

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»

# **ПРОГРАМУВАННЯ. ОБ'ЄКТНО-ОРІЄНТОВНЕ ПРОГРАМУВАННЯ: КОМП'ЮТЕРНИЙ ПРАКТИКУМ**

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського  
як навчальний посібник для студентів,  
які навчаються за спеціальністю  
151 «Автоматизація та комп'ютерно-інтегровані технології»,  
спеціалізацією «Технічні та програмні засоби автоматизації»*

Київ  
КПІ ім. Ігоря Сікорського  
2020

Програмування. Об'єктно-орієнтовне програмування: комп'ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», спеціалізації «Технічні та програмні засоби автоматизації» / КПІ ім. Ігоря Сікорського; уклад.: Д. О. Ковалюк. – Електронні текстові данні (1 файл: 1,4 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 29 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 8 від 09.04.2020 р.)*

*за поданням Вченої ради інженерно-хімічного факультету (протокол № 2 від 24.02.2020 р.)*

Електронне мережне навчальне видання

## **ПРОГРАМУВАННЯ. ОБ'ЄКТНО-ОРІЄНТОВНЕ ПРОГРАМУВАННЯ КОМП'ЮТЕРНИЙ ПРАКТИКУМ**

Укладач: *Ковалюк Дмитро Олександрович, канд. техн. наук, доц.*

Відповідальний  
редактор *Жученко Анатолій Іванович, д-р техн. наук, проф.*

Рецензенти: *Гулієнко Сергій Валерійович., канд. техн. наук, доцент*

Посібник призначений для формування у студентів комплексу знань, умінь та досвіду, необхідних для створення комп'ютерних систем керування та інформаційних систем. Дозволяє скорочено за формою, але всебічно за змістом викласти студентам принципи об'єктно-орієнтованого програмування, які використовуються при розробці програмного забезпечення даних систем.

© КПІ ім. Ігоря Сікорського, 2020

## ЗМІСТ

Вступ.....	4
Робота 1	
Налаштування середовища JAVA.....	5
Робота 2	
Типи даних, операції, конструкції .....	9
Робота 3	
Основи роботи з об'єктами.....	12
Робота 4	
Реалізація принципів ООП.....	14
Робота 5	
Робота з масивами об'єктів.....	15
Робота 6	
Вкладені класи.....	18
Робота 7	
Поліморфізм.....	19
Робота 8	
Exceptions.....	24
Робота 9	
Інтерфейси.....	25
Робота 10	
Колекції.....	26

## **ВСТУП**

Автоматизація виробничих процесів є важливою задачею, що дозволяє суттєво підвищити продуктивність праці, покращити якість продукції, зберегти енергоресурси, оптимізувати склад обслуговуючого персоналу, підвищити надійність роботи.

Успішне розв'язання усіх вище перерахованих задач досягається впровадженням у виробництво автоматичних систем управління, складовими яких виступає програмне забезпечення. Знання концепцій розробки програмного забезпечення, зокрема об'єктно-орієнтованого програмування, відіграє при цьому вирішальну роль.

## РОБОТА №1

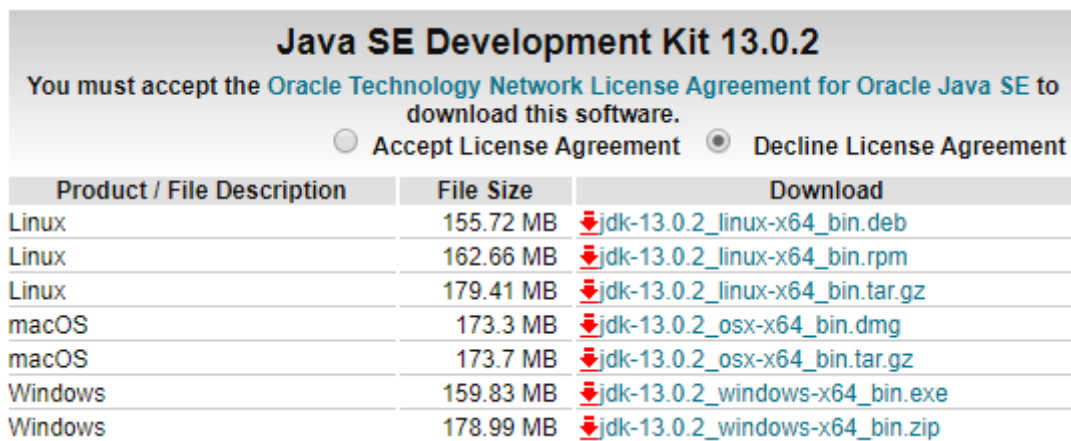
### Налаштування середовища JAVA

**Мета роботи** – навчитися встановлювати Java-платформу (JDK), написати елементарну програму, запустити її через IDE (Netbeans, IntelliJ) та за допомогою командної строки.

#### Хід роботи

1. Завантажити і встановити Java (JDK) за посиланням

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



Java SE Development Kit 13.0.2		
You must accept the <a href="#">Oracle Technology Network License Agreement</a> for Oracle Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux	155.72 MB	<a href="#">jdk-13.0.2_linux-x64_bin.deb</a>
Linux	162.66 MB	<a href="#">jdk-13.0.2_linux-x64_bin.rpm</a>
Linux	179.41 MB	<a href="#">jdk-13.0.2_linux-x64_bin.tar.gz</a>
macOS	173.3 MB	<a href="#">jdk-13.0.2_osx-x64_bin.dmg</a>
macOS	173.7 MB	<a href="#">jdk-13.0.2_osx-x64_bin.tar.gz</a>
Windows	159.83 MB	<a href="#">jdk-13.0.2_windows-x64_bin.exe</a>
Windows	178.99 MB	<a href="#">jdk-13.0.2_windows-x64_bin.zip</a>

Рис. 1.1. Вибір інсталяційного пакету JDK

2. Завантажити IDE NetBeans за посиланням та встановити

<http://netbeans.apache.org/download/index.html>



Рис. 1.2. Встановлення IDE NetBeans

3. Завантажити за посиланням та встановити IntelliJ IDEA EDU

<https://www.jetbrains.com/ru-ru/idea/download>



## Изучаете или преподаете программирование?

Попробуйте специальные образовательные возможности IntelliJ IDEA EDU.

Скачать

.exe ▼

Бесплатная, с открытым кодом

Рис. 1.3. Встановлення IntelliJ IDEA EDU

4. Запустити середовище NetBeans, створити java-проект

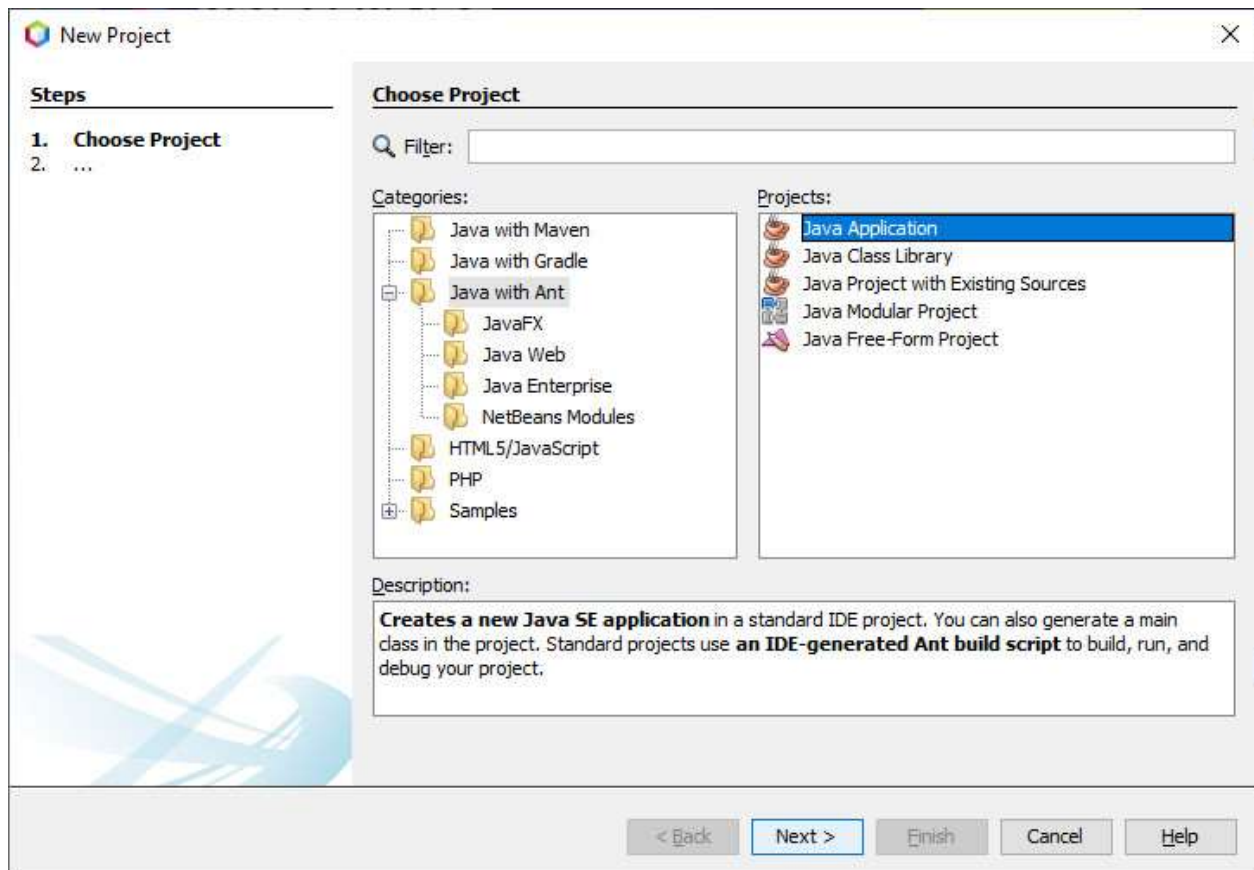


Рис. 1.4. Створення проекту в IDE NetBeans

## 5. Написати код програми

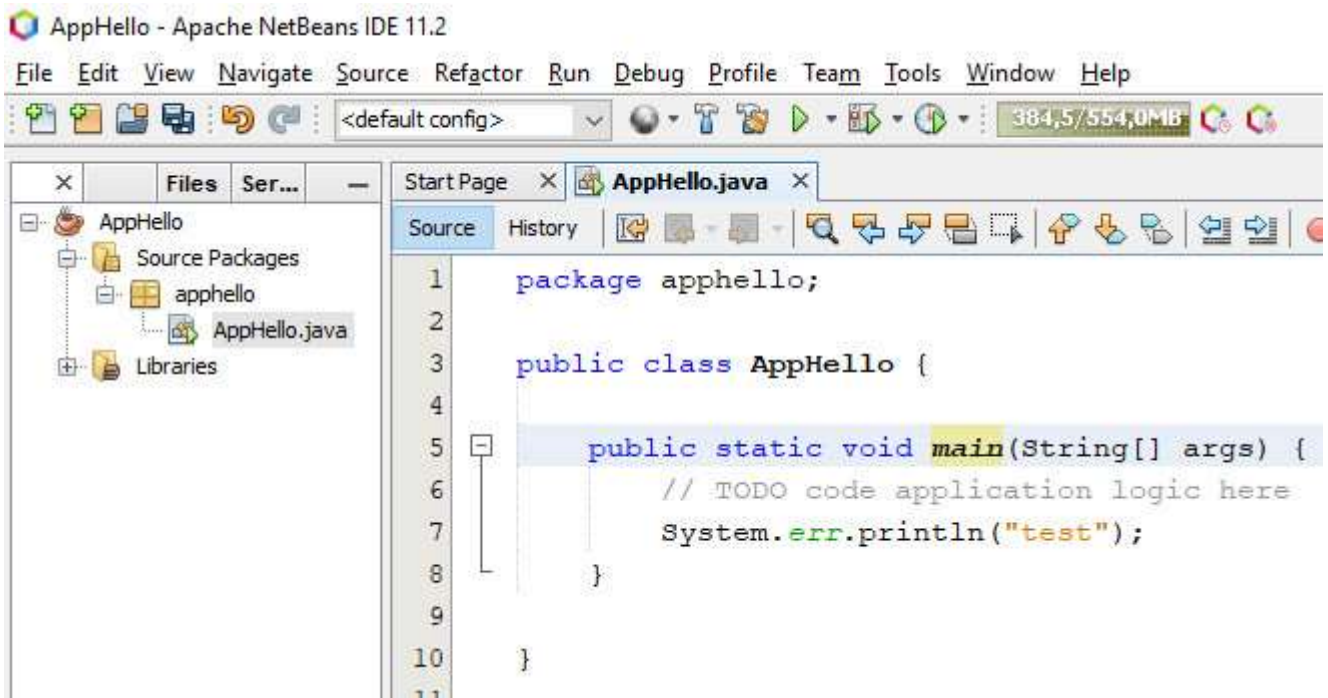


Рис. 1.5. Написання коду програми

6. Відкомпілювати проект, виконати відлагодження.
7. Ознайомитися з структурою проєкту
8. Відкомпілювати і запустити програму з командної строки.

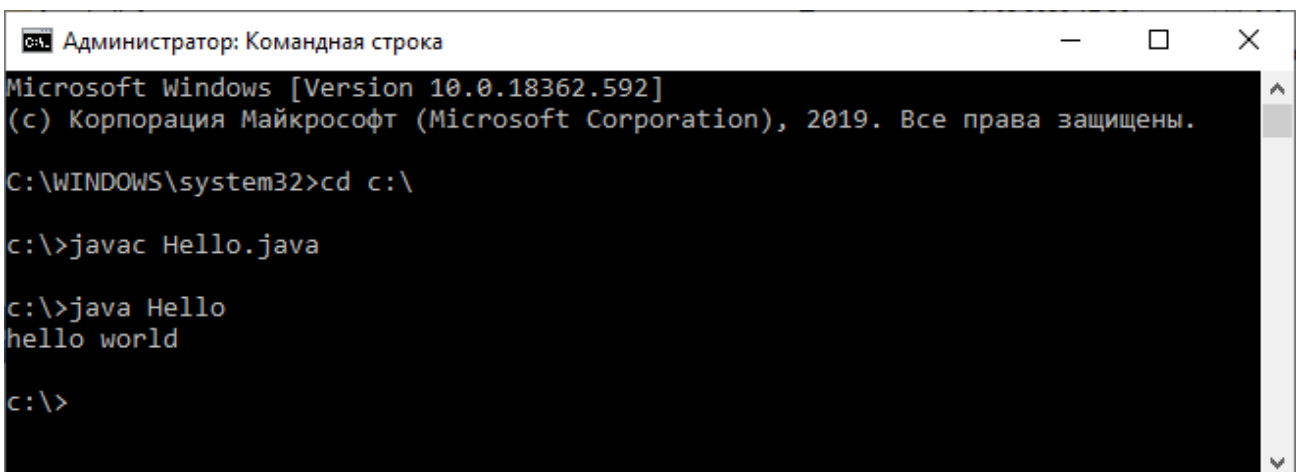


Рис. 1.6. Запуск програми з командної строки

## **Контрольні запитання**

1. Що включає в себе платформа JAVA
2. Як реалізовані компоненти платформи (бібліотеки)
3. Етапи запуску програми на JAVA
4. Основні переваги і недоліки віртуальної машини
5. Основні переваги IDE
6. Компіляція і запуск програм в консолі



## РОБОТА №2

### Типи даних, оператори, конструкції

**Мета роботи** – ознайомитися з основними типами даних, операціями і конструкціями мови, написати програму для виконання арифметичних операцій.

### Завдання

1. Написати програму «Калькулятор», яка виконує найпростіші арифметичні операції – додавання, віднімання, множення, порівняння.
2. Кожну арифметичну операцію оформити у вигляді окремого методу
3. Задати константи для кожної з арифметичних операцій.
4. Викликати відповідну функцію, використовуючи інструкцію `switch`, яка приймає на вхід змінну зі значенням арифметичної операції.

### Приклад виконання

```
package calculator;

import java.util.Scanner;

public class Calculator {

    //константи - перелік можливих операція
    final static int NOT_FOUND = 0;
    final static int ADDITION = 1;
    final static int SUBTRACTION = 2;
    final static int MULTIPLICATION = 3;
    final static int COMPARE = 4;

    int operandA;
    int operandB;
    int operationCode;
    int operationResult;

    public int sum(int a, int b){
        return a + b;
    }

    public int subtract(int a, int b){
        return a - b;
    }
}
```

```

    }
    public int multiply(int a, int b){
        int result = 0;
        for (int i = 0; i < b; i++) {
            result = result + a;
        }
        return result;
    }

    public int compare(int a, int b){
        return (a > b) ? a : b;
    }

    public void getOperationCode(String operationName){
        if(operationName.equalsIgnoreCase("+")){
            operationCode = ADDITION;
        }else if(operationName.equalsIgnoreCase("-")){
            operationCode = SUBTRACTION;
        } else if(operationName.equalsIgnoreCase(">")){
            operationCode = COMPARE;
        } else if(operationName.equalsIgnoreCase("*")){
            operationCode = MULTIPLICATION;
        } else {
            operationCode = NOT_FOUND;
        }
    }

    public void doOperation(){
        switch(operationCode){
            case ADDITION:
                operationResult = sum(operandA, operandB);
                break;

            case SUBTRACTION:
                operationResult = subtract(operandA, operandB);
                break;

            case MULTIPLICATION:
                operationResult = multiply(operandA, operandB);
                break;

            case COMPARE:
                operationResult = compare(operandA, operandB);
                break;
        }
    }
}

```

```

    public void inputArguments() {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter A: ");
        operandA = Integer.valueOf(in.nextLine());

        System.out.println("Enter B: ");
        operandB = Integer.valueOf(in.nextLine());
    }

    public void inputOperation() {
        System.out.println("operation: '+' '-' '>' '*' ');

        Scanner in = new Scanner(System.in);
        String userOperation = in.nextLine();
        getOperationCode(userOperation);
    }

    public void printResult() {
        System.out.println("Result = " + operationResult);
    }

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

        Calculator calc = new Calculator();
        calc.inputOperation();

        if(calc.operationCode == NOT_FOUND) {
            System.out.println("Error operation, sorry");
        } else {
            calc.inputArguments();
            calc.doOperation();
            calc.printResult();
        }
    }
}

```

## Контрольні запитання

1. Примітивні типи даних, значення за замовчуванням
2. Змінні – визначення, оголошення, класифікація, область видимості
3. Літерали
4. Оголошення і виклик функцій
5. Циклічні конструкції
6. Оператори розгалуження

## РОБОТА №3

### Основи роботи з об'єктами

**Мета роботи** – навчитися виконувати опис класів, створювати об'єкти, викликати методи об'єктів.

#### Завдання

1. Написати клас, який містить інформацію про країну з такими атрибутами: назва, столиця, населення, грошова одиниця.
2. Створити два об'єкти даного класу, ініціалізувати їх поля.
3. Вивести на екран країну, що має більше населення.
4. Створити третій об'єкт даного класу.
5. Виконати присвоєння об'єктів (першій країні присвоїти третю)
6. Змінити значення довільного поля в третьому об'єкті
7. Вивести значення полів першого об'єкту

#### Приклад виконання

```
public class Country {
    String title;
    String capital;
    String currency;
    int population;
}

public static void main(String[] args) {
    Country Ukraine = new Country();
    Ukraine.title = "Україна";
    Ukraine.capital = "Київ";
    Ukraine.currency = "UAH";
    Ukraine.population = 37000000;

    System.out.println(
        "країна: " + Ukraine.title + '\n' +
        "столиця: " + Ukraine.capital + '\n'
    );
}
```

### **Контрольні запитання**

1. Визначення класу та об'єкту – співвідношення між ними
2. Створення об'єкту
3. Поведінка і стан об'єкту
4. Принципи ООП
5. Переваги і недоліки ООП

## **РОБОТА №4**

### **Реалізація принципів ООП**

**Мета роботи** – навчитися виконувати інкапсуляцію та наслідування класів.

#### **Завдання**

1. Написати базовий клас «Людина», який містить інформацію про ім'я та прізвище. Клас повинен містити конструктор.
2. Створити клас «Студент», який наслідує базовий клас і містить інформацію про ЗНО, ФДП, АТЕСТАТ та РЕЙТИНГ студента.
3. В класі студент реалізувати два перевантажених конструктора – з ФДП та без ФДП
4. Створити декілька об'єктів класу студент.
5. Розрахувати і вивести рейтинг кожного студента

#### **Контрольні запитання**

1. Інкапсуляція – визначення та переваги
2. Наслідування, синтаксис, переваги
3. Виклик конструктора базового класу
4. Ключове слово *this*
5. Перевантаження методів

## РОБОТА №5

### Робота з масивами об'єктів

**Мета роботи** – навчитися працювати з масивом об'єктів та з статичними членами класів.

### Завдання

1. Створити клас «Посилка», який містить наступні атрибути: ширину, довжину, висоту, ідентифікатор. Клас також містить статичне поле, для підрахунку кількості посилки.
2. Створити масив об'єктів типу «Посилка». Можна даний масив оформити у вигляді окремого класу.
3. Створити декілька об'єктів класу «Посилка», і записати їх в масив.
4. Знайти посилку найменшого об'єму
5. Перевірити кількість існуючих посилки. Якщо це значення перевищує якесь граничне значення – вивести повідомлення

### Приклад виконання

```
public class Box {  
    private int id;  
  
    private double width;  
    private double height;  
    private double depth;  
  
    private static int numberOfBoxes = 0;  
  
    public Box() {  
        width = 10;  
        height = 10;  
        depth = 10;  
        id = ++numberOfBoxes;  
    }  
}
```

```

    public Box(double w, double h, double d){
        width = w;
        height = h;
        depth = d;
        id = ++numberOfBoxes;
    }

    public double getVolume(){
        return depth * height * width;
    }

    public int getId() {
        return id;
    }
}

public class Boxes {
    private Box[] boxes;

    public void generateBoxes(){
        Box boxRed = new Box(0.1, 0.2, 300);
        Box boxBlue = new Box(10, 20, 3);
        Box boxPink = new Box();

        boxes = new Box[3];
        boxes[0] = boxRed;
        boxes[1] = boxBlue;
        boxes[2] = boxPink;
    }

    public Box getSmallestBox(){
        Box min = boxes[0];
        for(Box item : boxes){
            if(item.getVolume() < min.getVolume()){
                min = item;
            }
        }
        return min;
    }
}

```



```
public class BoxesDemo {  
    public static void main(String[] args) {  
        Boxes boxes = new Boxes();  
  
        boxes.generateBoxes();  
        Box smallestBox = boxes.getSmallestBox();  
  
        System.out.println("The smallest box is: " +  
            smallestBox.getId());  
    }  
}
```

### **Контрольні запитання**

1. Правила оголошення масивів
2. Створення масивів об'єктів
3. Формат циклу `for` для обходу масиву
4. Статичні члени класу, оголошення, звернення
5. Статичні методи класу, призначення

## **РОБОТА №6**

### **Вкладені класи**

**Мета роботи** – навчитися оголошувати і використовувати вкладені класи.

### **Завдання**

1. Написати клас «Студент», який містить наступні поля: прізвище, середній бал, кількість пропусків, вік, стать, ознаку отримання стипендії та ознаку необхідності військового обліку.
2. В класі «Студент» написати вкладений клас для обчислення стипендії. Стипендія дається студентам в яких середній бал більше 4,5 та кількість пропусків менша 10.
3. В класі «Студент» написати локальний клас для визначення військовозобов'язаних студентів. Використати його в конструкторі. Військовозобов'язаними вважати чоловіків старших 18 років.
4. Створити декілька об'єктів класу студент.
5. Розрахувати і вивести характеристики кожного студента

### **Контрольні запитання**

1. Класифікація вкладених класів
2. Причини використання вкладених класів
3. Оголошення внутрішніх (inner) та локальних вкладених класів
4. Створення об'єктів вкладених класів
5. Доступ до членів зовнішнього класу

## РОБОТА №7

### Поліморфізм

**Мета роботи** – навчитися використовувати абстрактні класи для реалізації поліморфізму.

#### Завдання

1. Написати абстрактний клас «Документ», який містить поля «назва», «автор», конструктор і абстрактний метод для друку цих полів.
2. Створити дочірні класи «Книга» та «Стаття», з власною реалізацією методу для друку.
3. Створити масив об'єктів класу «Документ», який містить декілька статей та книг.
4. В циклі по масиву вивести інформацію про видання.
5. Створити ієрархію класів робітників компанії з погодинною оплатою праці та фіксованою ставкою з премією. В кожному з класів реалізувати метод нарахування зарплати. Створити декілька об'єктів цих класів, в циклі розрахувати заробітну плату для всіх працівників компанії

#### Приклад виконання

Абстрактний клас робітник

```
public abstract class Employee {  
    private int empCode;  
    private String firstName;  
    private String lastName;  
  
    public Employee(int empCode, String firstName, String lastName) {  
        this.empCode = empCode;  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
}
```

```

public int getEmpCode() {
    return empCode;
}

public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}

public void setEmpCode(int empCode) {
    this.empCode = empCode;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

@Override
public String toString() {
    return "Employee{" + "empCode=" + empCode
        + ", firstName=" + firstName + ", lastName=" + lastName + '}';
}

public abstract double pay();
}

```

Клас робітник з погодинною оплатою праці

```
public class HourlyEmployee extends Employee{
    private double rate;
    private double workingHours;

    /**
     *
     * @param rate
     * @param workingHours
     * @param empCode
     * @param firstName
     * @param LastName
     */
    public HourlyEmployee(double rate, double workingHours, int empCode,
        String firstName, String lastName) {

        super(empCode, firstName, lastName);
        setRate(rate);
        setWorkingHours(workingHours);
    }

    public double getRate() {
        return rate;
    }

    public double getWorkingHours() {
        return workingHours;
    }

    public void setRate(double rate) {
        this.rate = (rate < 0.0) ? 0.0 : rate;
    }

    public void setWorkingHours(double workingHours) {
        this.workingHours = (workingHours < 0.0) ? 0.0 : workingHours;
    }

    @Override
    public double pay() {
        return getRate() * getWorkingHours();
    }

}
```

Клас робітник з фіксованим окладом

```
public class SalariedEmployee extends Employee{

    private double basic;
    private double grossSale;
    private double comission;

    /**
     * @param basic
     * @param grossSale
     * @param comission
     * @param empCode
     * @param firstName
     * @param lastName
     */
    public SalariedEmployee(double basic, double grossSale, double comission,
        int empCode, String firstName, String lastName) {

        super(empCode, firstName, lastName);
        this.basic = basic;
        this.grossSale = grossSale;
        this.comission = comission;
    }

    public double getBasic() {
        return basic;
    }

    public double getGrossSale() {
        return grossSale;
    }

    public double getComission() {
        return comission;
    }

    public void setBasic(double basic) {
        this.basic = (basic < 0.0) ? 0.0 : basic;
    }

    public void setGrossSale(double grossSale) {
        this.grossSale = (grossSale < 0.0) ? 0.0 : grossSale;
    }

    public void setComission(double comission) {
        this.comission = (comission > 0.0) && (comission < 1.0)
            ? comission : 0.0;
    }

    @Override
    public double pay() {
        return basic + comission * grossSale;
    }
}
```

## Клас робітники (організація)

```
public class Employees {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        Employee[] employees = new Employee[4];  
        employees[0] = new HourlyEmployee(100, 120.25, 1, "Andriy", "One");  
        employees[1] = new HourlyEmployee(110, 130.50, 2, "Semen", "Two");  
        employees[2] = new SalariedEmployee(2000, 531, 0.2, 33, "Alex", "Tee");  
        employees[3] = new SalariedEmployee(2000, 1531, 0.2, 13, "Yan", "Six");  
  
        for (Employee employee : employees) {  
            System.out.println(employee.toString());  
            System.out.printf("Pay: %2f\n", employee.pay());  
            System.out.println("-----");  
        }  
    }  
}
```

## Контрольні запитання

1. Визначення та переваги поліморфізму
2. Визначення абстрактного класу
3. Реалізація поліморфізму через батьківський клас
4. Реалізація поліморфізму через інтерфейс
5. Різниця між overload та override методами.

## РОБОТА №8

### Exceptions

**Мета роботи** – навчитися оброблювати і генерувати виняткові ситуації.

### Завдання

Написати клас з методами, які генерують наступні виняткові ситуації

- ділення на нуль (без обробки)
- ділення на нуль (з блоком *try ... catch*).
- з декількома блоками *catch*.
- з виведенням опису помилки та стеку викликів
- методи, які повертають помилку *throwable*
- в методі, описаному як *throwable* згенерувати помилку і обробити її в головній програмі

Написати користувацький клас виключення, згенерувати і перехопити помилку даного класу.

### Контрольні запитання

1. Визначення виняткових ситуацій
2. Ієрархія класів помилок
3. Механізм захисту коду і обробки виняткових ситуацій
4. Власне генерування помилок
5. Блок *finally* - застосування



## **РОБОТА №9**

### **Інтерфейси**

**Мета роботи** – навчитися працювати з інтерфейсами, реалізовувати поліморфізм на основі інтерфейсів.

### **Завдання**

1. Написати інтерфейс касового апарату, який реалізує наступні операції:
  - а) початок зміни (введення логіну оператора)
  - б) друк чека (виводить назву товару, ціну і кількість)
  - в) звіт (загальна сума продажів) за день
2. Написати 2 класи, які реалізують цей інтерфейс: Samsung, RF
3. Написати клас ARM (автоматизоване робоче місце касира), для продажу товарів, яка реалізує наступні функції
  - а) початок зміни оператора
  - б) підключення касового апарату
  - в) продаж товару (збереження товару і друк чеку)
  - г) звіт за день
4. Промодельовати роботу каси, по черзі підключити два касових апарати, продати 2-3 товари, вивести суму за день.

### **Контрольні запитання**

1. Поняття інтерфейсу, що може містити інтерфейс
2. Розширення інтерфейсу
3. Реалізація поліморфізму через інтерфейси
4. Правило застосування інтерфейсів чи абстрактних класів

## РОБОТА №10

### Узагальнені типи та колекції

**Мета роботи** – навчитися працювати з групами об'єктів, використовуючи колекції.

### Завдання

1. Написати клас «Студент», який містить ім'я, прізвище та рейтинг студента.
2. Написати клас «Деканат», який містить список студентів і реалізований як колекція. В класі «Деканат» реалізувати метод додавання, пошуку, видалення, студента та виведення інформації по студенту.
3. Реалізувати сортування студентів двома способами:
  - за допомогою компараторів (за рейтингом)
  - за допомогою інтерфейсу (Comparable) – за рейтингом та прізвищем
4. В коді програми зробити обхід колекції трьома способами:
  - за допомогою Aggregate Operations
  - за допомогою Iterators
  - циклом for

### Приклад виконання

Класи які використовуються в проекті

1. Student – містить інформацію про студента
2. StudentsRatingComparator – компаратор для сортування студентів
3. CollectionSort – клас, який працює зі студентами

```

package collectionsort;

import java.util.Objects;

public class Student {
    private String lastName;
    private float rating;

    public Student(String lastName, float rating) {
        this.lastName = lastName;
        this.rating = rating;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public void setRating(float rating) {
        this.rating = rating;
    }

    public String getLastName() {
        return lastName;
    }

    public float getRating() {
        return rating;
    }

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 41 * hash + Objects.hashCode(this.lastName);
        hash = 41 * hash + Float.floatToIntBits(this.rating);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        ...
    }
}

```

```

package collectionsort;

import java.util.Comparator;

public class StudentsRatingComparator implements Comparator<Student>{

    @Override
    public int compare(Student s1, Student s2) {
        float ratingS1 = s1.getRating();
        float ratingS2 = s2.getRating();

        if(ratingS1 > ratingS2){
            return -1;
        } else if (ratingS1 < ratingS2){
            return 1;
        } else {
            return 0;
        }
    }
}

```

```

package collectionsort;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

public class CollectionSort {

    private static void printCollection(String title, List<Student> list) {
        System.out.println(title);

        for(Student st : list) {
            System.out.format("%15s %f", st.getLastName(), st.getRating());
        }
        System.out.println();

        list.stream().forEach((st) -> {
            System.out.format("%15s %f", st.getLastName(), st.getRating());
        });
        System.out.println();

        for(Iterator<Student> it = list.iterator(); it.hasNext();) {
            Student st = it.next();
            System.out.format("%15s %f", st.getLastName(), st.getRating());
        }
        System.out.println();
    }
}

```

```

public static void main(String[] args) {

    List<Student> la61 = new ArrayList<Student>();
    la61.add(new Student("ivanov", 4));
    la61.add(new Student("petrov", 4));
    la61.add(new Student("sidorov", 3));
    la61.add(new Student("fedorov", 5));

    // Сортировка с классом Comparator
    Collections.sort(la61, new StudentsRatingComparator());
    printCollection("После сортировки", la61);
}
}

```

## Контрольні запитання

1. Основні складові Collection Framework
2. Ієрархія інтерфейсів колекцій
3. Способи обходу елементів колекцій
4. Операції над усіма елементами колекції
5. Сортування елементів колекцій