки з F1. Підрахувати розмір файлів F1 і F2 (в байтах).

10. Компоненти файлу A- цілі числа, значення яких можуть повторюватися. Отримати файл B, утворений з A, в якому розміщені числа без повторень.

Контрольні питання

1. Який пакет треба імпортувати для роботи з класом File?
2. За допомогою якого конструктору не можна створювати екземпляр класу Fie:
   File(File parent, String child)
   File(String pathname)
   File(String parent, String child)
   File(URI uri)
   File(URL url)
3. Назвіть основні методи класу File.

МОДУЛЬ 2. СТВОРЕННЯ ГРАФІЧНИХ ДОДАТКІВ ТА ОБРОБКА ДАНИХ В JAVA

ЛАБОРАТОРНА РОБОТА №7

Тема: Розробка графічного інтерфейсу в Java

Мета: формування навиків розробки графічних додатків із застосуванням бібліотек AWT і Swing.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Робота з вікнами і графікою в Java здійснюється в апплетах і графічних додатках. Аплети - це невеликі програми, що вбудовуються в Web-документ і використовують для своєї візуалізації засоби Web-браузера. Графічні додатки самі відповідають за своє промальовування.

Графічні інструменти в мові Java реалізовані за допомогою двох бібліотек:

1. Пакет AWT (завантажується java.awt) містить набір класів, що дозволяють виконувати графічні операції і створювати елементи управління.

2. Пакет Swing (завантажується javax.swing, ім'я javax позначає, що пакет не є основним, а лише розширенням мови) містить нові класи, здебільшого аналогічні AWT. До імен цих класів додається J (JButton, JLabel і т. Д.). Пакет є частиною бібліотеки JFC (Java Foundation Class).

Для ефективної роботи з візуальними компонентами необхідна установка наступних трьох архітектурних складових.

1. Схеми (layout). Swing містить безліч схем, які представляють собою класи, що керують розміщенням компонентів в додатку і тим, що повинно статися з ними при зміні розмірів вікна програми або при видаленні, додаванні компонентів.

2. Події (event). Програма повинна реагувати на натискання клавіш, натискання кнопки миші і на все інше.

3. Моделі (model). Для більш просунутих компонентів (списки, таблиці, дерева) і навіть для деяких більш простих, наприклад, JComboBox, моделі - це самий ефективний спосіб роботи з даними (згадайте MVC).

Модель делегування подій в Java

У моделі подій будь-який графічний компонент може породжувати (initiate) подію. У цій моделі кожна подія - це клас, успадкований від класу java.util.EventObject. Все AWT події успадковуються від класу java.awt.AWTEvent.

Для зручності різні типи подій AWT поміщені в окремий пакет java.awt.event. Коли подію запущено, вона приходить до одного або декількох слухачів (listener), які її обробляють. Слухач кожної події - це об'єкт класу, що реалізує певний вид інтерфейсу. Для обробки події необхідно створити об'єкт-слухач і зареєструвати його в компоненті, який запускає певну подію. Реєстрація проводиться за допомогою відповідного методу addXXXListener (), що знаходиться в компоненті, який запускає подію, де XXX позначає тип події, яка буде оброблятися слухачем.

Приклад 1 найпростішої програми з використанням графічного пакету swing

```
package helloworld;
import javax.swing.*;
public final class Helloworld {
public static void main(String[] args) {

// Створимо вікно із заголовком "Hello, World!"
        JFrame f = new JFrame ("Hello, World!");
        f.setDefaultCloseOperation (JFrame.
        DISPOSE_ON_CLOSE );

        f.add(new JLabel("Hello World"));
        f.pack();// Показати вікно
        f.setVisible(true);
    }
}
```

Приклад 2. Безпосередньо у вікні форми елементи управління не розміщуються. Для цього служить панель вмісту JPanel, що займає весь простір вікна. Це щось на зразок контейнера для компонентів, який займає прямокутну область екрану і показує компоненти, вирівняні за простим принципом. Звернутися до цієї панелі можна методом getContentPane () класу JFrame. Як саме вирівняні компоненти, залежить від типу схеми розміщення, яка встановлена для панелі:

BorderLayout - простір контейнера розбивається на п'ять частин, компоненти розташовані по краях і один великий компонент розміщено в середині;

FlowLayout – встановлює компоненти в рядок по горизонталі;

GridLayout - встановлює компоненти в довільній таблиці n * m.

Ключова ідея в тому, що "компонентом" може бути не тільки кнопка або прапорець, а й інша JPanel. Можна отримати досить складний для користувача інтерфейс, просто розташовуя панелі одна на інший і обирая для них планування.

Контейнер найвищого рівня, який з'являється на екрані, є екземпляром JFrame, а не JPanel. Щоб додати вашу основну панель в екземпляр JFrame треба викликати myJFrame.getContentPane (). add (myJPanel, BorderLayout.Center).

Якщо у вас є екземпляр об'єкта JPanel, викличте метод .setLayout, щоб встановити тип планування, і потім метод .add, щоб додати на панель компоненти.

Щоб змусити додаток робити щось більше, ніж просто виклик вікна, що з'являється на екрані, потрібно застосовувати інтерфейс ActionListener. У будь-якого неабстрактного ActionListenerє тільки один метод actionPerformed, який викликається, коли користувач виконує "дію" над компонентом, в якому зареєстрований слухач (наприклад, дію над кнопкою - її натискання). Щоб зареєструвати слухача дій для кнопки або будь-якого іншого компонента, викличте метод.addActionListener ().

Приклад коду із застосуванням IPanel:

```
package wikihow;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class WikiHow extends JFrame implements ActionListener {

    /**  * Кнопка.  */
    JButton myButton = new JButton("Button");
    /**  * Прапорець  */
    JCheckBox myCheckBox = new JCheckBox("Check");
    /**  * Текстове поле.  */
    JTextArea myText = new JTextArea("My text");
    /** Нижня панель, яка містить кнопку.  */
    JPanel bottomPanel = new JPanel();
```

```
        /**     * Батьківська панель, яка містить всі
компоненти     */
        JPanel holdAll = new JPanel();

        /**     * Конструктор.     */
        public WikiHow() {
            bottomPanel.setLayout(new FlowLayout());
            bottomPanel.add(myCheckBox);
            bottomPanel.add(myButton);
            holdAll.setLayout(new BorderLayout());
            holdAll.add(bottomPanel,
BorderLayout.SOUTH);
            holdAll.add(myText, BorderLayout.CENTER);
            getContentPane().add(holdAll,
BorderLayout.CENTER);
            myButton.addActionListener(this);
            myCheckBox.addActionListener(this);

setDefaultCloseOperation(DISPOSE_ON_CLOSE);    }
        public static void main(String[] args) {
            WikiHow myApplication = new WikiHow();
      // Вказуємо, де повинно з'явитися  вікно:
            myApplication.setLocation(10, 10);
            myApplication.setSize(300, 300);
      // Показати  вікно!
            myApplication.setVisible(true);
          }

        /* Любий неабстрактний клас, який реалізує
ActionListener повинен мати  цей метод.
         ** @param e Подія.
         */
        public void actionPerformed(ActionEvent e) {
          if (e.getSource() == myButton) {
            myText.setText("A button click");
          } else if (e.getSource() == myCheckBox) {
            myText.setText("The checkbox state changed
to "   + myCheckBox.isSelected());
```

```
                } else {
                    myText.setText("E ...?");
                }
            }
        }
```

Створення головного меню за допомогою JMenuBar

В арсеналі компонентів графічного інтерфейсу користувача бібліотеки Swing є такий компонент, як головне меню JMenuBar. МенюJMenuBar, як правило розташовується у верхній частині вікна програми у вигляді горизонтальної смужки. Меню може мати довільну вкладеність і складися з інших пунктів меню. При натисканні на пункті меню можуть відбуватися якісь дії, передбачені розробником.

При організації меню на Java Swing використовується класи JMenuBar, JMenu і JMenuItem. Всі ці класи розташовуються в пакеті javax.swing. * У найпростішому випадку ми створюємо екземпляр JMenuBar, додаємо до нього необхідний набір JMenu і JMenuItem (іноді, тільки JMenu), а потім за допомогою методу setJMenuBar говоримо JFrame використовувати наше нове меню в якості головного меню. Параметр у метода setJMenuBar тільки один - посилання на JMenuBar.

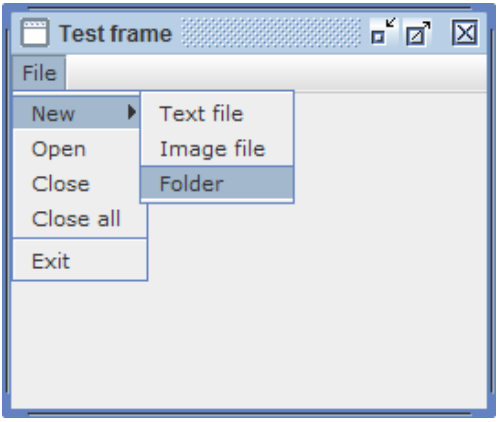

Рис. 9. Приклад системи меню

Для організації ієрархії вкладеності меню використовуються JMenu і JMenuItem. JMenu - це меню, яке містить в собі JMenuItem або вкладені меню JMenu. JMenuItem - це пункт меню, який представляє собою вже конкретну команду меню. Принципи створення меню подивимося на прикладі меню New.

```
JMenu newMenu = new  JMenu("New");

JMenuItem txtFileItem = new JMenuItem("Text file");
newMenu.add(txtFileItem);

JMenuItem imgFileItem = new JMenuItem("Image file");
newMenu.add(imgFileItem);

JMenuItem folderItem = new JMenuItem("Folder");
newMenu.add(folderItem);
...
```

Приклад 3. Повний код додатку, що демонструє приклад роботи з меню наведено нижче:

```
package wikihow;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class WikiHow implements ActionListener {
JLabel  jlab;

public WikiHow() {

// створюється новий контейнер типу JFrame
JFrame  jfrm = new JFrame("Menu Demo");

// встановлюється схема компановки  FlowLayout
jfrm.setLayout(new FlowLayout () );
```

```
// задаються вихідні розміри фрейму
jfrm.setSize (220, 200);

// завершити прикладну програму, коли користувач
// закриє вікно
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//створити мітку для відображення результатів вибору
//меню
jlab = new JLabel();

// створити рядок меню
JMenuBar jmb = new JMenuBar();

// створити меню File
JMenu jmFile = new JMenu("File"); // Файл
// Відкрити
JMenuItem jmiOpen = new JMenuItem("Open");
// Закрити
JMenuItem jmiClose = new JMenuItem("Close");
JMenuItem  jmiSave = new JMenuItem("Save"); // Зберегти
JMenuItem  jmiExit = new JMenuItem("Exit"); // Вихід
jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);

// створити меню Options
JMenu jmOptions = new JMenu("Options"); // Параметри

// створити підменю Colors
JMenu jmColors = new JMenu("Colors"); // колір
JMenuItem jmiRed = new JMenuItem("Red"); // Червоний
// Зелений
JMenuItem jmiGreen = new JMenuItem("Green");
JMenuItem jmiBlue = new JMenuItem("Blue"); // Синій
```

```
jmColors.add(jmiRed);
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
jmOptions.add(jmColors);

// створити підменю
JMenu jmPriority = new JMenu("Priority");// Пріоритет
JMenuItem jmiHigh = new JMenuItem("High"); // Високий
JMenuItem jmiLow = new JMenuItem("Low"); // Низький
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
jmOptions.add(jmPriority);

// створити пункт меню Reset
JMenuItem jmiReset = new JMenuItem("Reset"); // Скинути
jmOptions.addSeparator();
jmOptions.add(jmiReset);

// Ввести всі обрані меню в  рядок меню
jmb.add(jmOptions);

// створити меню Help
JMenu jmHelp = new JMenu("Help"); // Довідка
// Про   програму
JMenuItem jmiAbout = new JMenuItem("About");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);

//ввести  приймачі дій від пунктів меню
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this) ;
```

```
        jmiReset.addActionListener(this);
        jmiAbout.addActionListener (this) ;

        // вивести мітку на панель
        jfrm.add(jlab);
        // додати рядок меню до фрейму
        jfrm.setJMenuBar(jmb) ;
        // відобразити фрейм
        jfrm.setVisible(true);

        }



           // обробити події  пунктів меню
        public void actionPerformed(ActionEvent e) {

        // отримати команду дій від обраного  меню
           String comStr = e.getActionCommand();

        // вийти з програми, якщо обрано пункт меню Exit
           if(comStr.equals("Exit")) System.exit(0);

        // відобразити результат вибору з меню
           jlab.setText(comStr + " Selected"); // Обрано вказане
        }

           public static void main(String[] args)
           {
              // створити фрейм у потоці диспетчеризації дій-
              // це найбільш рекомендований варіант коду, що
дозволяє уникнути
              // виникнення помилок при роботі з фреймом
              SwingUtilities.invokeLater(new Runnable(){
              public void run() {
                new WikiHow();
              }
              });
```

```
    }

}
```

Часто пункти меню об'єднуються у групи. Одна група від іншої відділяється горизонтальною лінією. На рис. 9 риска проведена зверху команди Exit. Ця риска створюється методом addSeparator () класу JMenu:

```
fileMenu.addSeparator();
```

Шрифти. Для встановлення шрифту необхідно створити об'єкт класу Font з параметрами: назва гарнітури шрифту (тип String), стиль шрифту (тип int) і розмір шрифту в пунктах (тип int):

```
Font f = new Font ("Times Roman", Font.BOLD, 72);
```

Деякі методи класу Font:

getFamily - отримує назву шрифту, залежне від платформи;

getName - отримує логічне ім'я шрифту;

getStyle - отримує стиль шрифту;

getSize - отримує розмір шрифту в пунктах;

isPlain - повертає true, якщо шрифт простий;

isBold - повертає true, якщо шрифт напівжирний;

isItalic - повертає true, якщо шрифт курсивний.

getFont - отримує шрифт зі списку параметрів системи

Якщо вибрати шрифт, який не встановлений на машині, Java замінить його стандартним шрифтом (наприклад, Courier). Для того, щоб дізнатися які шрифти доступні, можна скористатися методом getAllFonts (), що визначений в класі GraphicsEnvironment.

Приклад 4. Створимо аплет FontsList, у вікні якого відображається список усіх доступних шрифтів:

```
package javaapplication30;

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class FontsList extends JApplet {
```

```
    Font f[];
    int count = 0, x = 10, y = 20;
    double w, h;
    int tek = 0;

    public FontsList() {
    }

    @Override
    public void start() {
        Dimension d = getSize();
        w = d.getWidth();
        h = d.getHeight();
    }

    @Override
    public void init() {
        GraphicsEnvironment ge          =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        f = ge.getAllFonts();
        MouseListener ml = new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                repaint();
            }
        };
        addMouseListener(ml);
        resize(800, 300);
    }

    @Override
    public void paint(Graphics g) {
        g.clearRect(0, 0, (int) w, (int) h);
        int i;
        for (i = tek; i < f.length; i++) {
            String s = "";
            s += f[i].getFontName();
            s += " ";
```

```
                s += f[i].getFamily();
                s += " ";
                s += f[i].getSize();
                s += " ";
                s += f[i].toString();
                g.drawString(s, x, y);
                y = y + 20;
                if (y >= h) {
                   y = 20;
                   break;
                }   // перехід до початку
             }
             if (i == f.length) {
                i = 0;
                y = 20;
             }   // на  початок списку
             tek = i;
          }

          public static void main(String[] args) {
             JFrame frame = new JFrame("Приклад");
             FontsList appl = new FontsList();
             appl.init();
             appl.start();
             frame.getContentPane().add(appl);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setSize(800, 300);
             frame.setVisible(true);
          }
       }
```

Якщо ми хочемо змінити колір шрифту і його вид, то можна додати код в метод paint (у цикл for):

```
g.setFont(new Font(f[i].getFontName(),Font.PLAIN,14));
g.setColor(new Color((float)Math.random(),
(float)Math.random(),  (float)Math.random()));
```

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити графічний додаток (відповідно варіанту) із застосуванням графічних бібліотек java.
3. Підготувати звіт, в якому повинні бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Введення тексту в графічне вікно програми. У вікні встановлено рядок меню (JMenuBar), в якій встановлено два меню (JMenu) - "Шрифт" і "Стиль". В меню "Шрифт" встановлено три пункти меню (JRadioButtonMenuItem): "Times New Roman" (шрифт за замовчуванням), "Arial" і "Verdana". В меню "Стиль" визначено чотири пункти меню (JRadioButtonMenuItem): "Простий" (шрифт за замовчуванням), "Жирний", "Курсив" і "Жирний курсив". У текстовій панелі (JTextPane) "Введення тексту" вікна програми вводиться текст з клавіатури. При виборі одного з пунктів меню текст в панелі виводиться відповідним шрифтом і / або стилем.

2. Виведення рядка заданим шрифтом і заданого кольору в графічному вікні. У верхній панелі (JPanel) "Управління виведенням" задається напис (JLabel) "Текст:" і текстове поле (JTextField), напис (JLabel) "Гарнітура:", список, що обертається (JSpinner), зі значеннями "Times New Roman" (шрифт по замовчуванням), "Arial" і "Verdana", напис (JLabel) "колір:" і список (JSpinner) зі значеннями "Чорний" (колір за замовчуванням), "Червоний", "Зелений" і "Синій", а також кнопка (JButton) "Вивести рядок". У нижній панелі (JPanel) "Виведення рядка" виводиться в графічному контексті (за допомогою методу drawString ()) в довільному місці порожній рядок. При завданні тексту рядка в текстовому полі, параметрів рядка в верхній панелі і натисканні кнопки "Виведення рядка" рядок заданого вмісту і кольору виводиться заданим шрифтом в нижній панелі.

3. Написати програму, при запуску якої в головному вікні з'являється меню, яке містить два пункти – «Пуск» та

«Вихід», кнопка «Пуск», написи (JLabel) «Результат:» та число 10. При кожному натисканні кнопки «Пуск» це число зменшується на 1, до тих пір, поки воно не стане дорівнює 1. При подальшому натисканні на кнопку виводиться діалогове вікно повідомлень, яке повідомляє про кінець ітерацій натискання кнопки. Дію кнопки дублює пункт меню «Пуск» .

4. Розробіть елементарний калькулятор, який передбачає поля введення чисел та відображення результату математичних операцій (арифметичні діє, квадратний корінь, ступінь 2), кнопки виконання певних математичних операцій. Додаток повинен мати меню, в якому містяться пункти, що дублюють кнопки математичних операцій. При виникненні виняткових ситуацій повинно виводитися діалогове вікно з відповідним повідомленням.

5. Введення тексту в графічне вікно програми. У верхній панелі вікна (JPanel) "Параметри шрифту" запропоновано такі компоненти: напис (JLabel) "Стиль:", список, що розкривається (JComboBox) з наступними пунктами: "Простий" (стиль за замовчуванням), "Жирний" і "Курсив", напис (JLabel) "розмір:" і список, що розкривається, (JComboBox) з пунктами: "10pt" (стиль за замовчуванням), "12pt" (розмір за замовчуванням) і "14pt". У текстовій панелі (JTextPane) "Введення тексту" вікна програми вводиться текст. При виборі одного з пунктів списків, що розкривається - стилю і розміру тексту, вміст текстової панелі виводиться відповідним форматом. При введенні значення у текстове поле перевіряється, чи є в цьому тексті числа, якщо є – передбачити відображення цієї інформації на формі (самостійно визначити організацію відображення цієї інформації – вибір елементів тощо).

Контрольні питання

1. Які існують способи створення діалогових вікон?
2. Для яких цілей використовуються менеджери компонування в мові Java і як вони реалізуються?
3. Як в Swing реалізована панель інструментів (tool bar)?

ЛАБОРАТОРНА РОБОТА №8

Тема: Робота із зображеннями в Java
Мета: формування навиків розробки графічних додатків із застосуванням бібліотек AWT і Swing.
Час виконання роботи: 4 години.
Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Графічні операції завжди виконуються над об'єктом Graphics (контекст відображення). Деякі методи класу Graphics:

clearRect - очищає вказаний прямокутник, заповнює кольором фон;

clipRect - задає область обмеження виведення;

copyArea - копіює область екрану;

create - створює новий об'єкт, який є копією вихідного об'єкта;

draw3DRect - малює прямокутник з об'ємним ефектом;

drawArc - малює дугу поточним кольором;

drawBytes - малює зазначені байти поточним шрифтом і кольором;

drawChars - малює зазначені символи поточним шрифтом і кольором;

drawImage - малює вказане зображення типу Image;

drawLine - малює лінію між точками;

drawOval - малює овал всередині зазначеного прямокутника поточним кольором;

drawPolygon - малює багатокутник поточним кольором;

drawRect - малює контур прямокутника поточним кольором;

drawRoundRect - малює контур прямокутника із закругленими краями;

drawString - малює зазначений рядок поточним шрифтом і поточним кольором;

fill3DRect - розфарбовує кольором прямокутник з об'ємним ефектом;

fillArc - заповнює дугу поточним кольором;

fillOval - заповнює овал поточним кольором;

fillPolygon - заповнює багатокутник поточним кольором;

fillPolygon - заповнює об'єкт класу Polygon поточним кольором;

fillRect - заповнює прямокутник поточним кольором;

fillRoundRect - заповнює прямокутник із закругленими краями;

setPaintMode - встановлює режим заповнення поточним кольором.

Колір. Для завдання кольору використовується метод setColor () класу Graphics. Наступний приклад створює випадковий колір і встановлює його g - об'єкт Graphics:

g.setColor (new Color ((float) Math.random (), (float) Math.random (), (Float) Math.random ()));

Колірна модель мови Java є 24-розрядною моделлю RGB (червоний, синій, зелений), тобто об'єкти класу Color можуть містити 24 розрядів колірної інформації (що відповідає 16 мільйонам відтінків).

Таймер. Для реалізації таймера в Java є клас Timer в пакеті java.util. Основні методи таймера: schedule і cancel, які запускають і зупиняють виконання таймера.

Крім того, для таймера необхідно підготувати завдання – клас, що успадковує від класу TimerTask (клас Updater в прикладі). В цьому класі необхідно перевизначити метод public void run (), який викликається, коли таймер робить свій черговий відлік.

Приклад 1. У наведеному нижче прикладі програми демонструється застосування класів Timer і TimerTask. У цій програмі визначається завдання, що запускається за таймером, а при його виконанні метод run () виводить повідомлення "Завдання виконується за таймером". Це завдання планується для запуску кожні пів секунди після початкової паузи протягом однієї секунди.

```
package javaapplication40;

import java.util.Timer;
import java.util.TimerTask;

class MyTimerTask extends TimerTask {
```

```
        @Override
        public void run() {
            System.out.println("Завдання виконується за таймером.");
        }
    }

    public class JavaApplication40 {

        public static void main(String[] args) {

            MyTimerTask myTask = new MyTimerTask();
            Timer myTimer = new Timer();

          /* Встановлює початкову паузу протягом 1 секунди,
         а потім повторює завдання кожні пів секунди    */
            myTimer.schedule(myTask, 1000, 500);

            try {
                Thread.sleep(5000);
            } catch (InterruptedException exc) {
            }
            myTimer.cancel();
        }
    }
```

Графіка 2D. На базі Java реалізований пакет Java2D, який дозволяє створювати потужну графіку. Використання цього пакету можливе тільки в Java2-додатках, побудованих на основі Java Foundation Classes. Замість типу Graphics тут можливе використання типу Graphics2D і всіх методів, що надаються класом Graphics2D.

Приклад 2. Нижче наводиться приклад аплету, який малює окружність з градієнтною заливкою. В перевизначеному методі Paint() відбувається малювання еліпсу.

```
package drawfig;

import java.awt.Color;
```

```
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Ellipse2D;
import javax.swing.JApplet;
import javax.swing.JFrame;

public class DrawFig extends JApplet  {
  public void init() {   }
  @Override
  public void paint(Graphics g)   {
     Graphics2D g2d = (Graphics2D)g;
     GradientPaint gradient = new
 GradientPaint(0,0,Color.red,175,175,Color.yellow,true);
     g2d.setPaint(gradient);
// координати елінсу
     Ellipse2D.Double circle =  new
Ellipse2D.Double(10,10,350,350);
     g2d.fill(circle);//заповнити коліром
     g2d.setPaint(Color.red);//колір окружності
     g2d.draw(circle);
  }
  public static void main(String[] args) {
     JFrame frame = new JFrame ("Приклад");
     DrawFig appl = new   DrawFig();
     appl.init();
     appl.start();
     frame.getContentPane().add(appl);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setSize(400, 400);
     frame.setVisible(true);
  }
}
```

Проста анімація у Java. Анімація може бути створена з використанням об'єктів, що належать класу javax.swing.Timer. Об'єкт Timer може генерувати послідовність подій без втручання користувача. При роботі з анімацією необхідно створити об'єкт

Timer і обробляти кожне повідомлення таймера, відображаючи один кадр анімації за іншим. При цьому таймер працює в вигляді самостійного потоку (фоновий режим).

Події, які генеруються таймером, - це події типу ActionEvent. Конструктор таймера в якості параметрів отримує два значення - проміжок часу між двома подіями і прослуховувач подій ActionListener, який буде повідомлятися про настання події:

Timer (int delayTime, ActionListener listener);

Прослуховувач подій повинен відповідати на виникаючі події, для цього використовується метод actionPerformed ().

Таймер не розпочинає своє роботу автоматично відразу після того, як він буде створений. Щоб запустити таймер, необхідно скористатися методом start (). Щоб зупинити таймер використовується метод stop (). Цей метод викликається для того, щоб зупинити процес генерації подій таймера. Якщо таймер був зупинений, то для відновлення роботи таймера використовується метод restart (). Метод start () повинен бути викликаний тільки один раз. Всі перераховані методи не мають параметрів.

Приклад 3.У наступному прикладі запускається анімація при натисканні кнопк Start. Після натискання кнопки напис на ній змінюється, ми побачимо новий напис Stop. Натискання цієї кнопки в подальшому призведе до зупинки анімації. Цей аплет "говорить": "Hello, World!". Анімація використовується для поступової зміни кольору тексту, що виводиться.

```
package helloworldspectrum;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HelloWorldSpectrum extends JApplet {

public static void main(String[] args) {
    JFrame frame = new JFrame ("Приклад");
  HelloWorldSpectrum  appl = new   HelloWorldSpectrum ();
    appl.init();
    appl.start();
    frame.getContentPane().add(appl);
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(400, 400);
                frame.setVisible(true);
            }

    // JPanel вкладеного класу Display для
    // відображення "Hello, World!"
            Display display;
   // кнопка, що застосовується для запуску й
   //остановки  анімації
            JButton startStopButton;
            Timer timer; // таймер, що керує анімацією
    // Таймер запускається, коли користувач натискає кнопку.
    //Кожен раз при отриманні події таймера змінюється колір
    // виведеного тексту.
    // Значення змінної відповідає null, якщо анімація не працює.
            public int colorIndex;
            // Число, яке визначає колір, в проміжку від 0 до 100.
            // При отриманні події це число збільшується на 1
            @Override
            public void init() {
            // викликається системою для ініціалізації аплету
            display = new Display();
            // компонент, що відображує "Hello, World!"
            getContentPane().add(display, BorderLayout.CENTER);
            // вставляє панель відображення в центр панелі
            //аплету JApplet
            JPanel buttonBar = new JPanel();
            buttonBar.setBackground(Color.gray);
            getContentPane().add(buttonBar, BorderLayout.SOUTH);
            startStopButton = new JButton("Start");
            buttonBar.add(startStopButton);

            startStopButton.addActionListener(new ActionListener() {
            // Прослуховувач подій, який реагує на натискання
            //кнопки, яка запускає і зупиняє анімацію і перевіряє
            //значення таймера. Якщо таймер не null –
```

```
//анімація запущена, то є її /треба зупинити.
// Якщо таймер null, то анімацію слід запустити.
@Override
public void actionPerformed(ActionEvent evt) {
if (timer == null) {
startAnimation();
            } else {
stopAnimation();
            }
          }
       });
   }
   ActionListener timerListener = new ActionListener() {
// Описує прослуховувач подій, що відповідає на
//події таймера.
// При отриманні подій таймера індекс кольору
//змінюється на 1.
@Override
public void actionPerformed(ActionEvent evt) {
colorIndex++;
// число от 0 до 100
if (colorIndex > 100) {
colorIndex = 0;
          }
float hue = (colorIndex / 100.0f); // От 0.0F до 1.0F.
display.setColor(Color.getHSBColor(hue, 1, 1));
}
   };

void startAnimation() {
// Запускається анімація, якщо вона ще не запущена.
// Стан перевіряється за значенням таймера.
// Якщо значення null, то анімація не запущена,
if (timer == null) {
// створюється таймер, який буде
// створювати події кожні 50 мс та відправляя їх
//прослуховувачу подій
timer = new Timer(50, timerListener);
```

```
timer.start(); // Запускаємо таймер.
startStopButton.setText("Stop");
        }
    }

void stopAnimation() {
if (timer != null) {
timer.stop();
// зупиняємо таймер timer = null;
// змінної таймера присвоюємо null
startStopButton.setText("Start");
}
}

@Override
public void  stop() {
// Метод аплету stop () викликається системою перед
//тим, як аплет повинен припинити роботу тимчасово
//або назавжди. Працюючий таймер не потрібен тоді,
//коли аплет не працює, зупиняємо анімацію.
//Якщо анімація не працює, то її не треба зупиняти.
stopAnimation();
    }
// вкладений клас надає "полотно" для малювання
class Display extends JPanel {
// Вкладений клас представляє панель для відображення
// рядка "Hello, World!". Колір і шрифт
// запам'ятовуються в змінних colorNum і textFont.
Color color; // колір, яким виводиться текст
//(спочатку червоний)
Font textFont; // шрифт, яким виводиться текст
// об'єкт шрифту містить розмір і стиль

Display() {
// Конструктор класу Display. Здається колір фону і
// початкові значення для змінних кольору та шрифту.
setBackground(Color.black);
color = Color.red;
```

```
// початковий колір тексту
textFont = new Font("Serif", Font.BOLD, 36);
// створюється об'єкт шрифту
}

void setColor(Color color) {
// Метод викликається з основного класу для
//встановлення кольору  тексту і змінює колір тексту.
//Колір Color не повинен бути null
this.color = color;
repaint();
      }

@Override
public void paintComponent(Graphics g) {
// Викликається системою для промальовування
//панелі Jpanel.
//Викликається клас super.paintcomponent()
//для заповнення фону заданим кольором.
// Потім виводиться текст відповідно до поточного
//кольору,
super.paintComponent(g);
g.setColor(color);
g.setFont(textFont); // задається шрифт
g.drawString("Hello World!", 25, 50); // виводиться текст
      }
   }
}
```

Приклад 4. Ще один приклад анімації – малювання кольорової кульки та її переміщення по діагоналі. У цьому вікні два поля: робоча область зліва з білим фоном та зелена кулька, а праворуч кнопка, після натискання якої кулька починає рухатися, натиснути ще раз - зупиняється.

```
package javaapplication38;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
```

```
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Ellipse2D;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.Timer;

@SuppressWarnings("serial")
public class MovingCircle extends JComponent implements
ActionListener {

private double scale;
private Color color;
private Timer timer;
public double x = 10;
public double y = 10;

public MovingCircle(Color color, int delay) {
scale = 1.0;

timer = new Timer(delay, this);
        this.color = color;
setPreferredSize(new Dimension(500, 500));
    }

public void start() {
timer.start();
    }

public void stop() {
timer.stop();
    }
```

```
    @Override
  public void actionPerformed(ActionEvent arg0) {
  repaint();
    }

    @Override
  protected void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
  g2d.setColor(Color.white);
  int width = 500;
  int height = 500;
  g.fillRect(0, 0, width, height);
  g2d.setColor(Color.black);
  g2d.drawRect(0, 0, width - 1, height - 1);
  g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING
  , RenderingHints.VALUE_ANTIALIAS_ON);
  g2d.setColor(color);
  g2d.scale(scale, scale);
  x++;
  y++;
        Ellipse2D el = new Ellipse2D.Double(x, y, 20, 20);
g2d.fill(el);
   }

public static void main(String[] args) {
SwingUtilities.invokeLater(new Runnable() {
public void run() {
            JFrame frame = new JFrame("Moving Circle");
            JPanel panel = new JPanel();

final MovingCircle MovingCircleGreen = new
MovingCircle(Color.green, 20);
panel.add(MovingCircleGreen);
frame.getContentPane().add(panel);
final JButton button = new JButton("Start");
button.addActionListener(new ActionListener() {
private boolean pulsing = false;
```

```
                @Override
public void actionPerformed(ActionEvent e) {
if (pulsing) {
pulsing = false;
MovingCircleGreen.stop();
button.setText("Start");
                    } else {
pulsing = true;
MovingCircleGreen.start();
button.setText("Stop");
                    }
                }
            });
panel.add(button);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(600, 550);
frame.setVisible(true);
            }
        });
    }
}
```

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити графічний додаток (відповідно варіанту) із застосуванням графічних бібліотек java.
3. Підготувати звіт, в якому повинні бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Створити вікно, в якому розмістити еліпс, заповнений певним кольором. По натисканню кнопки Start еліпс змінює колір. Зупинка процесу відбувається при натискання кнопки Stop.

2. Створити вікно, в якому розмістити еліпс, який по натисканню кнопки Start змінює свій розмір. Зупинка процесу відбувається при досягненні певних розмірів.

3. Створити вікно, в якому розмістити прямокутник, який по натисканню кнопки Start змінює свою форму і стає квадратом.

4. Створити вікно, в якому розмістити текст HelloWorld, який при натисканні кнопки Start починає рухатися у горизонтальному напрямку. При натисканні кнопки Stop рядок зупиняється.

5. Створити вікно, в якому за допомогою графічних компонентів Java намалювати рисунок, в якому застосувати різні геометричні фігури, кольори, лінії.

Контрольні питання

1. Як створити об'єкт таймера, який би використовувався для однократного створення події?
2. Що таке аплет?
3. Які ви знаєте способи створення анімації в Java?

ЛАБОРАТОРНА РОБОТА №9

Тема: Обробка математичних функцій та побудова графіків функцій в Java

Мета: навчитися будувати графіки функцій за допомогою стандартних засобів, що надаються бібліотекою awt Java.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Класс Math

Клас Math містить методи, пов'язані з геометрією і тригонометрією та іншою математикою. Методи реалізовані як static, тому можна відразу викликати через Math.methodName () без створення екземпляра класу.

У класі визначено дві константи типу double: E і PI.

Наступні методи для тригонометричних функцій приймають параметр типу double, що виражає кут в радіанах.

sin (double d)
cos (double d)
tan (double d)
asin (double d)
acos (double d)
atan (double d)
atan2 (double y, double x)

Існують також гіперболічні функції: *sinh (), cosh (), tanh ().*

Експонентні функції: *cbrt (), exp (), expm1 (), log (), log10 (), log1p (), pow (), scalb (), sqrt ().*

Зведення в ступінь - *pow (),* квадратний корінь - *sqrt ().*

Наприклад, щоб витягти квадратний корінь з числа, застосовують метод sqrt.

```
double x = 4;
double y = Math.sqrt (x);
System.out.println (y); // Друкує число 2.0.
```

Перетворення й приведення числових типів

Часто виникає необхідність перетворити один числовий тип в інший. На рис. 10 показані дозволені перетворення.

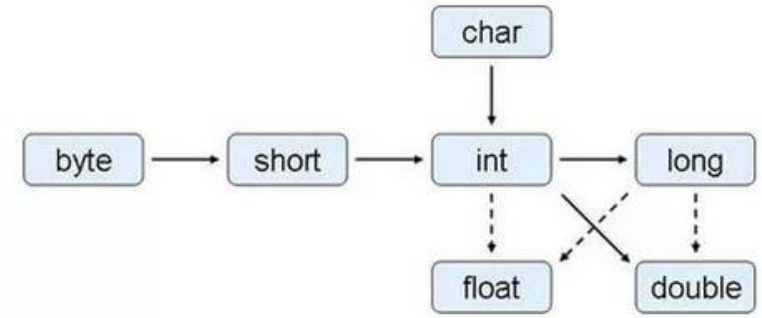

Рис.10 Дозволені перетворення числових типів

П'ять суцільних ліній зі стрілками позначають перетворення, які виконуються без втрати інформації. Три штрихові лінії позначають перетворення, при яких може відбутися втрата точності.

Якщо два значення об'єднуються бінарним оператором (наприклад n + f, де n - ціле число, а f - число з плаваючою точкою), то перед виконанням операції обидва операнда перетворюються в числа, що мають однаковий тип.

Якщо хоча б один з операндів має тип double, то другий теж перетворюється в число типу double.

В випадку, якщо хоча б один з операндів має тип float, то другий теж перетворюється в число типу float.

В випадку, якщо хоча б один з операндів має тип long, то другий теж перетворюється в число типу long. В іншому випадку обидва операнда перетворюються в числа типу int.

Перетворення чисел в мові Java можливі, однак при цьому може відбуватися втрата інформації. Такі перетворення називаються приведенням типів (cast). Синтаксично приведення типу задається парою дужок, усередині яких вказується бажаний тип, а потім ім'я змінної. наприклад:

```
double x = 9.997;
int nx = (int) x;
```

Побудова графіків функцій

Існує безліч платних і безкоштовних бібліотек, що реалізують різні прийоми побудови графіків. Наприклад, jfreechart. Але наше завдання - розібратися з принципами побудови графіків з використанням стандартного інструментарію.

Для малювання графіка необхідно вирішити, де він буде розташовуватися. Намалюємо наш графік безпосередньо на «поверхні» основного вікна аплету. Для цього використаємо клас Graphics. В методі paint розмістимо код для малювання графіка. Наш початковий код буде виглядати приблизно так:

```
import java.applet. *;
import java.awt. *;

public class MyApplet extends Applet {
```

```
// малювання аплету
public void paint (Graphics g)
{

}
}
```

Цей код створить пусте вікно аплета. Тепер потрібно намалювати графік. Спершу треба згадати принципи малювання на екрані, які в загальних рисах практично однакові для будь-якої мови програмування. Верхня ліва точка екрану (в нашому випадку - вікна аплету) є нулем декартовой двовимірної системи координат. Від'ємні точки за замовчуванням на екрані не відображаються (що природно), додатний напрямок за x - вправо, за y - вниз (рис.11).

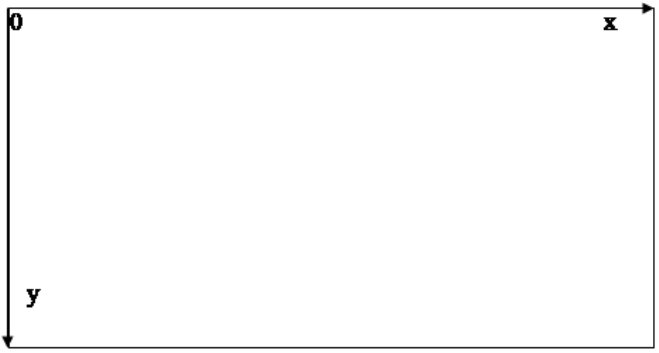

Рис.11. Декартова система координат на екрані

Для початку потрібно вирішити, якого розміру буде наш графічний аплет. Нехай розмір буде 800 на 800 точок. Відразу ж намалюємо систему координат. Нехай вісь абсцис знаходиться посередині (координата y = 400), а вісь ординат - на 10 точок від лівого краю екрана (координата x = 10). Для малювання ліній використовуємо метод drawLine. Аргументи методу - чотири координати (x0, y0, x1, y1), що задають дві точки. Відповідно, на контейнері буде намальований відрізок, що проходить через дві

точки. Код, що задає розмір контейнера-аплета і осі координат, буде виглядати наступним чином:

```
this.setSize (800, 800);
g.drawLine (10, 0, 10, 800);
g.drawLine (0, 400, 800, 400);
```

Тепер перейдемо безпосередньо до малювання графіка. Нехай це буде графік функції 350 * cos (0.05 * x). Для виклику математичної функції використовується клас Math. Оскільки він входить в стандартний пакет Java, то його не потрібно підключати за допомогою інструкції import.

При виконанні функції потрібно уважно дивитися на тип аргументів для неї і в разі необхідності виконати приведення типів. Так, функція cos класу Math, як і більшість інших функцій, працюють з аргументами типу double. Ми будемо підставляти в якості аргументу змінну циклу, проте вона буде цілочисельного типу (int).

У нашому випадку ми множимо аргумент на 0.05, що автоматично призведе результат до типу double. Однак малювати графік будемо за допомогою того ж методу drawLine, а його аргументи повинні бути цілочисельними. Для цього доведеться приводити результат до типу int.

Для перебору значень аргументу x організуємо цикл за допомогою інструкції for. Отже, код для побудови власне графіка функції:

```
for (int i = 1; i <790; i ++) {
g.drawLine (i + 10, (int) (400-350 * Math.cos (i * 0.05)), i +
11, (int) (400-350 * Math.cos (0.05 * (i + 1)) ));
}
```

Ми малюємо графік відрізками довжиною в одну точку. Отриманий графік можна підписати за допомогою інструкції drawString:

```
g.drawString ( "350 * cos (i * 0.05)", 15, 40);
```

Весь код аплету цілком:

```
import java.applet. *; // пакет, що містить
//клас Applet
import java.awt. *; // пакет, що містить
// клас Graphics
```

```
public class MyApplet extends Applet {
 // малювання аплету
public void paint (Graphics g)
{
this.setSize (800, 800);
g.drawLine (10, 0, 10, 800);
g.drawLine (0, 400, 800, 400);
for (int i = 1; i <790; i ++) {

g.drawLine (i + 10, (int) (400-350 * Math.cos (i *
 0.05)), i + 11, (int) (400-350 * Math.cos (0.05 * (i +
1)) ));
}
g.drawString ( "350 * cos (i * 0.05)", 15, 40);
}
}
```

Побудований графік з вікном аплета наведено на рис.12.

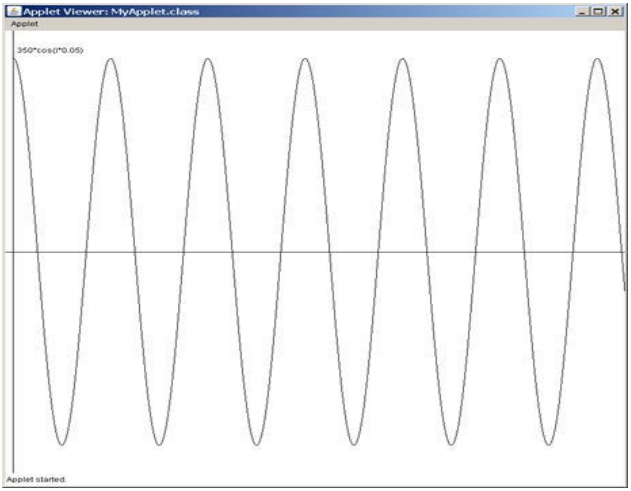

Рис.12. Результат побудови графіка функції

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити додаток, за допомогою якого можна побудувати два графіка функції в одному вікні різними кольорами. Варіанти двох функції вибрати зі списку нижче за наступним правилом: 1 графік – ваш номер варіанта, 2 графік – номер, якій є результатом додавання до номеру вашого варіанту 5. Якщо результат додавання більш 10 – відкидаєте від нього першу цифру. Масштаб по осях абсцис і ординат підібрати самостійно. Графіки підписати.
3. Підготувати звіт, в якому повинні бути код програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

Функція у для побудові графіку:

1. y = 100 * sin (0.1 * x-pi / 2);
2. y = 250 * cos (10 * x + pi);
3. y = 10 * exp (-0.1 * t) * sin (10 * t);
4. y = 12 * exp (-0.01 * t) * cos (t);
5. y = 130 * exp (-0.02 * t) * sin (2 * t);
6. y = 200 * (1-exp (-0.02 * t));
7. y = 100 * (1-exp (-0.1 * t));
8. y = 80 * (1-exp (-0.04 * t));
9. y =1 0 * sin (5 * t) + 8 * cos (7 * t);
10. y = 10 * sin (0.5 * t) * sin (10 * t).

Контрольні питання

1. Що таке функція?
2. Що таке аргумент?
3. Що означає Math в коді?
4. Знайдіть помилку у коді:

```
package mathmethods;
import java.math;

public class MathMethods {
```

```
public static void main (String [] args) {
final int MAX_VALUE = 10;
double d;
d = Math.random () * MAX_VALUE;
System.out.println ("d =" + d);
System.out.println ("Округлене до цілого =" + Math.round
(d));
System.out.println ("Найближче ціле,"+ "<= числа =" +
Math.floor (d));
System.out.println ("Найближче ціле,"+ "> = числа ="+
Math.ceil (d));
System.out.println ("Найближче ціле значення"+ "До
числа =" + Math.rint (d));
        }       }
```

ЛАБОРАТОРНА РОБОТА №10

Тема: Реалізація багатопотоковості в Java

Мета: придбання навичок роботи з потоками при програмуванні на мові Java.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

У Java реалізована повна підтримка потоків. Для реалізації багатопотоковості необхідно скористатися класом java.lang.Thread. В цьому класі визначено всі методи для створення потоків, управління їх станом і синхронізацію.

Деякі методи класу Thread:

currentThread () - повертає посилання на виконуваний в цей момент об'єкт класу Thread;

sleep () - переводить потік, що виконується в даний час, в режим очікування протягом зазначеного проміжку часу;

start () - починає виконання потоку. Цей метод призводить до виклику відповідного методу run ();

run () - фактичне тіло потоку. Цей метод викликається після запуску потоку;

stop () - зупиняє потік (застарілий метод);

isAlive () - визначає, чи є потік активним (запущеним або не зупиненим);

setPriority () - встановлює пріоритет потоку (приймає значення від MIN_PRIORITY до MAX_PRIORITY);

getPriority () - повертає пріоритет потоку;

wait () - переводить потік в стан очікування виконання умови, що визначається змінною умови;

join () - очікує, поки даний потік не завершить свого існування нескінченно довго або протягом деякого часу;

setDaemon () - зазначає даний потік як потік-демон або призначений для користувача потік. Коли в системі залишаться тільки потоки-демони, програма на мові Java завершить свою роботу;

isDaemon () - повертає ознака потоку-демона.

Існує два способи реалізації використання потоків у програмах:

- розширенням класу Thread;
- реалізацією інтерфейсу Runnable.

Реалізація Runnable

Найпростіший спосіб створення потоку - це оголошення класу, що реалізує інтерфейс Runnable. Runnable абстрагує одиницю виконуваного коду. Ви можете конструювати потік з будь-якого об'єкта, що реалізує інтерфейс Runnable. Щоб реалізувати Runnable, клас повинен оголосити єдиний метод run ():

public void run ()

Усередині run () ви визначаєте код, який, власне, становить новий потік. Важливо розуміти, що run () може викликати інші методи, використовувати інші класи, оголошувати змінні точно так же, як це робить головний потік. Єдиною відмінністю є те, що run () встановлює точку входу для

іншого, паралельного потоку всередині вашої програми. Цей потік завершиться, коли run () поверне управління.

Після того як буде оголошений клас, який реалізує інтерфейс Runnable, ви створите об'єкт типу Thread з цього класу. У Thread визначено кілька конструкторів. Той, який повинен використовуватися в даному випадку, виглядає наступним чином:

Thread (Runnable об'єкт потоку, String ім'я потоку)

У цьому конструкторі об'єкт потоку - це екземпляр класу, що реалізує інтерфейс Runnable. Він визначає, чі почнеться виконання потоку. Ім'я нового потоку передається в змінну *ім'я потоку*.

Після того, як новий потік буде створено, він не запускається до тих пір, поки ви не викличете метод start (), оголошений в класі Thread. По суті, start () виконує виклик run (). Метод start () показаний нижче:

void start ()

Розглянемо приклад, що створює новий потік і запускає його на виконання:

Приклад 1. Створення потоку за реалізацією інтерфейсу Runnable

```
package runnableexample;

class MyThread implements Runnable
{
   Thread thread;
MyThread() {
thread = new Thread(this, "Додатковий потік");
System.out.println("Створено додатковий потік " + thread);
thread.start();
   }
   @Override
public void run() {
try {
for (int i = 5; i > 0; i--) {
System.out.println("\t додатковий потік: " + i);
Thread.sleep(500);
        }
```

```
        } catch (InterruptedException e) {
System.out.println("\t додатковий потік перервано");
}
System.out.println("\t додатковий потік завершено");
}
}
public class RunnableExample
{
public static void main(String[] args)
   {
new MyThread();
try {
for (int i = 5; i > 0; i--) {
System.out.println("Головний потік: " + i);
Thread.sleep(1000);
          }
        } catch (InterruptedException e) {
System.out.println("Головний потік перервано ");
}
        System.out.println("Головний потік завершено");
    }
}
```

Розширення Thread.

Другий спосіб створення потоку - це оголосити клас, який розширює Thread, а потім створити екземпляр цього класу. У цьому класі необхідно перевизначити метод run (), який є точкою входу для нового потоку. Також потрібно викликати start () для запуску виконання нового потоку.

Нижче наведено приклад програми. У прикладі ChickenEgg розглядається паралельна робота двох потоків (головний потік і потік Egg), в яких йде суперечка, «що було раніше, яйце чи курка?». Кожен потік висловлює свою думку після невеликої затримки, що формується методом ChickenEgg.getTimeSleep (). Перемагає той потік, який останнім говорить своє слово.

Приклад 2. Створення потоку з використанням розширення Thread.

```
package chickenegg;
```

```
import java.util.Random;

class Egg extends Thread{
   @Override
public void run()
   {
for(int i = 0; i < 5; i++) {
try {
            // Призупиняємо потік
sleep(ChickenEgg.getTimeSleep());
System.out.println ("Яйце");
}catch(InterruptedException e){}
      }
   }
}

public class ChickenEgg
{
public static int getTimeSleep()
   {
final Random random = new Random();
int tm = random.nextInt(1000);
if (tm < 10)
tm *= 100;
else if (tm < 100)
tm *= 10;
return tm;
   }
public static void main(String[] args)
{
      Egg egg = new Egg (); // Створення потоку
      System.out.println("Починаємо спір: хто з'явився
першим?");

egg.start(); // Запуск потоку
for(int i = 0; i < 5; i++) {
try {
            // Призупиняємо потік
```

```
Thread.sleep(ChickenEgg.getTimeSleep());
System.out.println("Курка");
}catch(InterruptedException e){}
}
      if(egg.isAlive()) { // Чи сказало яйце останнє слово?
try{
            egg.join(); // Чикаємо, поки яйце завершить
//своє висловлювання
}catch(InterruptedException e){}

         System.out.println("Першим з'явилося  яйце !!!");
      } else {
         // якщо опонент вже закінчив висловлюватися
         System.out.println("Першою з'явилася курка !!!");
      }
      System.out.println("Спір завершено");
   }
}
```

Неможливо точно передбачити, який потік закінчить висловлюватися останнім. Під час наступного запуску «переможець» може змінитися. Це відбувається внаслідок так званого «асинхронного виконання коду». Асинхронність забезпечує незалежність виконання потоків. Або, іншими словами, паралельні потоки незалежні один від одного, за винятком випадків, коли бізнес-логіка залежності виконання потоків визначається передбаченими для цього засобами мови.

Синхронізація потоків synchronized

У процесі функціонування потоки часто використовують загальні ресурси програми, визначені поза потоком. Якщо кілька потоків почнуть одночасно вносити зміни в загальний ресурс, то результати виконання програми можуть бути непередбачуваними.

Розглянемо наступний приклад:

```
packagesynchronizedthread;
class CommonObject
{
```

```
int counter = 0;
}
class CounterThread implements Runnable
{
   CommonObject res;
CounterThread(CommonObject res)
   {
      this.res = res;
   }
   @Override
public void run()
   {
//      synchronized(res) {
          res.counter = 1;
for (int i = 1; i < 5; i++){
System.out.printf("'%s' - %d\n",
Thread.currentThread().getName(), res.counter);
res.counter++;
try {
Thread.sleep(100);
             }
catch(InterruptedException e){}
          }
//      }
   }
}
public class SynchronizedThread
{
public static void main(String[] args) {
       CommonObject commonObject= new
CommonObject();
for (int i = 1; i < 6; i++) {
          Thread t = new Thread(new
CounterThread(commonObject));
t.setName("Потік " + i);
t.start();
}
   }
```

```
}
```

У прикладі визначений загальний ресурс у вигляді класу CommonObject, в якому є цілочисельне поле counter. Даний ресурс використовується внутрішнім класом, що створює потік CounterThread для збільшення в циклі значення counter на одиницю. При старті потоку полю counter присвоюється значення 1. Після завершення роботи потоку значення res.counter має дорівнювати 4.

У головному класі програми SynchronizedThread.main запускається п'ять потоків. Тобто, кожен потік повинен в циклі збільшити значення res.counter з одиниці до чотирьох; і так п'ять разів. Але результат роботи програми, що відображається в консолі, буде іншим (рис.13):

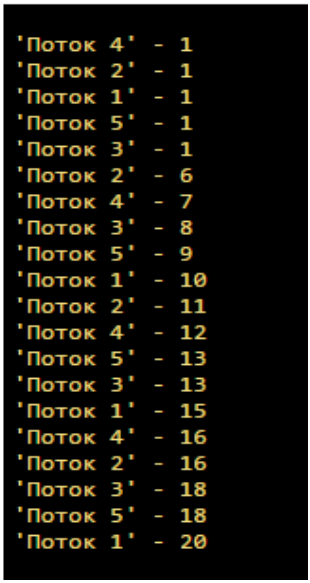

Рис.13. Перший варіант виконання програми

Тобто, із загальним ресурсів res.counter працюють всі потоки одночасно, почергово змінюючи значення. Щоб уникнути

подібної ситуації, потоки необхідно синхронізувати. Одним із способів синхронізації потоків пов'язаний з використанням ключового слова synchronized. Оператор synchronized дозволяє визначити блок коду або метод, який повинен бути доступний тільки одному потоку. Можна використовувати synchronized в своїх класах для визначення синхронізованих методів або блоків. Але не можна використовувати synchronized в змінних або атрибутах у визначенні класу.

Таким чином, у прикладі слід видалити рядкові коментарі в класі CounterThread (вони підкреслені), після чого загальний ресурс буде блокуватися, як тільки його захопить один з потоків; інші потоки будуть чекати в черзі звільнення ресурсу. Результат роботи програми при синхронізації доступу до загального ресурсу різко зміниться (рис.14):

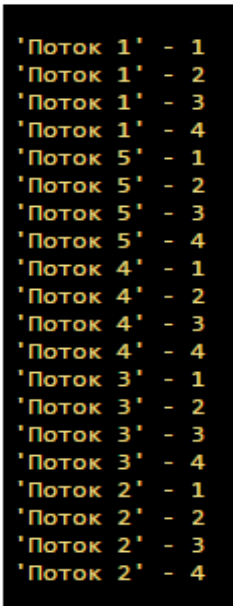

Рис.14. Другий варіант виконання програми (синхронізація потоків)

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. Розробити багатопоточний додаток (відповідно варіанту).
3. Підготувати звіт, в якому повинні бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Створити два потоки. Перший потік створює запис у файлі випадкових чисел, другий здійснює читання даних з цього файлу і виведення їх на екран.

2. Створіть додаток, який в окремому потоці обчислює значення W і безперервно оновлює його в призначеному для користувача інтерфейсі. Для розрахунку числа використовувати наступну формулу:

$W = 1 + \sin(x) - 2\cos(x) + 4\sin^2(x) - 8\cos^2(x)$

3. Створити два потоки. Перший шукає числа Фібоначчі (кожне наступне число дорівнює сумі двох попередніх чисел), другий - прості числа. Результат роботи кожного потоку зберігаються в окремих файлах. Після зупинки потоку - програма робить аналіз файлів, виводить їх на екран, а так само показує кількість знайдених чисел Фібоначчі і простих чисел.

4. Імітація грального автомату - 3 потоку, що генерують числа від 0 до 9. При натисканні кнопки потоки зупиняються і результат аналізується. При аналізі використовувати такі комбінації (три однакових числа, два однакових числа, три одиниці, три сімки, дві одиниці, є четвірка).

5. Потік управляє переміщенням прямокутника на єкрані за координатою X. Створити три потоки, які управляють переміщенням прямокутників (кожен своїм) і влаштувати "перегони" серед них.

6. Створити два потоки. Перший потік створює читання чисел з файла, другий здійснює запис цих чисел у другий файлу й виводе їх на екран.

7. Створіть додаток, який в окремому потоці обчислює значення P і безперервно оновлює його в призначеному для

користувача інтерфейсі. Для розрахунку числа використовувати наступну формулу:

P = 4-4/3+4/5-4/7+4/9 - …

8. Створити два потоки. Перший обчислює суму чисел заданої матриці, другий шукає максимальне число цієї матриці. Результат роботи кожного потоку зберігаються в окремих файлах. Після зупинки потоку - програма робить аналіз файлів, виводить їх на екран.

9. Імітація грального автомату - 5 потоків, що генерують числа від 0 до 9. При натисканні кнопки потоки зупиняються і результат аналізується. При аналізі використовувати такі комбінації (всі однакові числа, чотири однакових числа, три однакових числа, п’ять п’ятірок, п’ять одиниць).

10. Потік управляє переміщенням кнопки на єкрані за координатою Y. Створити три потоки, які управляють переміщенням кнопок (кожен своїм) і влаштувати "перегони" серед них.

Контрольні питання

1. Які два способи використовуються для реалізації потоків в Java?
2. Які операції над потоками визначені в Java?
3. Що таке потоки-демони і чим вони відрізняються від звичайних потоків?
4. Як задати і отримати значення пріоритету для потоку?
5. Як можна створити групу потоків, і які операції визначені для групи потоків в Java?

ЛАБОРАТОРНА РОБОТА №11

Тема: Класи-колекції

Мета: придбання навичок роботи з класами-колекціями при програмуванні на мові Java.

Час виконання роботи: 4 години.

Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X).

Короткі теоретичні відомості

Під час програмування доводиться працювати з великою кількістю об'єктів. Зручно мати засоби групування об'єктів. Для цих цілей в Java розроблено набір інтерфейсів і класів під назвою колекції. В основі ієрархії колекцій знаходиться інтерфейс Collection.

Collection – базовий інтерфейс. На його основі в структурі колекцій є ще декілька інтерфейсів, які розширюють базовий інтерфейс Collection. Зокрема, List, Set та SortedSet.

Інтерфейси колекцій

1). ***Collection*** - самий загальний інтерфейс, його основні методи:

add (T e) - додати елемент e;

clear () – очистити;

addAll (Collection col) - додати всі елементи іншої колекції, зі схожим типом даних;

contains (Object o) – перевіряє, чи містить колекція елемент;

isEmpty () – перевіряє, чи є порожньою колекція;

remove (Object o) - видаляє елемент;

removeAll (Collection col) - видаляє всі елементи, які є в колекції col;

size () - повертає кількість елементів в колекції;

containsAll (Collection col) - перевіряє, чи містяться всі елементи col в колекції;

toArray (T [] a) - повертає масив, який містить всі елементи колекції, на вхід приймає масив, який буде заповнений ними;

retainAll (Collection col) - видаляє всі елементи, які не належать col;

toArray () - повертає масив об'єктів, який містить всі елементи колекції.

2). ***List*** - пронумерований список, містить всі методи інтерфейсу Collection, а також свої. Як і в звичайному масиві всі елементи списку нумеруються, починаючи з нуля, і до будь-якого елементу можна звернутися за індексом:

get (int index) - отримуємо елемент за індексом;

add (int index, T e) - вставляє елемент в позицію;

indexOf (Object obj) - повертає перше входження елемента у список;

lastIndexOf (Object obj) - повертає останнє входження;

set (int index, T e) - замінює елемент в позиції index;

subList (int from, int to) - повертає новий список, що є частиною головного.

3). ***Set*** - множина. Містить всі операції інтерфейсу Collection, але деякі з них мають інший зміст. Наприклад, операція add додає тільки унікальні елементи, проте операції пошуку елемента відбуваються швидше ніж в списку.

4). ***Queue*** - черга. Реалізує основні операції черги:

poll () - повертає перший елемент і видаляє його;

peek () - повертає перший елемент черги, не видаляючи його;

offer (T e) - додає елемент у кінець черги.

5). ***Map.*** Інтерфейс Map із пакету java.util описує колекцію, що складається з пар "ключ - значення". У кожного ключа тільки одне значення, що відповідає математичному поняттю однозначної функції або відображення (map). Таку колекцію часто називають ще словником (dictionary) або асоціативним масивом (associative array).

6). ***Iterator.*** В Java API введено інтерфейс Iterator, що описує спосіб обходу всіх елементів колекції. Інтерфейс Iterator, визначений у пакеті java.util. У кожної колекції є метод iterator (), який повертає реалізацію інтерфейсу Iterator для зазначеної колекції. Отримавши цю реалізацію, можна обходити колекцію в порядку, визначеному цим ітератором, за допомогою методів, описаних в інтерефейсі Iterator.

В інтерфейсі Iterator описані три методи:

- метод hasNext () повертає true, якщо обхід ще не завершений;

- метод next () робить поточним наступний елемент колекції і повертає його у вигляді об'єкта класу Object;

- метод remove () видаляє поточний елемент колекції.

Итератор - це покажчик на елемент колекції. При створенні ітератора покажчик встановлюється перед першим елементом. Метод next () переміщає покажчик на перший елемент. Наступне застосуванн я методу next () переміщує покажчик на другий елемент колекції. Під час останнього застосування методу next () виводиться покажчик на останній елемент колекції.

```
Vector v = new Vector ();
.....
for (int i = 0; i <v.size (); i ++)
     System.out.print (v.get (i) + ".");
System.out.println ();
Iterator it = v.iterator (); // Отримуємо ітератор
//вектора
try {
    while (it.hasNext ()) // Поки вектор має
// елементи,
     System.out.println (it.next ()); // виводимо
//поточний елемент
}
catch (Exception e) {
}
```

Класи колекцій

Vector - клас реалізує звичайний список. У класі Vector з пакета java.util зберігаються елементи типу object, а значить, будь-якого типу. Кількість елементів може бути будь-яким і наперед не визначатися (реалізація динамічного масиву). Елементи отримують індекси 0, 1, 2, ... До кожного елементу вектора можна звернутися за індексом, як і до елементу массиву.

Приклад використання:

```
Vector objects = new Vector();
objects.addElement(new Button("Hello"));
objects.addElement(new Panel());
```

Цей клас вважається застарілим. Більш сучасним та швидким є клас ArrayList, що має спільни риси з Vector**.**

ArrayList - реалізує інтерфейс List, тобто в тих випадках, коли вам потрібен упорядкований список на основі масиву вам слід використовувати його.

Приклад 1 використання цього классу:

```
package arraylisttest;
import java.util.*;
import java.io.*;

public class ArrayListTest {
ArrayList lst = new ArrayList();
Random generator = new Random();

void addRandom() {
lst.add(new Integer(generator.nextInt()));
}

public String toString() {
return lst.toString();
}

public static void main(String args[]) {
ArrayListTest tst = new ArrayListTest();
for(int i = 0; i < 100; i++ )
tst.addRandom();
System.out.println("Сто випадкових чисел:
"+tst.toString());
}
}
```

У цьому прикладі застосовується клас-колекція ArrayList. Random - клас з java.util. Розширює можливості класу Math (генерація випадкових чисел). Integer - так званий wrapper-клас (клас-обгортка) для цілих (int). Він використаний тому, що в колекцію не можна занести дані елементарних типів, а тільки об'єкти класів.

Клас ArrayListTest має два поля - поле lst класу ArrayList і поле generator класу Random, що використовується для генерації випадкових чисел. Метод addRandom () генерує і заносить до колекції чергове випадкове число. Метод toString () просто

звертається до методу toString () класу ArrayList, який забезпечує формування уявлення списку у вигляді рядка.

Метод main (...) створює об'єкт класу ArrayListTest і організовує цикл породження ста випадкових чисел із занесенням їх до колекції за допомогою метода addRandom (). Після цього він друкує результат.

Цей приклад не демонструє особливих переваг колекцій, а лише техніку їх використання. З нього видно, що додати елемент в колекцію можна методом add (...) класу ArrayList і при цьому ми ніде не вказуємо розмір колекції.

LinkedList - двонаправлений список, який реалізує інтерфейс List і Queue. Якщо використовувати його як List, то швидкість операцій вставки і видалення зросте, а швидкість операцій отримання значення за індексом навпаки впаде. LinkedList є гарною реалізацією черги.

Stack - стек є підкласом класу Vector, який реалізує простий механізм типу "останній увійшов - перший вийшов" (LIFO). Розглянемо його операції:

push (T e) - додає елемент до вершини стека;

pop () - видаляє і повертає зі стека верхній елемент;

peek () - повертає верхній елемент, не видаляя його;

empty () – перевіряє, чи порожній стек;

search (Object item) - повертає місці, на якому знаходиться елемент в стеку.

Приклад застосування класу Stack:

```
Stack stack = new Stack();
stack.push("aaa");
stack.push("bbb");
 stack.push("ccc");
for (int i = 0; i < stack.size(); i++)
      System.out.print(stack.get(i) + " - ");
System.out.println();
stack.pop();
for (int i = 0; i < stack.size(); i++)
     System.out.print(stack.get(i) + " - ");
System.out.println();
```

HashMap. Відображення в Java представлені кількома класами. Найбільш поширеним класом відображень є HashMap,

який реалізує інтерфейс Map і успадковується від класу AbstractMap.

Для розуміння того, як це працює на практиці, розглянемо приклад простого додатка.

Приклад 2.

```
package hashmapdemo;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class HashMapDemo {
   public static void main(String[] args) {
      HashMap hashMap = new HashMap<>();

      System.out.println("Adding elements into
hashMap...");
      hashMap.put("Proselyte","Java");
      hashMap.put("AsyaSmile","UI/UX");
      hashMap.put("Peter","C++");
      hashMap.put("Ann","PHP");

      System.out.println("Initial hashMap:");
      System.out.println(hashMap);


System.out.println("\n====================\n");
      System.out.println("Initial hashMap content
using Set:");
      Set set = hashMap.entrySet();

      for (Object element : set) {
         Map.Entry mapEntry = (Map.Entry) element;
         System.out.println(mapEntry.getKey() + ":"
+ mapEntry.getValue());
      }
```

```
System.out.println("\n====================\n");

                System.out.println("Modifying Proselyte's
specialty...");
                String specialty = (String)
hashMap.get("Proselyte");
                specialty += " Developer (Changed)";
                hashMap.put("Proselyte", specialty);


System.out.println("\n====================\n");
                System.out.println("Final hashMap content
                using Set:");
                set = hashMap.entrySet();

                for (Object element : set) {
                   Map.Entry mapEntry = (Map.Entry) element;
                   System.out.println(mapEntry.getKey() + ":"
                + mapEntry.getValue());
                }

                System.out.println("\n===============
                ====\n");
            }
}
```

HashSet - реалізує інтерфейс Set на основі хешів.

Hashtable. Клас Hashtable розширює абстрактний клас Dictionary. В об'єктах цього класу зберігаються пари "ключ - значення".

Порядок виконання роботи

1. Проаналізувати та відпрацювати приклади коду, що наведено у теоретичному матеріалі.
2. У звіт додати аналіз коду та скріншоти результатів прикладу 2.
3. Розробити додаток з застосуванням класів-колекцій (відповідно варіанту).

4. Підготувати звіт, в якому повинно бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

1. Увести рядки з файлу, записати їх в стек. Вивести рядки в файл в зворотному порядку.
2. Увести число, занести його цифри в стек. Вивести в число, у якого цифри йдуть в зворотному порядку.
3. Скласти два багаточлена заданого ступеня, коефіцієнти багаточленів зберігаються в об'єкті HashMap.
4. Згенеруйте масив цілих чисел, використовуючи клас Random (). Використовуючи методи класів-колекцій найдіть унікальні (що не дублюються) числа.
5. Згенеруйте масив цілих чисел, використовуючи клас Random (). Використовуючи методи класів-колекцій створіть масив, який містить всі числа з першого масиву окрім непарних.
6. Визначити клас Stack. Оголосити об'єкт класу. Ввести послідовність символів і вивести її в зворотному порядку.
7. Помножити два багаточлени заданого ступеня, якщо коефіцієнти багаточленів зберігаються в списках.
8. Згенеруйте масив цілих чисел, використовуючи клас Random (). Використовуючи методи класів-колекцій створіть масив, який містить всі додатні числа з першого масиву.
9. Визначити клас Set на основі безлічі цілих чисел, n = розмір. Створити методи для визначення перетину множин.
10. Визначити клас Set на основі безлічі цілих чисел, n = розмір. Створити методи для визначення об'єднання множин.

Контрольні питання

1. Які класи колекцій дають доступ до будь-якого елементу?

2. Що таке черга і стек, назвіть відмінності між ними?
3. Назвіть відмінності між Vector і ArrayList?
4. Що відбувається за цим кодом?
   List reversedList = Collections.reverse(list);

ЛАБОРАТОРНА РОБОТА №12
Тема: Робота з базами даних в Java
Мета: отримати навички роботи з базами даних MS Access із застосуванням драйверу UCanAccess
Час виконання роботи: 4 години.
Програмне забезпечення: JDK (версія 8uXXX), IDE NetBeans (версія 8.X), UCanAccess драйвер.

Короткі теоретичні відомості

JDBC - (Java DataBase Connectivity - з'єднання з базами даних на Java) – платформно незалежний промисловий стандарт взаємодії Java-додатків з різними СУБД, реалізований у вигляді пакета java.sql, що входить до складу Java SE. JDBC дозволяє встановлювати з'єднання з БД за допомогою різних типів підключень, відсилати SQL-запити і обробляти результати.

Існують чотири типи драйверів JDBC:

- драйвер, який реалізує методи JDBC викликами функцій ODBC. Це так званий міст (bridge) JDBC-ODBC. Безпосередній зв'язок з базою при цьому здійснює драйвер ODBC. ***Увага!*** Цей драйвер видалено починаючи з версії Java SE 8. Замість JDBC-ODBC моста використовується UCanAccess драйвер.
- драйвер, який реалізує методи JDBC викликами функцій API самої СУБД;
- драйвер, який реалізує методи JDBC викликами функцій мережевого протоколу, незалежного від СУБД;

– драйвер, який реалізує методи JDBC викликами функцій мережевого протоколу СУБД.

Доступ і обробка даних за допомогою JDBC включає три етапи:

– установка зв'язку між Java-програмою і диспетчером бази даних (підключення до бази даних).
– передача SQL-команди в базу даних за допомогою об'єкта інтерфейсу Statement пакета java.sql.
– читання отриманих результатів із бази даних і використання їх в програмі.

Порядок виконання роботи:

Розглянемо приклад підключення бази даних MS Access у Java-додатку. Як було зазначено раніше в версіях Java SE 8 драйвер ODBC видалено. Замість нього можна застосовувати UCanAccess, що є чистою реалізацією java-драйвера JDBC, який дозволяє здійснювати читання / запис у базу даних Access (.mdb і .accdb файлів).

На першому етапі для роботи з базою даних в Java необхідно виконати наступні дії.

1. Попередньо необхідно мати базу даних. Як приклад створимо у СУБД MS Access базу даних mydatabase.accdb. У базі розмістимо таблицю student з наступною структурою: nom (лічильник), fio (коротке текстове поле), data (числове).

2. Скачати драйвер UCanAccess за адресою http://ucanaccess.sourceforge.net/site.html.

3. UCanAccess має всі необхідні jar-файли у скачаному дистрибутиві. Коли ви розпакувати його, то побачите щось на зразок:

ucanaccess-4.0.2.jar
/lib/
commons-lang-2.6.jar
commons-logging-1.1.1.jar
hsqldb.jar
jackcess-2.1.6.jar

На наступному етапі потрібно додати ці п’ять jar-файлів до java-проекту. Для цього необхідно створити java-проект у середовищі NetBeans. Клацнути правою кнопкою миші на імені проекту и вибрати пункт Властивості. У вікні Властивості проекту обрати у лівій панелі пункт Бібліотеки (рис. 15).

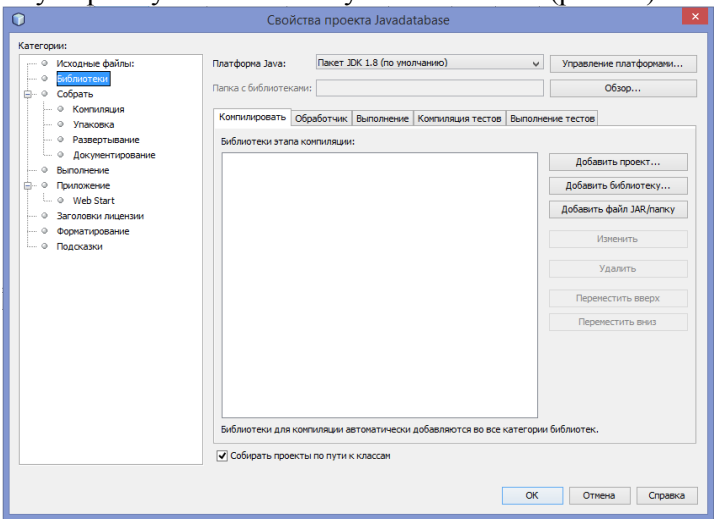

Рис. 15. Опція Бібліотеки вікна Властивості проекту

Натисніть кнопку Додати файл Jar/папку. Поетапно додайте всі п’ять jar-файлів (чотири файли вкладено у папці lib). Після цього в проекті можна побачити ці файли (рис. 16).

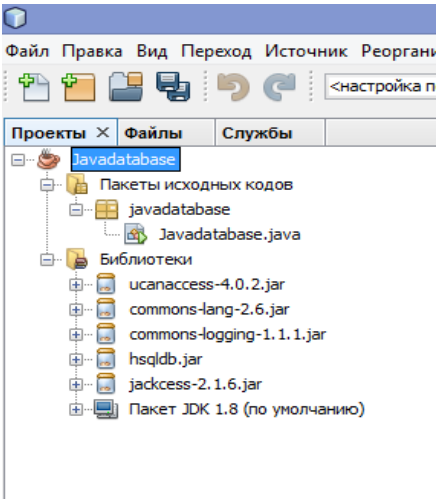

Рис. 16. Структура проекту після підключення UCanAccess драйверу

4. У програмному коді створимо підключення бази даних mydatabase.accdb за допомогою драйверу Ucanaccess.

```
String driver="net.ucanaccess.jdbc.UcanaccessDriver";
Class.forName(driver);
Connection  conn;
conn=DriverManager.getConnection("jdbc:ucanaccess://C:/
mydatabase.accdb");
```

Змінна driver містить повну назву драйверу. Метод GetConnection за допомогою параметра URL (C:/mydatabase.accdb) знаходить java.sql.Driver відповідної бази даних і викликає у нього метод підключення.

Під час з'єднання можна отримати об'єкт java.sql.Statement для виконання запитів до бази. Це робиться так:

```
// формуємо порожню команду  SQL
Statement sq = db.createStatement ();
// Визначаємо рядок запиту на вибірку
 String sq_str = "SELECT * FROM student";
// Виконуємо команду SQL
ResultSet rs = sq.executeQuery (sq_str);
```

І це майже все. У наведеному прикладі сформований рядок запиту sq_str до таблиці student і видана команда на його виконання rs.

Ясно, що змінна rs буде містити всі знайдені записи у таблиці student. Для доступу до цих записів можна організувати наступний цикл:

```
while (rs.next ())
{
String s = rs.getString ("fio");
System.out.println ( s);
}
System.out.println ("OK");
```

Команда rs.next () видає черговий запис і повертає true або false залежно від того, досягнуто кінець набору чи ні. Команда rs.getstring ("fio") повертає вміст поля fio, яке повинно мати тип char (varchar). Для отримання значень полів іншого типу слід використовувати такі команди:

```
// - для поля таблиці типу boolean
        rs.getBoolean ();
// - для поля таблиці типу int
        rs.getInt ();
// - для поля таблиці типу Date
        rs.getDate ();
// - для поля таблиці ти для поля Numeric
        rs.getFloat ();
```

Далі наведено повний текст програми.

```
package javadatabase;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Javadatabase {
```

```
        public static void main(String[] args) throws
ClassNotFoundException, SQLException {
            String driver =
"net.ucanaccess.jdbc.UcanaccessDriver";
            Class.forName(driver);
            Connection conn;
            conn =
DriverManager.getConnection("jdbc:ucanaccess://C:/mydatabase.a
ccdb");
        // формуємо порожню команду  SQL
            Statement sq = conn.createStatement();
        // Визначаємо рядок запиту на /вибірку
         String sq_str = "SELECT * FROM student";
        // Виконуємо команду SQL
         ResultSet rs = sq.executeQuery(sq_str);
            while (rs.next()) {
               String s = rs.getString("fio");
               System.out.println(s);
            }
            System.out.println("OK");

       }
    }
```

Результат виконання програми буде наступний (рис.17):

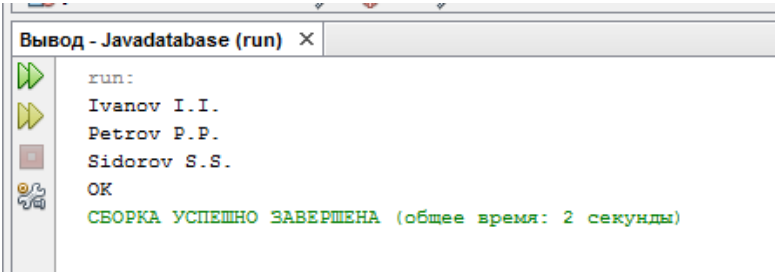

Рис. 17. Результат виконання програми

5. Розробити додаток, в якому буде відображатися інформація з бази даних MS Access (за тематикою з урахуванням

варіанту). База даних повинна мати мінімум одну таблицю (структуру бази обирає студент). Дані представити у вигляді таблиць (використовувати компонент JTable). Крім відображення інформації з БД, додаток повинен дозволяти додавати і видаляти записи.

6. Підготувати звіт, в якому повинно бути коди програми з коментарем та скріншоти опрацювання програми, відповіді на контрольні запитання. До звіту додаються файли проекту.

Варіанти завдань до лабораторної роботи

Тематика бази даних:

1. Облік кадрів.
2. Облік продажу квитків для авіакомпанії.
3. Облік продажу туристичних путівок.
4. Облік прийому пацієнтів для медичного закладу.
5. Облік заробітної плати співробітників.
6. Облік продажу квитків у кінотеатрі.
7. Електронний деканат (розклад іспитів і заліків).
8. Електронний деканат (відомості та екзаменаційні листки)
9. Електронний деканат (розклад занять).
10. Електронний деканат (облік студентів).

Контрольні питання

1. Що таке драйвер бази даних?
2. Для чого використовується менеджер драйверів?
3. Що робить клас Statement?
4. Для чого потрібен клас ResultSet?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Шилдт. Г. Java 8. Полное руководство / Герберт Шилдт. – 9-е изд. : пер. с англ. . — М.: ООО И.Д.Вильямс», 2015. — 1376 с.
2. Монахов В. В. Язык программирования Java и среда NetBeans / В.В. Монахов. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. – 704 с.
3. Сеттер Р. В. Изучаем JAVA на примерах и задачах/ Р.В. Сеттер**.** — СПб.: Наука и Техника, 2016. — 240 с.
4. Васильев А**.** Н**.** Java. Объектно-ориентированное программирование: Учебное пособие / А.Н. Васильев. — СПб.: Питер, 2011. — 400 с.
5. Хорстманн К.С. Java SE 8. Базовый курс / К.С. Хорстманн. – пер. с англ. – М. : ООО «И.Д. Вильямс», 2015. – 464 с.
6. Хорстманн, К. С. Java. Библиотека профессионала, том 1. Основы. / К. С. Хорстманн, Г. Корнелл. – 9-е изд. : Пер. с англ. — М. : ООО "И.Д.Вильямс", 2014. — 864 с.
7. МакГрат М.. Программирование на Java для начинающих / Майк МакГрат ; [пер. с англ. М.А. Райтмана]. – Москва : Издательство «Э», 2016**.** – 192 с.
8. Машнин Т. С. Современные Java-технологии на практике / Т. С. Машнин. — СПб.: БХВ-Петербург, 2010. — 560 с.

9. Седжвик Р. Алгоритмы на языке Java / Р. Седжвик, К. Уэйн. – 4-е изд. : пер. с англ. – М. : ООО "И.Д.Вильямс", 2013. — 848 с.
10. Шилдт. Г. SWING : руководство для начинающих / Герберт Шилдт. – пер. с англ. . — М.: ООО И.Д.Вильямс», 2007. — 704 с.
11. Фридл Д. Регулярные выражения / Джефри Фридл. – 3-е изд. –СПб., М..: изд-во «Символ», 2008. – 598 с.
12. Лафоре Р. Структуры данных и алгоритмы в Java. / Роберт Лафоре. – 2-е изд. — СПб.: Питер, 2013. — 704 с.
13. Farrell J. Java Programming / Joyce Farrell. – Boston : Course Technology, Cengage Learning, 2014. – 1118 p.

14. Блох Д. Java. Эффективное программирование / Джошуа Блох. – 2-е изд. : пер. с англ. – М. : Изд-во «Лори», 2014. – 461 с.
15. Портянкин И. А. Swing : эффектные пользовательские интерфейсы : Java Foundation Classes / Иван Александрович Портянкин. – СПб.: Питер, 2005. – 523 с.

Навчально-методичне видання

ПЕРЕЯСЛАВСЬКА Світлана Олександрівна
ЖУКОВА Вікторія Миколаївна
СМАГІНА Ольга Олександрівна

JAVA ПРОГРАМУВАННЯ

Методичні рекомендації до лабораторних робіт для студентів спеціальності 123 – „Комп'ютерна інженерія"

Редактор – Переяславська С.О.
Комп'ютерний макет –Жукова В. М., Смагіна О.О.

Здано до склад. 28.02.2018 р. Підп. до друку 30.03.2018 р.
Формат 60x84 1/16. Папір офсет. Гарнітура Times New Roman.
Друк ризографічний. Ум. друк. арк. 6,39. Наклад 50 прим.
Зам. № 19.

Видавець і виготовлювач
Видавництво Державного закладу
„Луганський національний університет імені Тараса Шевченка"
пл. Гоголя, 1, м. Старобільськ, 92703. Тел./факс: (06461) 2-26-70.
e-mail: mail@luguniv.edu.ua
Свідоцтво суб'єкта видавничої справи ДК № 3459 від 09.04.2009 р.