Звіт

**Лабораторна робота №1**

**Тема:** Знайомство з мовою програмування Java. Написання простих програм на мові програмування Java

**Мета роботи:** встановити IDE Intellij IDEA; створити репозиторій на GitLab; вивчити реалізацію базових алгоритмічних конструкцій у мові програмування Java; знайомство з правилами оформлення програмного коду.

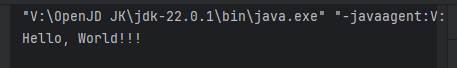
**Завдання 4.** Написання простих програм:

***Програма 1***

Ім'я класу: com.education.ztu.Task1

Напишіть клас, який реалізує функціональність відображення рядка «Hello, World!!!» у консолі.

package com.education.ztu;  
  
public class Task1 {  
 public static void main(String[] args) {  
 System.*out*.println("Hello, World!!!");  
 }  
}

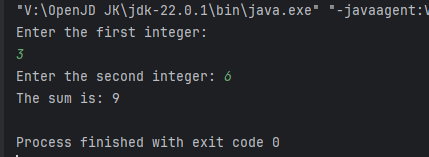


***Програма 2***

Ім'я класу: com.education.ztu.Task2

Напишіть клас, який реалізує функціональність додавання двох цілих чисел. Для зчитування даних використовувати методи класу Scanner.

package com.education.ztu;  
  
import java.util.Scanner;  
  
public class Task2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Enter the first integer: ");  
 int a = scanner.nextInt();  
 System.*out*.print("Enter the second integer: ");  
 int b = scanner.nextInt();  
  
 System.*out*.println("The sum is: " + (a + b));  
 scanner.close();  
 }  
}



***Програма 3***

Ім'я класу: com.education.ztu.Task3

Напишіть клас, який реалізує функціональність відображення параметрів командного рядка в консолі (відображення через пробіл між ними), результат не повинен закінчуватися пробілом. Аргументи передавати таким чином Task3.main(new String[]{"2", "3", "5", "8"}); в класі Main.

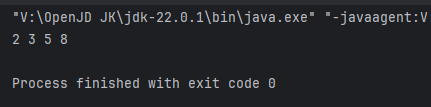
Task3

package com.education.ztu;  
  
public class Task3 {  
 public static void main(String[] args) {  
 if (args != null && args.length > 0) {  
 String result = String.*join*(" ", args);  
 System.*out*.println(result);  
 } else {  
 System.*out*.println("No arguments provided");  
 }  
 }  
}

Main

package com.education.ztu;  
  
public class Main {  
 public static void main(String[] args) {  
  
 Task3.*main*(new String[] { "2", "3", "5", "8" });  
 }  
}

Результат



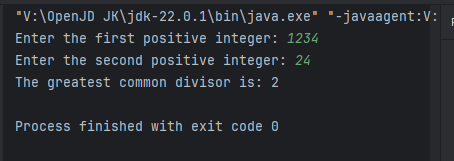
***Програма 4***

Ім'я класу: com.education.ztu.Task4

Напишіть клас, який реалізує функціональні можливості визначення найбільшого спільного дільника двох цілих додатних чисел. Для зчитування даних використовувати методи класу Scanner.

package com.education.ztu;  
  
import java.util.Scanner;  
  
public class Task4 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the first positive integer: ");  
 int a = scanner.nextInt();  
 System.*out*.print("Enter the second positive integer: ");  
 int b = scanner.nextInt();  
  
 System.*out*.println("The greatest common divisor is: " + *gcd*(a, b));  
 scanner.close();  
 }  
  
 private static int gcd(int a, int b) {  
 while (b != 0) {  
 int temp = b;  
 b = a % b;  
 a = temp;  
 }  
 return a;  
 }  
}

Результат:



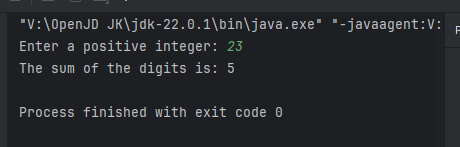
***Програма 5***

Ім'я класу: com.education.ztu.Task5

Напишіть клас, який реалізує функціональні можливості визначення суми цифр цілого позитивного числа. Для зчитування даних використовувати методи класу Scanner.

package com.education.ztu;  
  
import java.util.Scanner;  
  
public class Task5 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter a positive integer: ");  
 int num = scanner.nextInt();  
 System.*out*.println("The sum of the digits is: " + *sumOfDigits*(num));  
 scanner.close();  
 }  
  
 private static int sumOfDigits(int num) {  
 int sum = 0;  
 while (num > 0) {  
 sum += num % 10;  
 num /= 10;  
 }  
 return sum;  
 }  
}

Результат:



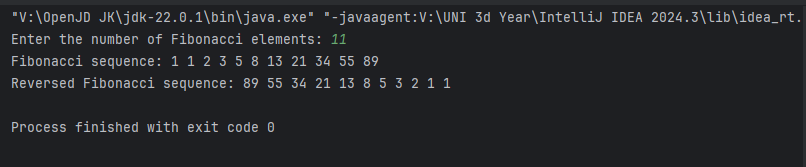
***Програма 6***

Ім'я класу: com.education.ztu.Task6

Напишіть клас, який створює масив із n елементів і заповнює його зростаючою послідовністю чисел Фібоначчі (1,1,2,3,5,8…). Створити новий масив та заповнити його зворотньою послідовністю Фібоначчі. Вивести в консоль обидва масиви. Для зчитування даних використовувати методи класу Scanner.

package com.education.ztu;  
  
import java.util.Scanner;  
  
public class Task6 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the number of Fibonacci elements: ");  
 int n = scanner.nextInt();  
  
 int[] fibonacci = new int[n];  
 int[] reversedFibonacci = new int[n];  
  
 if (n > 0)  
 fibonacci[0] = 1;  
 if (n > 1)  
 fibonacci[1] = 1;  
  
 for (int i = 2; i < n; i++) {  
 fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];  
 }  
  
 for (int i = 0; i < n; i++) {  
 reversedFibonacci[i] = fibonacci[n - 1 - i];  
 }  
  
 System.*out*.println("Fibonacci sequence: " + *arrayToString*(fibonacci));  
 System.*out*.println("Reversed Fibonacci sequence: " + *arrayToString*(reversedFibonacci));  
 scanner.close();  
 }  
  
 private static String arrayToString(int[] array) {  
 StringBuilder sb = new StringBuilder();  
 for (int num : array) {  
 sb.append(num).append(" ");  
 }  
 return sb.toString().trim();  
 }  
  
}

Результат:



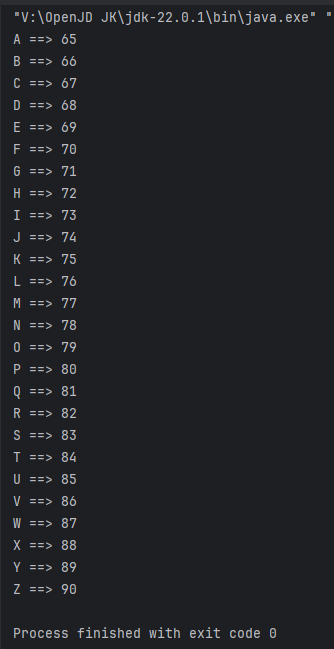
***Програма 7***

Ім'я класу: com.education.ztu.Task7

Створити масив символів латинського алфавіту та вести їх числові коди в такому форматі: A ==> 65 B ==> 66 C ==> 67

package com.education.ztu;  
  
public class Task7 {  
 public static void main(String[] args) {  
 char[] alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();  
  
 for (char letter : alphabet) {  
 System.*out*.println(letter + " ==> " + (int) letter);  
 }  
 }  
}

Результат:

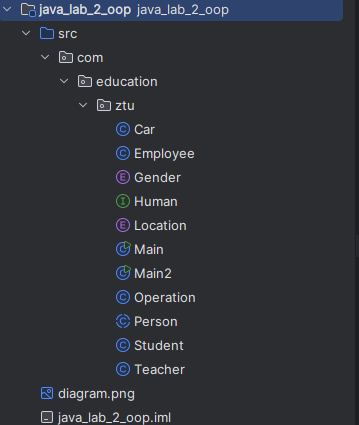


**Лабораторна робота №2**

**Тема:** Створення структури класу заданої предметної області.

**Мета роботи:** створити ієрархію класів заданої предметної області, робота з статичними методами. Завдання на лабораторну роботу

***Завдання 1***. Створити консольний Java проект java\_lab\_2\_oop з пакетом com.education.ztu



***Завдання 2.*** Створити ієрархію класів відповідно до UML діаграми:

- поля класів повинні бути приховані модифікаторами доступу private, protected;

- створити конструктор без аргументів та з агрументами;

- створити блок ініціалізації, в якому ініціалізуються значення полів за замовчуванням у разі, якщо викликається конструктор без аргументів;

- створити геттери та сеттери для полів;

- створити статичну змінну counter для підрахунку створених екземплярів даного класу та статичний метод showCounter для відображення значення змінної counter. - створити enam Location та Gender і використати їх в полях класів.

- створити інтерфейс Human з методами sayFullName, sayAge, sayLocation, sayGender та whoIAm (default)

- створити абстрактний клас Person з абстрактним методом getOccupation та звичайним методом getFullInfo, що імплементує Human;

- створити класу Student, Teacher, Employee, що наслідують Person та перевизначити необхідні методи та створити свої.

- для Teacher, Employee додати поле Car , що є об'єктом відповідного класу.

- створити в Car внутрішній клас Engine з методами startEngine, stopEngine, isEngineWorks та реалізувати їх логіку.

- додати до описаної функціональності свою (нові поля та методи).

- в методі main класу Main створити об'єкти відповідних класів та продемонструвати роботу їх методів.

- продемонтрувати роботу оператору instanceof.

Лістинг програми:

**ENAM:**

**Location**

package com.education.ztu;  
  
public enum Location {  
 *KIEV*, *ZHYTOMYR*, *VINNYTSYA*, *RIVNE*;  
}

**Gender**

package com.education.ztu;  
  
public enum Gender {  
 *MALE*, *FEMALE*;  
}

**INTERFACE:**

**Human**

package com.education.ztu;  
  
interface Human {  
 void sayFullName();  
 void sayAge();  
 void sayLocation();  
 void sayGender();  
 default void whoIAm() {  
 System.*out*.println("I am a human");  
 }  
}

**CLASSES:**

Person

package com.education.ztu;  
  
abstract class Person implements Human {  
 private static int *counter*;  
 private String firstname;  
 private String lastname;  
 private int age;  
 private Gender gender;  
 private Location location;  
  
 {  
 // Initialization block  
 this.firstname = "Unknown";  
 this.lastname = "Unknown";  
 this.age = 0;  
 this.gender = Gender.*MALE*;  
 this.location = Location.*KIEV*;  
 }  
  
 Person() {  
 *counter*++;  
 }  
  
 Person(String firstname, String lastname, int age, Gender gender, Location location) {  
 this();  
 this.firstname = firstname;  
 this.lastname = lastname;  
 this.age = age;  
 this.gender = gender;  
 this.location = location;  
 }  
  
 public static int showCounter() {  
 return *counter*;  
 }  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public Gender getGender() {  
 return gender;  
 }  
  
 public void setGender(Gender gender) {  
 this.gender = gender;  
 }  
  
 public Location getLocation() {  
 return location;  
 }  
  
 public void setLocation(Location location) {  
 this.location = location;  
 }  
  
 public String getFullInfo() {  
 return firstname + " " + lastname + ", Age: " + age + ", Gender: " + gender + ", Location: " + location;  
 }  
  
 @Override  
 public void sayFullName() {  
 System.*out*.println("Full Name: " + firstname + " " + lastname);  
 }  
  
 @Override  
 public void sayAge() {  
 System.*out*.println("Age: " + age);  
 }  
  
 @Override  
 public void sayLocation() {  
 System.*out*.println("Location: " + location);  
 }  
  
 @Override  
 public void sayGender() {  
 System.*out*.println("Gender: " + gender);  
 }  
  
 public abstract void getOccupation();  
}

Student

package com.education.ztu;  
  
class Student extends Person {  
 private static int *counter*;  
 private int course;  
 private String speciality;  
 private String university;  
  
 Student() {  
 super();  
 *counter*++;  
 }  
  
 Student(String firstname, String lastname, int age, Gender gender, Location location, int course, String speciality, String university) {  
 super(firstname, lastname, age, gender, location);  
 this.course = course;  
 this.speciality = speciality;  
 this.university = university;  
 *counter*++;  
 }  
  
 public static int showCounter() {  
 return *counter*;  
 }  
  
 public int getCourse() {  
 return course;  
 }  
  
 public void setCourse(int course) {  
 this.course = course;  
 }  
  
 public String getSpeciality() {  
 return speciality;  
 }  
  
 public void setSpeciality(String speciality) {  
 this.speciality = speciality;  
 }  
  
 public String getUniversity() {  
 return university;  
 }  
  
 public void setUniversity(String university) {  
 this.university = university;  
 }  
  
 @Override  
 public void getOccupation() {  
 System.*out*.println("Occupation: Student");  
 }  
}

Teacher

package com.education.ztu;  
  
class Teacher extends Person {  
 private static int *counter*;  
 private String subject;  
 private String university;  
 private Car car;  
  
 Teacher() {  
 super();  
 *counter*++;  
 }  
  
 Teacher(String firstname, String lastname, int age, Gender gender, Location location, String subject, String university, Car car) {  
 super(firstname, lastname, age, gender, location);  
 this.subject = subject;  
 this.university = university;  
 this.car = car;  
 *counter*++;  
 }  
  
 public static int showCounter() {  
 return *counter*;  
 }  
  
 public String getSubject() {  
 return subject;  
 }  
  
 public void setSubject(String subject) {  
 this.subject = subject;  
 }  
  
 public String getUniversity() {  
 return university;  
 }  
  
 public void setUniversity(String university) {  
 this.university = university;  
 }  
  
 public Car getCar() {  
 return car;  
 }  
  
 public void setCar(Car car) {  
 this.car = car;  
 }  
  
 @Override  
 public void getOccupation() {  
 System.*out*.println("Occupation: Teacher");  
 }  
}

Employee

package com.education.ztu;  
  
class Employee extends Person {  
 private static int *counter*;  
 private String company;  
 private String position;  
 private Car car;  
  
 Employee() {  
 super();  
 *counter*++;  
 }  
  
 Employee(String firstname, String lastname, int age, Gender gender, Location location, String company, String position, Car car) {  
 super(firstname, lastname, age, gender, location);  
 this.company = company;  
 this.position = position;  
 this.car = car;  
 *counter*++;  
 }  
  
 public static int showCounter() {  
 return *counter*;  
 }  
  
 public String getCompany() {  
 return company;  
 }  
  
 public void setCompany(String company) {  
 this.company = company;  
 }  
  
 public String getPosition() {  
 return position;  
 }  
  
 public void setPosition(String position) {  
 this.position = position;  
 }  
  
 public Car getCar() {  
 return car;  
 }  
  
 public void setCar(Car car) {  
 this.car = car;  
 }  
  
 @Override  
 public void getOccupation() {  
 System.*out*.println("Occupation: Employee");  
 }  
}

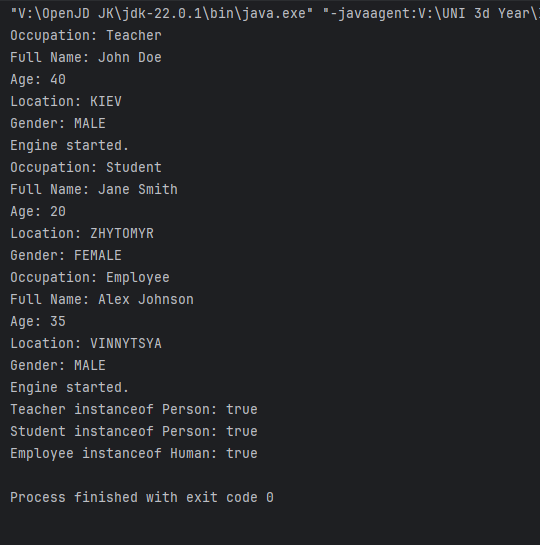
Car та внутрішній клас Engine

package com.education.ztu;  
  
class Car {  
 private String brand;  
 private Engine engine;  
  
 Car(String brand) {  
 this.brand = brand;  
 this.engine = new Engine();  
 }  
  
 public String getBrand() {  
 return brand;  
 }  
  
 public void setBrand(String brand) {  
 this.brand = brand;  
 }  
  
 public Engine getEngine() {  
 return engine;  
 }  
  
 class Engine {  
 private boolean engineWorks;  
  
 Engine() {  
 this.engineWorks = false;  
 }  
  
 void startEngine() {  
 engineWorks = true;  
 System.*out*.println("Engine started.");  
 }  
  
 void stopEngine() {  
 engineWorks = false;  
 System.*out*.println("Engine stopped.");  
 }  
  
 boolean isEngineWorks() {  
 return engineWorks;  
 }  
 }  
}

Main

package com.education.ztu;  
  
public class Main {  
 public static void main(String[] args) {  
 Teacher teacher = new Teacher("John", "Doe", 40, Gender.*MALE*, Location.*KIEV*, "Math", "University A", new Car("Toyota"));  
 teacher.getOccupation();  
 teacher.sayFullName();  
 teacher.sayAge();  
 teacher.sayLocation();  
 teacher.sayGender();  
 teacher.getCar().getEngine().startEngine();  
  
 Student student = new Student("Jane", "Smith", 20, Gender.*FEMALE*, Location.*ZHYTOMYR*, 3, "Computer Science", "University B");  
 student.getOccupation();  
 student.sayFullName();  
 student.sayAge();  
 student.sayLocation();  
 student.sayGender();  
  
 Employee employee = new Employee("Alex", "Johnson", 35, Gender.*MALE*, Location.*VINNYTSYA*, "Tech Corp", "Developer", new Car("Honda"));  
 employee.getOccupation();  
 employee.sayFullName();  
 employee.sayAge();  
 employee.sayLocation();  
 employee.sayGender();  
 employee.getCar().getEngine().startEngine();  
  
 System.*out*.println("Teacher instanceof Person: " + (teacher instanceof Person));  
 System.*out*.println("Student instanceof Person: " + (student instanceof Person));  
 System.*out*.println("Employee instanceof Human: " + (employee instanceof Human));  
 }  
}

Результат виконання програми:



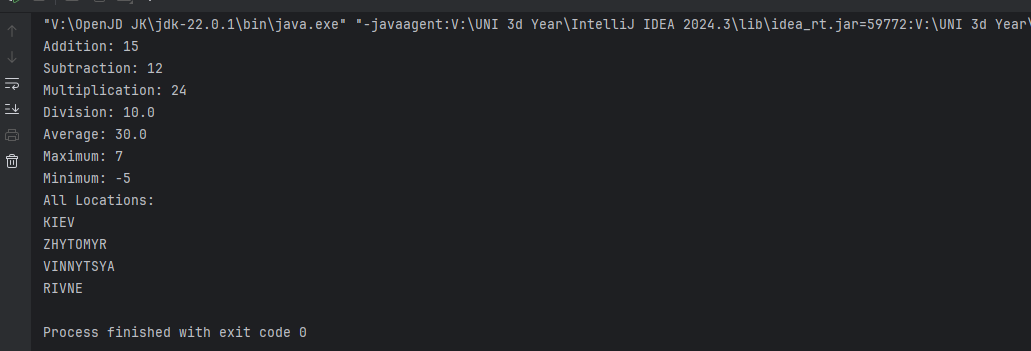
***Завдання 3.*** Створити клас Operation з статичними методами addition, subtraction, multiplication, division, average, maximum, minimum, що приймають необмежену кількість аргументів через varargs. в методі main класу Main2 продемонструвати роботу методів класу Operation - вивести всі значення enam Location.

package com.education.ztu;  
import java.util.Arrays;  
  
class Operation {  
 public static int addition(int... values) {  
 int sum = 0;  
 for (int value : values) {  
 sum += value;  
 }  
 return sum;  
 }  
  
 public static int subtraction(int... values) {  
 int result = values[0];  
 for (int i = 1; i < values.length; i++) {  
 result -= values[i];  
 }  
 return result;  
 }  
  
 public static int multiplication(int... values) {  
 int result = 1;  
 for (int value : values) {  
 result \*= value;  
 }  
 return result;  
 }  
  
 public static double division(int... values) {  
 double result = values[0];  
 for (int i = 1; i < values.length; i++) {  
 if (values[i] != 0) {  
 result /= values[i];  
 } else {  
 throw new ArithmeticException("Division by zero");  
 }  
 }  
 return result;  
 }  
  
 public static double average(int... values) {  
 int sum = *addition*(values);  
 return (double) sum / values.length;  
 }  
  
 public static int maximum(int... values) {  
 int max = values[0];  
 for (int value : values) {  
 if (value > max) {  
 max = value;  
 }  
 }  
 return max;  
 }  
  
 public static int minimum(int... values) {  
 int min = values[0];  
 for (int value : values) {  
 if (value < min) {  
 min = value;  
 }  
 }  
 return min;  
 }  
}

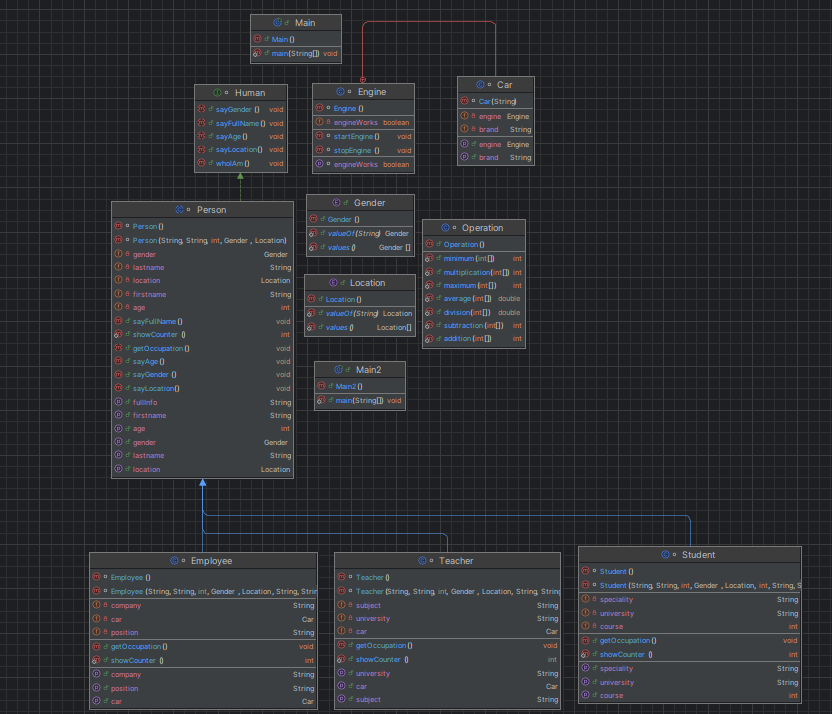
Main2

package com.education.ztu;  
  
public class Main2 {  
 public static void main(String[] args) {  
 System.*out*.println("Addition: " + Operation.*addition*(1, 2, 3, 4, 5));  
 System.*out*.println("Subtraction: " + Operation.*subtraction*(20, 5, 3));  
 System.*out*.println("Multiplication: " + Operation.*multiplication*(2, 3, 4));  
 System.*out*.println("Division: " + Operation.*division*(100, 2, 5));  
 System.*out*.println("Average: " + Operation.*average*(10, 20, 30, 40, 50));  
 System.*out*.println("Maximum: " + Operation.*maximum*(1, 3, 7, 0, -5));  
 System.*out*.println("Minimum: " + Operation.*minimum*(1, 3, 7, 0, -5));  
  
 System.*out*.println("All Locations:");  
 for (Location loc : Location.*values*()) {  
 System.*out*.println(loc);  
 }  
 }  
}

Результат:



***Завдання 4.*** Створити UML діаграму створеної структури ієрархії класів та зберегти як картинку.



**Лабораторна робота №3**

**Тема:** Використання узагальнень (generics). Клонування та порівняння об'єктів. **Мета роботи:** створити міні проект Game з використанням узагальнень, клонування та порівняння об'єктів.

Завдання на лабораторну роботу

***Завдання 1.***

Відкрити заготовлений проект з реалізованою базовою функціональністю.

***Завдання 2.***

За допомогою узагальнень (generics) встановити такі обмеження:

- до команди можна додавати тільки учасників, що відносяться до одної ліги (Schoolar, Student або Employee). –

- грати між собою можуть тільки команди з учасниками одної ліги (тобто команда студентів може грати тільки іншою командою студентів).

- продемонструвати створення команд, гравців, додавання гравців до команд, гри між ними.

Team.java

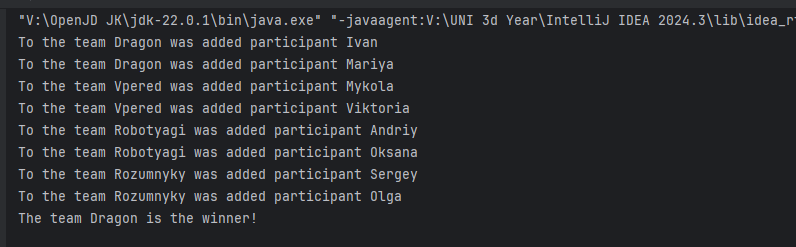
package com.education.ztu.game;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class Team<T extends Participant> {  
 private String name;  
 private List<T> participants = new ArrayList<>();  
  
 public Team(String name) {  
 this.name = name;  
 }  
  
 public void addNewParticipant(T participant) {  
 participants.add(participant);  
 System.*out*.println("To the team " + name + " was added participant " + participant.getName());  
 }  
  
 public void playWith(Team<T> team) {  
 String winnerName;  
 Random random = new Random();  
 int i = random.nextInt(2);  
 if (i == 0) {  
 winnerName = this.name;  
 } else {  
 winnerName = team.name;  
 }  
 System.*out*.println("The team " + winnerName + " is the winner!");  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public List<T> getParticipants() {  
 return participants;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setParticipants(List<T> participants) {  
 this.participants = participants;  
 }  
}

* Визначили тип T, який представляє тип учасників команди
* Обмеження T extends Participant гарантує, що тип T повинен бути підкласом Participant
* Це дозволяє командам приймати лише тих учасників, які наслідують клас Participant і не дозволить додавати об'єкти, які не є учасниками, наприклад, рядок "hello"
* Видалено перевірку instanceof, оскільки generis вже гарантує що параметр правильного типу
* Метод playWith тепер теж має тип Т, що гарнтує грати між собою тільки команди з учасниками одного типу

Game.java

package com.education.ztu.game;  
  
public class Game {  
 public static void main(String[] args) {  
 Schoolar schoolar1 = new Schoolar("Ivan", 13);  
 Schoolar schoolar2 = new Schoolar("Mariya", 15);  
 Student student1 = new Student("Mykola", 20);  
 Student student2 = new Student("Viktoria", 21);  
 Employee employee1 = new Employee("Andriy", 28);  
 Employee employee2 = new Employee("Oksana", 25);  
  
 Team<Schoolar> scollarTeam = new Team<>("Dragon");  
 scollarTeam.addNewParticipant(schoolar1);  
 scollarTeam.addNewParticipant(schoolar2);  
  
 Team<Student> studentTeam = new Team<>("Vpered");  
 studentTeam.addNewParticipant(student1);  
 studentTeam.addNewParticipant(student2);  
  
 Team<Employee> employeeTeam = new Team<>("Robotyagi");  
 employeeTeam.addNewParticipant(employee1);  
 employeeTeam.addNewParticipant(employee2);  
  
 // Valid match  
 Team<Schoolar> scollarTeam2 = new Team<>("Rozumnyky");  
 Schoolar schoolar3 = new Schoolar("Sergey", 12);  
 Schoolar schoolar4 = new Schoolar("Olga", 14);  
 scollarTeam2.addNewParticipant(schoolar3);  
 scollarTeam2.addNewParticipant(schoolar4);  
 scollarTeam.playWith(scollarTeam2);  
  
 // Invalid match (won't compile): scollarTeam.playWith(employeeTeam);  
 }  
}

Резульатат :



***Завдання 3***.

Клонування:

- для класу Participant імплементувати інтерфейс Cloneable та перевизначити метод clone.

- для класу Participant перевизначити методи hashCode та equals.

- для класу Participant та його підкласів перевизначити метод toString.

- для класу Team Реалізувати глибоке клонування через статичний метод або конструктор копіювання.

- продемонструвати клонування та використання методів hashCode, equals та toString

Team

package com.education.ztu.game;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class Team<T extends Participant> {  
 private String name;  
 private List<T> participants = new ArrayList<>();  
  
 public Team(String name) {  
 this.name = name;  
 }  
  
 public void addNewParticipant(T participant) {  
 participants.add(participant);  
 System.*out*.println("To the team " + name + " was added participant " + participant.getName());  
 }  
  
 public void playWith(Team<T> otherTeam) {  
 Random random = new Random();  
 String winner = random.nextBoolean() ? this.name : otherTeam.name;  
 System.*out*.println("The team " + winner + " is the winner!");  
 }  
  
 public Team<T> deepClone() {  
 Team<T> clonedTeam = new Team<>(this.name);  
 for (T participant : this.participants) {  
 clonedTeam.addNewParticipant((T) participant.clone());  
 }  
 return clonedTeam;  
 }  
  
 @Override  
 public String toString() {  
 return "Team{name='" + name + "', participants=" + participants + "}";  
 }  
}

Game

package com.education.ztu.game;  
  
public class Game {  
 public static void main(String[] args) {  
 Schoolar schoolar1 = new Schoolar("Ivan", 13);  
 Schoolar schoolar2 = new Schoolar("Mariya", 15);  
 Student student1 = new Student("Mykola", 20);  
 Student student2 = new Student("Viktoria", 21);  
 Employee employee1 = new Employee("Andriy", 28);  
 Employee employee2 = new Employee("Oksana", 25);  
  
 Team<Schoolar> scollarTeam = new Team<>("Dragon");  
 scollarTeam.addNewParticipant(schoolar1);  
 scollarTeam.addNewParticipant(schoolar2);  
  
 Team<Student> studentTeam = new Team<>("Vpered");  
 studentTeam.addNewParticipant(student1);  
 studentTeam.addNewParticipant(student2);  
  
 Team<Employee> employeeTeam = new Team<>("Robotyagi");  
 employeeTeam.addNewParticipant(employee1);  
 employeeTeam.addNewParticipant(employee2);  
  
 // Valid match  
 Team<Schoolar> scollarTeam2 = new Team<>("Rozumnyky");  
 Schoolar schoolar3 = new Schoolar("Sergey", 12);  
 Schoolar schoolar4 = new Schoolar("Olga", 14);  
 scollarTeam2.addNewParticipant(schoolar3);  
 scollarTeam2.addNewParticipant(schoolar4);  
 scollarTeam.playWith(scollarTeam2);  
  
 // Invalid match (won't compile): scollarTeam.playWith(employeeTeam);  
 }  
}

Main

package com.education.ztu;  
  
import com.education.ztu.game.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 // Creating participants  
 Student student1 = new Student("Mykola", 20);  
 Student student2 = new Student("Viktoria", 21);  
 Student student3 = new Student("Mykola", 20); // Same properties as student1  
  
 // Demonstrating equals and hashCode  
 System.*out*.println("student1 equals student2: " + student1.equals(student2)); // false  
 System.*out*.println("student1 equals student3: " + student1.equals(student3)); // true  
 System.*out*.println("student1 hashCode: " + student1.hashCode());  
 System.*out*.println("student3 hashCode: " + student3.hashCode());  
  
 // Demonstrating toString  
 System.*out*.println("student1: " + student1);  
 System.*out*.println("student2: " + student2);  
  
 // Cloning participants  
 Student clonedStudent = (Student) student1.clone();  
 System.*out*.println("Cloned student: " + clonedStudent);  
  
 // Creating and cloning a team  
 Team<Student> studentTeam = new Team<>("Vpered");  
 studentTeam.addNewParticipant(student1);  
 studentTeam.addNewParticipant(student2);  
  
 System.*out*.println("Original team: " + studentTeam);  
 Team<Student> clonedTeam = studentTeam.deepClone();  
 System.*out*.println("Cloned team: " + clonedTeam);  
  
 // Modifying the cloned team to ensure deep copy  
 clonedTeam.addNewParticipant(new Student("Oksana", 22));  
 System.*out*.println("Modified cloned team: " + clonedTeam);  
 System.*out*.println("Original team after modifying cloned team: " + studentTeam);  
 }  
}

***Завдання 4***.

Порівняння:

- для класу Participant імплементувати інтерфейс Comparable та перевизначити метод compareTo для сортування учасників по імені.

- створити Comparator для порівняння учасників по віку.

- \*створити компаратор з пріорітетом використовуючи можливості Java 8 (спочатку порівняння по імені, а потім по віку).

- продемонструвати роботу порівнянь на прикладі сортування учасників команд.

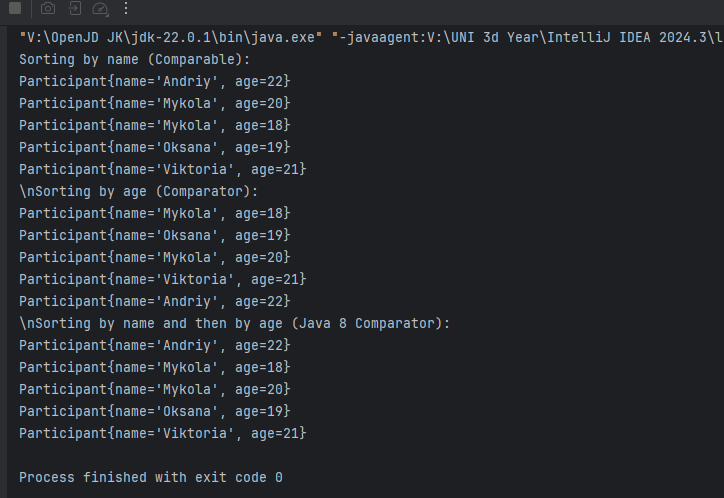
Participant.java

package com.education.ztu.game;  
  
import java.util.Objects;  
  
public abstract class Participant implements Cloneable, Comparable<Participant> {  
 private String name;  
 private int age;  
  
 public Participant(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 @Override  
 public int compareTo(Participant other) {  
 return this.name.compareTo(other.name);  
 }  
  
 @Override  
 public Participant clone() {  
 try {  
 return (Participant) super.clone();  
 } catch (CloneNotSupportedException e) {  
 throw new RuntimeException("Clone not supported", e);  
 }  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Participant that = (Participant) o;  
 return age == that.age && Objects.*equals*(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, age);  
 }  
  
 @Override  
 public String toString() {  
 return "Participant{name='" + name + "', age=" + age + "}";  
 }  
}

Main2:

package com.education.ztu;  
  
import com.education.ztu.game.\*;  
  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
public class Main2 {  
 public static void main(String[] args) {  
 List<Participant> participants = new ArrayList<>();  
 participants.add(new Student("Mykola", 20));  
 participants.add(new Student("Viktoria", 21));  
 participants.add(new Student("Andriy", 22));  
 participants.add(new Student("Oksana", 19));  
 participants.add(new Student("Mykola", 18));  
  
 System.*out*.println("Sorting by name (Comparable):");  
 participants.sort(null);  
 participants.forEach(System.*out*::println);  
  
 System.*out*.println("\\nSorting by age (Comparator):");  
 participants.sort(Comparator.*comparingInt*(Participant::getAge));  
 participants.forEach(System.*out*::println);  
  
 System.*out*.println("\\nSorting by name and then by age (Java 8 Comparator):");  
 participants.sort(Comparator.*comparing*(Participant::getName).thenComparing(Participant::getAge));  
 participants.forEach(System.*out*::println);  
 }  
}

Результат:



**Лабораторна робота №4**

**Тема:** Класи String, StringBuffer та StringBuilder. Локалізація та інтернаціоналізація. Робота з датами.

**Мета роботи:** робота з класами String, StringBuffer, StringBuilder та їх методами; практика використання локалізації та інтернаціоналізації; робота з датами.

**Завдання на лабораторну роботу**

***Завдання 1.*** Створити консольний Java проект java\_lab\_4 з пакетом

com.education.ztu

***Завдання 2.*** Практика методів класу String:

- Напишіть метод, який приймає як параметр будь-який рядок,

наприклад “I learn Java!!!”.

- Роздрукувати останній символ рядка.

- Перевірити, чи закінчується ваш рядок підрядком "!!!".

- Перевірити, чи починається ваш рядок підрядком "I learn ".

- Перевірити, чи містить ваш рядок підрядок "Java".

- Знайти позицію підрядка “Java” у рядку “I learn Java!!!”.

- Замінити всі символи "а" на "о".

- Перетворіть рядок на верхній регістр.

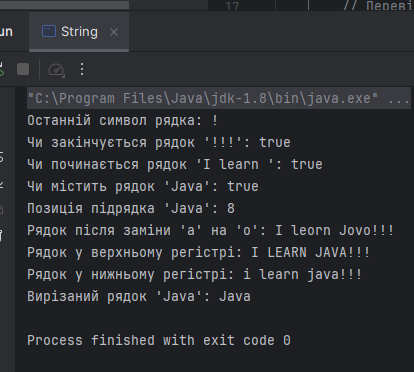
- Перетворіть рядок на нижній регістр.

- Вирізати рядок Java.

Лістинг програми

package com.education.ztu;  
  
public class String {  
 public static void main(java.lang.String[] args) {  
   
 java.lang.String inputString = "I learn Java!!!";  
  
   
 System.*out*.println("Останній символ рядка: " + inputString.charAt(inputString.length() - 1));  
  
   
 System.*out*.println("Чи закінчується рядок '!!!': " + inputString.endsWith("!!!"));  
  
 System.*out*.println("Чи починається рядок 'I learn ': " + inputString.startsWith("I learn "));  
  
 System.*out*.println("Чи містить рядок 'Java': " + inputString.contains("Java"));  
  
 System.*out*.println("Позиція підрядка 'Java': " + inputString.indexOf("Java"));  
  
 java.lang.String replacedString = inputString.replace('a', 'o');  
 System.*out*.println("Рядок після заміни 'a' на 'o': " + replacedString);  
  
   
 System.*out*.println("Рядок у верхньому регістрі: " + inputString.toUpperCase());  
  
 System.*out*.println("Рядок у нижньому регістрі: " + inputString.toLowerCase());  
  
 java.lang.String cutString = inputString.substring(inputString.indexOf("Java"), inputString.indexOf("Java") + 4);  
 System.*out*.println("Вирізаний рядок 'Java': " + cutString);  
 }  
}

Результат:



***Завдання 3.*** Створити рядок за допомогою класу StringBuilder або

StringBuffer та його методів:

- Дано два числа, наприклад, 4 і 36, необхідно скласти наступні рядки:

4 + 36 = 40

4 - 36 = -32

4 \* 36 = 144

- Використати метод StringBuilder.append().

- Замініть символ “=” на слово “рівно”. Використати методи

StringBuilder.insert(), StringBuilder.deleteCharAt().

- Замініть символ “=” на слово “рівно”. Використати метод

StringBuilder.replace().

- Змінити послідовність розташування символів в рядку на протилежну.

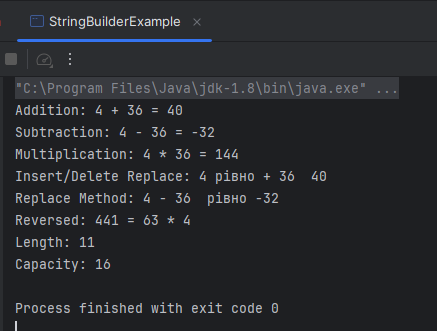
Використати метод StringBuilder.reverse().

- Визначити довжину та capacity.

Лістинг програми:  
package com.education.ztu;  
  
public class StringBuilderExample {  
 public static void main(String[] args) {  
 int a = 4;  
 int b = 36;

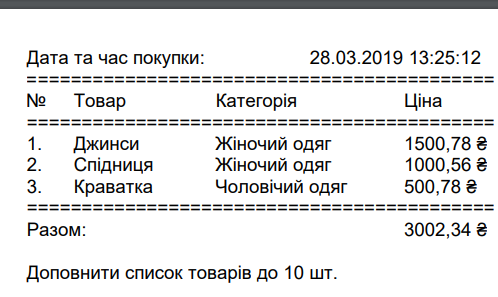
java.lang.StringBuilder addition = new java.lang.StringBuilder();  
 addition.append(a).append(" + ").append(b).append(" = ").append(a + b);  
 System.*out*.println("Addition: " + addition);  
  
 java.lang.StringBuilder subtraction = new java.lang.StringBuilder();  
 subtraction.append(a).append(" - ").append(b).append(" = ").append(a - b);  
 System.*out*.println("Subtraction: " + subtraction);  
  
 java.lang.StringBuilder multiplication = new java.lang.StringBuilder();  
 multiplication.append(a).append(" \* ").append(b).append(" = ").append(a \* b);  
 System.*out*.println("Multiplication: " + multiplication);  
  
   
 java.lang.StringBuilder replaceInsertDelete = new java.lang.StringBuilder(addition);  
 replaceInsertDelete.deleteCharAt(replaceInsertDelete.indexOf("="));  
 replaceInsertDelete.insert(replaceInsertDelete.indexOf(" "), " рівно");  
 System.*out*.println("Insert/Delete Replace: " + replaceInsertDelete);  
  
 java.lang.StringBuilder replaceMethod = new java.lang.StringBuilder(subtraction);  
 replaceMethod.replace(replaceMethod.indexOf("="), replaceMethod.indexOf("=") + 1, " рівно");  
 System.*out*.println("Replace Method: " + replaceMethod);  
  
   
 java.lang.StringBuilder reverseString = new java.lang.StringBuilder(multiplication);  
 reverseString.reverse();  
 System.*out*.println("Reversed: " + reverseString);  
  
   
 System.*out*.println("Length: " + addition.length());  
 System.*out*.println("Capacity: " + addition.capacity());  
 }  
}

Результат:



***Завдання 4.*** Вивести у форматованому вигляді чек з купленими

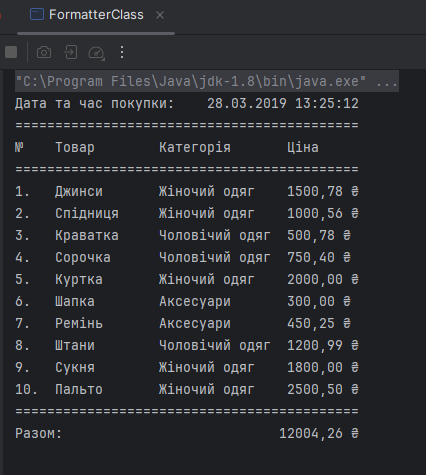
товарами використовуючи можливості класу Formatter:



Лістинг програми:

package com.education.ztu;  
  
import java.util.Formatter;  
  
public class FormatterClass {  
 public static void main(String[] args) {  
   
 String[][] items = {  
 {"1.", "Джинси", "Жіночий одяг", "1500,78 ₴"},  
 {"2.", "Спідниця", "Жіночий одяг", "1000,56 ₴"},  
 {"3.", "Краватка", "Чоловічий одяг", "500,78 ₴"},  
 {"4.", "Сорочка", "Чоловічий одяг", "750,40 ₴"},  
 {"5.", "Куртка", "Жіночий одяг", "2000,00 ₴"},  
 {"6.", "Шапка", "Аксесуари", "300,00 ₴"},  
 {"7.", "Ремінь", "Аксесуари", "450,25 ₴"},  
 {"8.", "Штани", "Чоловічий одяг", "1200,99 ₴"},  
 {"9.", "Сукня", "Жіночий одяг", "1800,00 ₴"},  
 {"10.", "Пальто", "Жіночий одяг", "2500,50 ₴"}  
 };  
  
 double total = 1500.78 + 1000.56 + 500.78 + 750.40 + 2000.00 +  
 300.00 + 450.25 + 1200.99 + 1800.00 + 2500.50;  
  
 Formatter formatter = new Formatter();  
 formatter.format("Дата та час покупки: %s\n", "28.03.2019 13:25:12");  
 formatter.format("===========================================\n");  
 formatter.format("%-4s %-12s %-15s %-10s\n", "№", "Товар", "Категорія", "Ціна");  
 formatter.format("===========================================\n");  
  
 for (String[] item : items) {  
 formatter.format("%-4s %-12s %-15s %-10s\n", item[0], item[1], item[2], item[3]);  
 }  
  
 formatter.format("===========================================\n");  
 formatter.format("%-30s %10.2f ₴\n", "Разом:", total);  
  
 System.*out*.println(formatter);  
 formatter.close();  
 }  
}

Разультат:



***Завдання 5***. Реалізувати інтернаціоналізацію для відображення

чеку з товарами українською, англійською та будь-якою третьою мовою на

ваш вибір. Для цього використати класи Locale та ResourceBundle. Для

виведення валюти країни використати можливості класу NumberFormat.

- створити директорію resources в корені проекту та позначити її як

директорію з ресурсами.

- створити три файли з розширенням properties для кожної локалі

(наприклад: data\_ua\_UA) та заповнити даними (для кирилиці використати

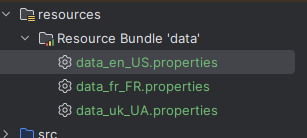
escape послідовності).

- об'єднати їх у Resourse Bundle

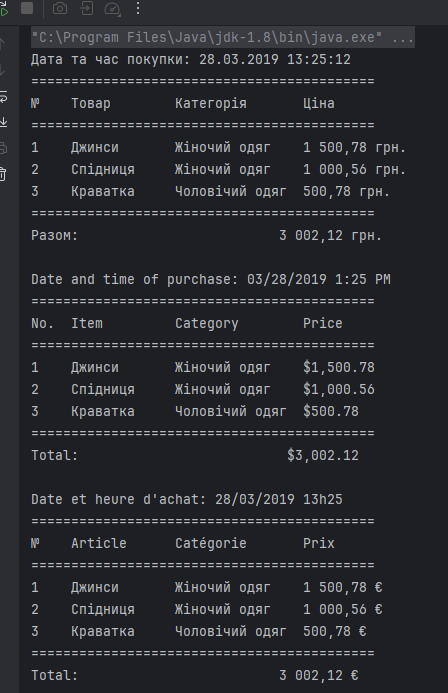
- реалізувати функціонал отримання та роботи з даними для кожної

локалі.Лістинг програми:

package com.education.ztu;  
  
import java.text.NumberFormat;  
import java.util.Locale;  
import java.util.ResourceBundle;  
  
public class ReceiptExample {  
 public static void main(String[] args) {  
 // Вибір мови  
 Locale uaLocale = new Locale("uk", "UA");  
 Locale enLocale = new Locale("en", "US");  
 Locale frLocale = new Locale("fr", "FR");  
  
 // Демонстрація трьох локалей  
 *printReceipt*(uaLocale);  
 *printReceipt*(enLocale);  
 *printReceipt*(frLocale);  
 }  
  
 private static void printReceipt(Locale locale) {  
 // Завантаження локалізованих ресурсів  
 ResourceBundle bundle = ResourceBundle.*getBundle*("data", locale);  
  
 // Дані про товари  
 Object[][] items = {  
 {1, "Джинси", "Жіночий одяг", 1500.78},  
 {2, "Спідниця", "Жіночий одяг", 1000.56},  
 {3, "Краватка", "Чоловічий одяг", 500.78}  
 };  
  
 // Форматування заголовка  
 System.*out*.println(bundle.getString("header") + " " + bundle.getString("date"));  
 System.*out*.println("===========================================");  
  
 // Форматування колонок  
 String[] columns = bundle.getString("columns").split(",");  
 System.*out*.printf("%-4s %-12s %-15s %-10s\n", columns[0], columns[1], columns[2], columns[3]);  
 System.*out*.println("===========================================");  
  
 // Форматування товарів  
 double total = 0;  
 NumberFormat currencyFormatter = NumberFormat.*getCurrencyInstance*(locale);  
  
 for (Object[] item : items) {  
 total += (double) item[3];  
 System.*out*.printf("%-4d %-12s %-15s %-10s\n", item[0], item[1], item[2],  
 currencyFormatter.format(item[3]));  
 }  
  
 // Вивід загальної суми  
 System.*out*.println("===========================================");  
 System.*out*.printf("%-30s %10s\n", bundle.getString("total"), currencyFormatter.format(total));  
 System.*out*.println();  
 }  
}



Результат



***Завдання 6.*** Робота з датами:

- Створіть об'єкт будь-якого класу для роботи з датами на власний вибір,

вказуючи дату та час початку сьогоднішньої лабораторної з Java.

- Вивести на консоль день тижня, день у році, місяць, рік, години,

хвилини, секунди.

- Перевірити чи рік високосний.

- Створіть об'єкт будь-якого класу для роботи з датами, який

представляє поточний час.

- Порівняйте його з датою початку лабораторної з Java, використовуючи

методи isAfter(), isBefore().

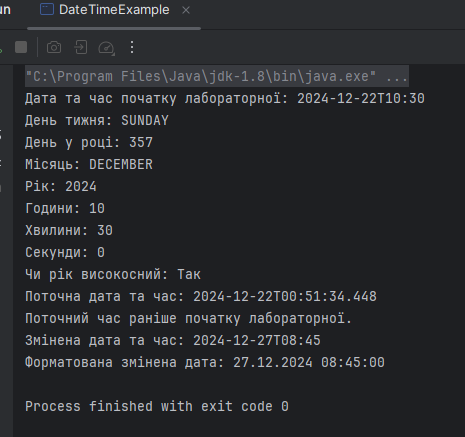
- Змініть значення елементів дати та часу на власний розсуд

використовуючи методи обраного вами класу для роботи з датами.

Лістинг програми:

package com.education.ztu;  
  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.time.DayOfWeek;  
import java.time.Month;  
  
public class DateTimeExample {  
 public static void main(String[] args) {  
 LocalDateTime labStartDateTime = LocalDateTime.*of*(2024, 12, 22, 10, 30, 0);  
 System.*out*.println("Дата та час початку лабораторної: " + labStartDateTime);  
   
 DayOfWeek dayOfWeek = labStartDateTime.getDayOfWeek();  
 int dayOfYear = labStartDateTime.getDayOfYear();  
 Month month = labStartDateTime.getMonth();  
 int year = labStartDateTime.getYear();  
 int hour = labStartDateTime.getHour();  
 int minute = labStartDateTime.getMinute();  
 int second = labStartDateTime.getSecond();  
  
 System.*out*.println("День тижня: " + dayOfWeek);  
 System.*out*.println("День у році: " + dayOfYear);  
 System.*out*.println("Місяць: " + month);  
 System.*out*.println("Рік: " + year);  
 System.*out*.println("Години: " + hour);  
 System.*out*.println("Хвилини: " + minute);  
 System.*out*.println("Секунди: " + second);  
   
 boolean isLeapYear = labStartDateTime.toLocalDate().isLeapYear();  
 System.*out*.println("Чи рік високосний: " + (isLeapYear ? "Так" : "Ні"));  
  
 LocalDateTime currentDateTime = LocalDateTime.*now*();  
 System.*out*.println("Поточна дата та час: " + currentDateTime);  
  
 if (currentDateTime.isAfter(labStartDateTime)) {  
 System.*out*.println("Поточний час пізніше початку лабораторної.");  
 } else if (currentDateTime.isBefore(labStartDateTime)) {  
 System.*out*.println("Поточний час раніше початку лабораторної.");  
 } else {  
 System.*out*.println("Поточний час співпадає з початком лабораторної.");  
 }  
   
 LocalDateTime modifiedDateTime = labStartDateTime.plusDays(5).minusHours(2).withMinute(45);  
 System.*out*.println("Змінена дата та час: " + modifiedDateTime);  
  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd.MM.yyyy HH:mm:ss");  
 System.*out*.println("Форматована змінена дата: " + modifiedDateTime.format(formatter));  
 }  
}

Результат виконання:



**Лабораторна робота №5**

**Тема:** Java Collections Framework

**Мета роботи:** робота з Java Collections Framework

Завдання на лабораторну роботу

***Завдання 1.*** Створити консольний Java проект java\_lab\_5 з пакетом

com.education.ztu

***Завдання 2.*** Створити клас Product та задати йому поля та методи

на власний вибір.

package com.education.ztu;  
  
public class Product {  
 private String name;  
 private double price;  
  
 public Product(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "Product{name='" + name + '\'' + ", price=" + price + '}';  
 }  
}

***Завдання 3***. Створити динамічний масив, що містить об'єкти класу

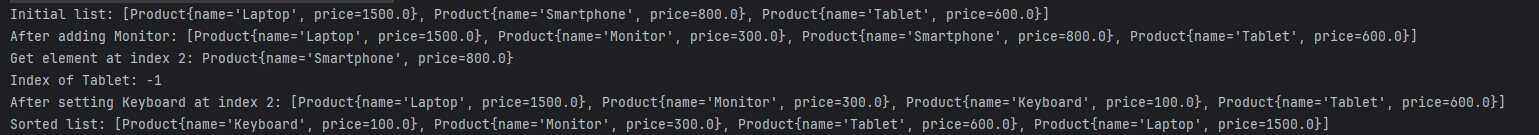
Product:

• Використовуємо клас ArrayList або LinkedList.

• Продемонструвати роботу з масивом використовуючи різні методи (add,

addAll, get, indexOf, lastIndexOf, iterator, listIterator, remove, set, sort,

subList, clear, contains, isEmpty, retainAll, size, toArray)

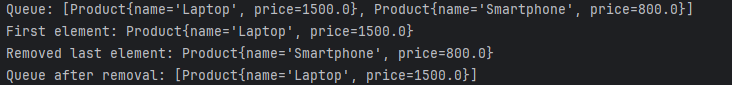


***Завдання 4***. Створити чергу, що містить об'єкти класу Product:

• Використовуємо клас ArrayDeque.

• Продемонструвати роботу з чергою використовуючи методи (push,

offerLast, getFirst, peekLast, pop, removeLast, pollLast та інші)



***Завдання 5.*** Створити множину, що містить об'єкти класу Product:

• Використовуємо клас TreeSet.

• Продемонструвати роботу з множиною використовуючи методи (add,

first, last, headSet, subSet, tailSet, ceiling, floor, higher, lower, pollFirst,

pollLast, descendingSet)  


**Завдання 6**. Створити Map що містить пари (ключ, значення) - ім'я

продукту та об'єкт продукту (клас Product).

• Використовуємо клас HashMap,

• Продемонструвати роботу з Map використовуючи методи (put, get, get,

containsKey, containsValue, clear, putIfAbsent, keySet, values, putAll,

remove, size)

• Викликати метод entrySet та продемонструвати роботу з набором

значень, що він поверне (getKey, getValue, setValue)



**Завдання 7**. Продемонструвати роботу з класом Collections:

• Для роботи використати масив створений через Arrays.asList

• Метод Collections.sort()

• Метод Collections.binarySearch()

• Методы Collections.reverse(), Collections.shuffle()

• Метод Collections.fill()

• Методы Collections.max(), Collections.min()

• Метод Collections.copy()

• Метод Collections.rotate()

• Метод Collections.checkedCollection()

• Метод Collections.frequency()

Лістинг Main:

package com.education.ztu;  
  
import java.util.\*;  
public class Main {  
 public static void main(String[] args) {  
 // Завдання 3: Динамічний масив  
 List<Product> productList = new ArrayList<>();  
 productList.add(new Product("Laptop", 1500));  
 productList.add(new Product("Smartphone", 800));  
 productList.add(new Product("Tablet", 600));  
  
 System.*out*.println("Initial list: " + productList);  
 productList.add(1, new Product("Monitor", 300));  
 System.*out*.println("After adding Monitor: " + productList);  
  
 Product product = productList.get(2);  
 System.*out*.println("Get element at index 2: " + product);  
  
 int index = productList.indexOf(new Product("Tablet", 600));  
 System.*out*.println("Index of Tablet: " + index);  
  
 productList.set(2, new Product("Keyboard", 100));  
 System.*out*.println("After setting Keyboard at index 2: " + productList);  
  
 productList.sort(Comparator.*comparing*(Product::getPrice));  
 System.*out*.println("Sorted list: " + productList);  
  
 // Завдання 4: Черга  
 ArrayDeque<Product> productQueue = new ArrayDeque<>();  
 productQueue.offerLast(new Product("Laptop", 1500));  
 productQueue.offerLast(new Product("Smartphone", 800));  
 System.*out*.println("Queue: " + productQueue);  
  
 Product first = productQueue.getFirst();  
 System.*out*.println("First element: " + first);  
  
 Product last = productQueue.pollLast();  
 System.*out*.println("Removed last element: " + last);  
 System.*out*.println("Queue after removal: " + productQueue);  
  
 // Завдання 5: Множина  
 TreeSet<Product> productSet = new TreeSet<>(Comparator.*comparing*(Product::getName));  
 productSet.add(new Product("Laptop", 1500));  
 productSet.add(new Product("Smartphone", 800));  
 productSet.add(new Product("Tablet", 600));  
  
 System.*out*.println("TreeSet: " + productSet);  
 Product firstProduct = productSet.first();  
 System.*out*.println("First product: " + firstProduct);  
  
 // Завдання 6: Map  
 Map<String, Product> productMap = new HashMap<>();  
 productMap.put("Laptop", new Product("Laptop", 1500));  
 productMap.put("Smartphone", new Product("Smartphone", 800));  
  
 Product retrievedProduct = productMap.get("Laptop");  
 System.*out*.println("Retrieved from Map: " + retrievedProduct);  
  
 // Завдання 7: Collections  
 List<Product> arrayList = Arrays.*asList*(  
 new Product("Item1", 200),  
 new Product("Item2", 300),  
 new Product("Item3", 100)  
 );  
  
 Collections.*sort*(arrayList, Comparator.*comparing*(Product::getPrice));  
 System.*out*.println("Sorted list: " + arrayList);  
  
 int indexBinarySearch = Collections.*binarySearch*(arrayList, new Product("Item2", 300), Comparator.*comparing*(Product::getPrice));  
 System.*out*.println("Binary search index of Item2: " + indexBinarySearch);  
 }  
}

Лабораторна робота №6

**Тема:** Обробка виключних ситуацій. Потоки вводу-виводу.

Робота з файлами.

**Мета роботи:** обробка виключних ситуація, створення власних класів

винятків, робота з потоками вводу-виводу.

Завдання на лабораторну роботу

***Завдання 1.*** Створити консольний Java проект java\_lab\_6 з пакетом

com.education.ztu. Створіть в корені проекту папку directory\_for\_files. Всі

файли з якими ви будете працювати при виконанні завдань повинні

знаходитись в ній.

***Завдання 2.*** Перевірка логіну та паролю:

• Створити статичний метод checkCredentials, який приймає на вхід три

параметри: login, password і confirmPassword.

• Login повинен містити лише латинські літери, цифри та знак

підкреслення. Довжина login має бути меншою за 20 символів. Якщо

login не відповідає цим вимогам, необхідно викинути

WrongLoginException.

• Password повинен містити лише латинські літери, цифри та знак

підкреслення. Довжина password має бути менше 20 символів. Також

password і confirmPassword повинні бути рівними. Якщо password не

відповідає цим вимогам, необхідно викинути WrongPasswordException.

• WrongPasswordException і WrongLoginException - користувацькі класи

виключення з двома конструкторами - один за замовчуванням, другий

приймає повідомлення виключення і передає його в конструктор класу

Exception.

• Обробка винятків проводиться усередині методу.

• Використовуємо multi-catch block.

• Метод повертає true, якщо значення є вірними або false в іншому

випадку.

package com.education.ztu;  
  
public class Main {  
 // Завдання 2: Метод перевірки логіну та паролю  
 public static boolean checkCredentials(String login, String password, String confirmPassword) {  
 try {  
 if (!login.matches("[a-zA-Z0-9\_]{1,19}")) {  
 throw new WrongLoginException("Login must contain only latin letters, numbers, underscores, and be less than 20 characters.");  
 }  
 if (!password.matches("[a-zA-Z0-9\_]{1,19}") || !password.equals(confirmPassword)) {  
 throw new WrongPasswordException("Password must meet requirements and match confirmPassword.");  
 }  
 return true;  
 } catch (WrongLoginException | WrongPasswordException e) {  
 System.*out*.println("Error: " + e.getMessage());  
 return false;  
 }  
 }  
  
 public static void main(String[] args) {  
 // Завдання 2: Перевірка логіну та паролю  
 System.*out*.println(*checkCredentials*("valid\_user", "password123", "password123")); // true  
 System.*out*.println(*checkCredentials*("invalid user", "password123", "password123")); // false  
 }  
}  
class WrongLoginException extends Exception {  
 public WrongLoginException() {  
 super("Invalid login");  
 }  
  
 public WrongLoginException(String message) {  
 super(message);  
 }  
}  
  
class WrongPasswordException extends Exception {  
 public WrongPasswordException() {  
 super("Invalid password");  
 }  
  
 public WrongPasswordException(String message) {  
 super(message);  
 }  
}

***Завдання 3.*** Запис звіту про покупки в текстовий файл та читання з

нього:

• Перевикористати код для формування звіту з покупок з лабораторної

роботи 4.

• Після покупки, записати звіт у файл, який містить інформацію про вміст

кошика.

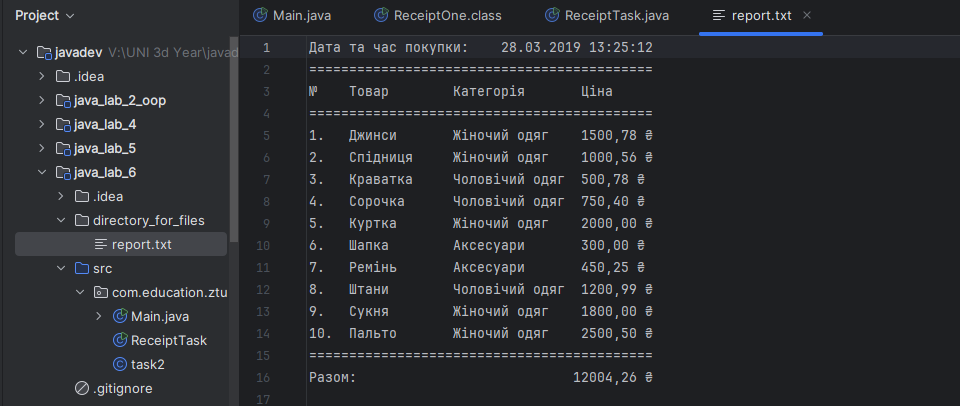
• Використовуємо клас FileWriter або PrintWriter для запису звіту.

• Використовуємо FileReader для читання звіту та відображення в

консолі.

• Не використовувати try-with-resources.

package com.education.ztu;  
  
import java.io.\*;  
import java.util.Formatter;  
  
public class ReceiptTask {  
  
 public static void main(String[] args) throws IOException {  
 // Створення папки для файлів  
 File directory = new File("javadev/java\_lab\_6/directory\_for\_files");  
 if (!directory.exists()) {  
 directory.mkdirs();  
 }  
  
 // Завдання: Формування звіту з лаби 4  
 String[][] items = new String[][]{{"1.", "Джинси", "Жіночий одяг", "1500,78 ₴"},  
 {"2.", "Спідниця", "Жіночий одяг", "1000,56 ₴"},  
 {"3.", "Краватка", "Чоловічий одяг", "500,78 ₴"},  
 {"4.", "Сорочка", "Чоловічий одяг", "750,40 ₴"},  
 {"5.", "Куртка", "Жіночий одяг", "2000,00 ₴"},  
 {"6.", "Шапка", "Аксесуари", "300,00 ₴"},  
 {"7.", "Ремінь", "Аксесуари", "450,25 ₴"},  
 {"8.", "Штани", "Чоловічий одяг", "1200,99 ₴"},  
 {"9.", "Сукня", "Жіночий одяг", "1800,00 ₴"},  
 {"10.", "Пальто", "Жіночий одяг", "2500,50 ₴"}};  
 double total = 12004.26;  
  
 // Запис звіту у файл  
 File reportFile = new File(directory, "report.txt");  
 try (FileWriter writer = new FileWriter(reportFile)) {  
 Formatter formatter = new Formatter(writer);  
 formatter.format("Дата та час покупки: %s\n", "28.03.2019 13:25:12");  
 formatter.format("===========================================\n");  
 formatter.format("%-4s %-12s %-15s %-10s\n", "№", "Товар", "Категорія", "Ціна");  
 formatter.format("===========================================\n");  
  
 for (String[] item : items) {  
 formatter.format("%-4s %-12s %-15s %-10s\n", item[0], item[1], item[2], item[3]);  
 }  
  
 formatter.format("===========================================\n");  
 formatter.format("%-30s %10.2f ₴\n", "Разом:", total);  
 formatter.close();  
 }  
  
 // Читання звіту з файлу  
 try (FileReader reader = new FileReader(reportFile);  
 BufferedReader bufferedReader = new BufferedReader(reader)) {  
 String line;  
 while ((line = bufferedReader.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 }  
 }  
}



***Завдання 4.*** Копіювання файлу до іншого файлу:

• Написати клас, який копіює вміст текстового файлу та картинки з одного

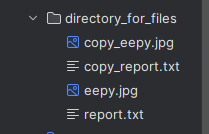
файлу до іншого.

• Використовуємо класи BufferedReader, FileReader, BufferedWriter,

FileWriter, FileInputStream, FileOutputStream.

• Використати try-with-resources.

package com.education.ztu;  
  
import java.io.\*;  
  
public class Copyfiles {  
  
 public static void main(String[] args) throws IOException {  
 // Шлях до вхідного та вихідного файлів  
 File sourceTextFile = new File("java\_lab\_6/directory\_for\_files/report.txt");  
 File targetTextFile = new File("java\_lab\_6/directory\_for\_files/copy\_report.txt");  
  
 File sourceImageFile = new File("java\_lab\_6/directory\_for\_files/eepy.jpg");  
 File targetImageFile = new File("java\_lab\_6/directory\_for\_files/copy\_eepy.jpg");  
  
 // Копіювання текстового файлу  
 try (BufferedReader reader = new BufferedReader(new FileReader(sourceTextFile));  
 BufferedWriter writer = new BufferedWriter(new FileWriter(targetTextFile))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 writer.write(line);  
 writer.newLine();  
 }  
 }  
 System.*out*.println("Текстовий файл скопійовано успішно.");  
  
 // Копіювання зображення  
 try (FileInputStream inputStream = new FileInputStream(sourceImageFile);  
 FileOutputStream outputStream = new FileOutputStream(targetImageFile)) {  
 byte[] buffer = new byte[1024];  
 int bytesRead;  
 while ((bytesRead = inputStream.read(buffer)) != -1) {  
 outputStream.write(buffer, 0, bytesRead);  
 }  
 }  
 System.*out*.println("Зображення скопійовано успішно.");  
 }  
}



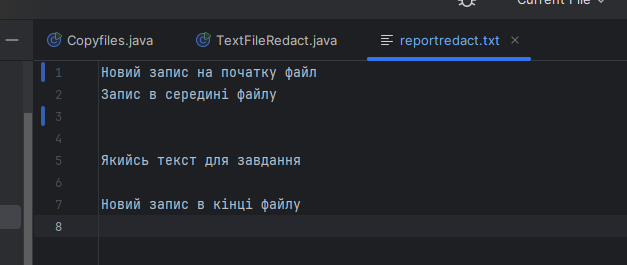
***Завдання 5.*** Робота з класом RandomAccessFile:

• Дописати текст в декількох місцях в текстовому файлі. Можна

використати текстовий файл зі списком товарів (наприклад, дописати

декілька товарів) або будь-який інший файл з текстом.

package com.education.ztu;  
  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
  
public class TextFileRedact {  
  
 public static void main(String[] args) throws IOException {  
 // Шлях до текстового файлу  
 File textFile = new File("java\_lab\_6/directory\_for\_files/reportredact.txt");  
  
 // Дописування тексту в файл  
 try (RandomAccessFile raf = new RandomAccessFile(textFile, "rw")) {  
 // Вставити текст на початок файлу  
 raf.seek(0);  
 String firstEntry = "Новий запис на початку файлу\n";  
 *insertTextAtPosition*(raf, firstEntry);  
  
 // Вставити текст в середину файлу  
 long middlePosition = raf.length() / 2;  
 raf.seek(middlePosition);  
 String middleEntry = "\nЗапис в середині файлу\n";  
 *insertTextAtPosition*(raf, middleEntry);  
  
 // Вставити текст в кінець файлу  
 raf.seek(raf.length());  
 String endEntry = "\nНовий запис в кінці файлу\n";  
 raf.write(endEntry.getBytes(StandardCharsets.*UTF\_8*));  
 }  
  
 System.*out*.println("Текст успішно дописано в файл.");  
 }  
  
 private static void insertTextAtPosition(RandomAccessFile raf, String text) throws IOException {  
 long currentPosition = raf.getFilePointer();  
 byte[] remainingBytes = new byte[(int) (raf.length() - currentPosition)];  
 raf.read(remainingBytes);  
  
 raf.seek(currentPosition);  
 raf.write(text.getBytes(StandardCharsets.*UTF\_8*));  
 raf.write(remainingBytes);  
 }  
}



***Завдання 6.*** Робота з класом File:

• Створити нову папку з ім'ям inner\_directory.

• Вивести абсолютний шлях створеної папки.

• Вивести ім’я батьківської директорії.

• Створити два текстових файли всередині папки inner\_directory.

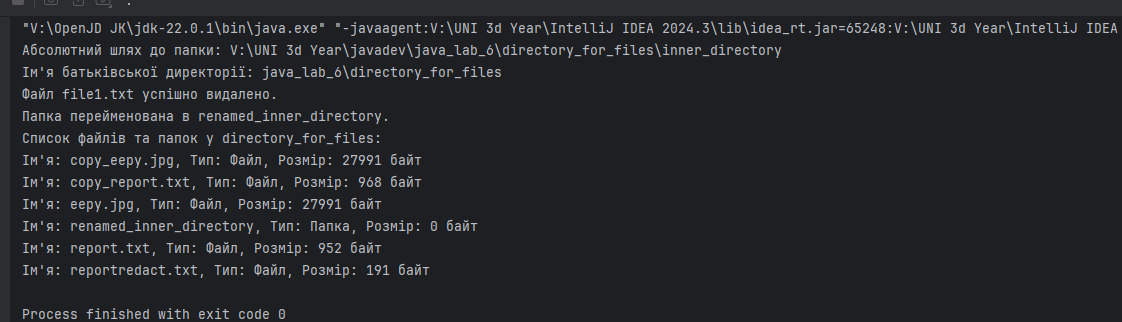
• Один файл видалити.

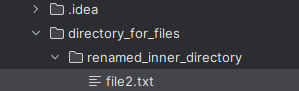
• Переіменувати папку inner\_directory в renamed\_inner\_directory

• Вивести список файлів та папок в папці directory\_for\_files, їх розмір та

тип (файл, папка).

package com.education.ztu;  
  
import java.io.\*;  
  
public class FileTask {  
  
 public static void main(String[] args) throws IOException {  
 // Шлях до головної папки  
 File mainDirectory = new File("java\_lab\_6/directory\_for\_files");  
  
 // Створення нової папки inner\_directory  
 File innerDirectory = new File(mainDirectory, "inner\_directory");  
 if (!innerDirectory.exists()) {  
 innerDirectory.mkdir();  
 }  
 System.*out*.println("Абсолютний шлях до папки: " + innerDirectory.getAbsolutePath());  
  
 // Виведення імені батьківської директорії  
 System.*out*.println("Ім'я батьківської директорії: " + innerDirectory.getParent());  
  
 // Створення двох текстових файлів  
 File file1 = new File(innerDirectory, "file1.txt");  
 File file2 = new File(innerDirectory, "file2.txt");  
 file1.createNewFile();  
 file2.createNewFile();  
  
 // Видалення одного файлу  
 if (file1.delete()) {  
 System.*out*.println("Файл file1.txt успішно видалено.");  
 } else {  
 System.*out*.println("Не вдалося видалити файл file1.txt.");  
 }  
  
 // Перейменування папки  
 File renamedDirectory = new File(mainDirectory, "renamed\_inner\_directory");  
 if (innerDirectory.renameTo(renamedDirectory)) {  
 System.*out*.println("Папка перейменована в renamed\_inner\_directory.");  
 } else {  
 System.*out*.println("Не вдалося перейменувати папку.");  
 }  
  
 // Виведення списку файлів та папок в directory\_for\_files  
 System.*out*.println("Список файлів та папок у directory\_for\_files:");  
 File[] files = mainDirectory.listFiles();  
 if (files != null) {  
 for (File file : files) {  
 System.*out*.println("Ім'я: " + file.getName() + ", Тип: " + (file.isDirectory() ? "Папка" : "Файл") + ", Розмір: " + file.length() + " байт");  
 }  
 }  
 }  
}





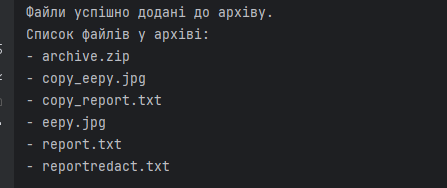
***Завдання 7.*** Створення архіву:

• Додати всі створені файли в папці directory\_for\_files до архіву.

Використати клас ZipOutputStream.

• Вивести список файлів з архіву. Використати клас ZipInputStream.

package com.education.ztu;  
  
import java.io.\*;  
import java.util.zip.\*;  
  
public class ArchiveTask {  
  
 public static void main(String[] args) throws IOException {  
 // Шлях до головної папки  
 File mainDirectory = new File("java\_lab\_6/directory\_for\_files");  
 File zipFile = new File(mainDirectory, "archive.zip");  
  
 // Створення архіву  
 try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zipFile))) {  
 File[] files = mainDirectory.listFiles();  
 if (files != null) {  
 for (File file : files) {  
 if (file.isFile()) {  
 try (FileInputStream fis = new FileInputStream(file)) {  
 ZipEntry zipEntry = new ZipEntry(file.getName());  
 zos.putNextEntry(zipEntry);  
  
 byte[] buffer = new byte[1024];  
 int bytesRead;  
 while ((bytesRead = fis.read(buffer)) != -1) {  
 zos.write(buffer, 0, bytesRead);  
 }  
 zos.closeEntry();  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println("Файли успішно додані до архіву.");  
  
 // Виведення списку файлів з архіву  
 try (ZipInputStream zis = new ZipInputStream(new FileInputStream(zipFile))) {  
 System.*out*.println("Список файлів у архіві:");  
 ZipEntry zipEntry;  
 while ((zipEntry = zis.getNextEntry()) != null) {  
 System.*out*.println("- " + zipEntry.getName());  
 }  
 }  
 }  
}



**Лабораторна робота №7**

**Тема:** Багатопоточне програмування в Java

**Мета роботи:** практика роботи з потоками в Java

**Завдання на лабораторну роботу**

***Завдання 1.*** Створити консольний Java проект java\_lab\_7 з пакетом

com.education.ztu.

***Завдання 2.*** Створити клас, що розширює Thread:

• Створити клас MyThread, що розширює Thread.

• Перевизначити метод run(). У циклі for вивести на консоль повідомлення

«Я люблю програмувати!!!» 100 разів.

• Створити екземпляр класу та запустити новий потік.

• Вивести ім'я створеного потоку, його пріорітет, превірити чи він живий,

чи є потоком демоном.

• Змінити ім'я, пріорітет створеного потоку та вивести в консоль оновлені

значення.

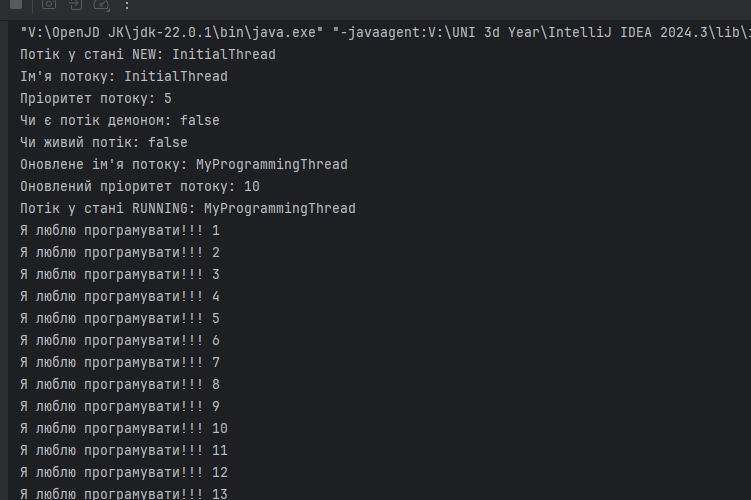
• Після завершення роботи створеного потоку (використати метод join())

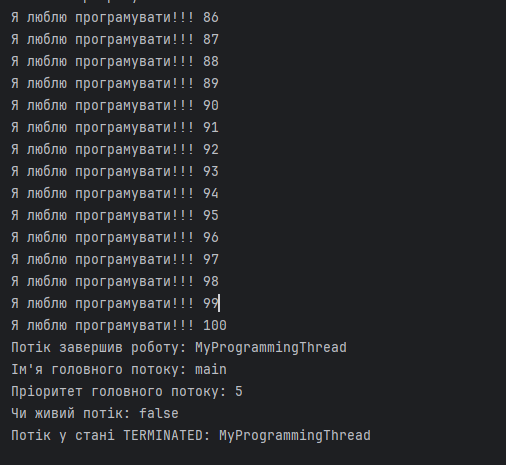
вивести ім’я головного потоку, та його пріорітет.

• Відобразити в консолі, коли ваш потік буде в стані NEW, RUNNUNG,

TERMINATED.

package com.education.ztu;  
  
public class MyThread extends Thread {  
  
 public MyThread(String name) {  
 super(name);  
 }  
  
 @Override  
 public void run() {  
 System.*out*.println("Потік у стані RUNNING: " + this.getName());  
 for (int i = 0; i < 100; i++) {  
 System.*out*.println("Я люблю програмувати!!! " + (i + 1));  
 }  
 System.*out*.println("Потік завершив роботу: " + this.getName());  
 }  
  
 public static void main(String[] args) {  
 MyThread myThread = new MyThread("InitialThread");  
  
 System.*out*.println("Потік у стані NEW: " + myThread.getName());  
 System.*out*.println("Ім'я потоку: " + myThread.getName());  
 System.*out*.println("Пріоритет потоку: " + myThread.getPriority());  
 System.*out*.println("Чи є потік демоном: " + myThread.isDaemon());  
 System.*out*.println("Чи живий потік: " + myThread.isAlive());  
   
 myThread.setName("MyProgrammingThread");  
 myThread.setPriority(Thread.*MAX\_PRIORITY*);  
 System.*out*.println("Оновлене ім'я потоку: " + myThread.getName());  
 System.*out*.println("Оновлений пріоритет потоку: " + myThread.getPriority());  
  
 myThread.start();  
  
 try {  
 myThread.join();  
 } catch (InterruptedException e) {  
 System.*out*.println("Потік було перервано.");  
 }  
  
 Thread mainThread = Thread.*currentThread*();  
 System.*out*.println("Ім'я головного потоку: " + mainThread.getName());  
 System.*out*.println("Пріоритет головного потоку: " + mainThread.getPriority());  
   
 System.*out*.println("Чи живий потік: " + myThread.isAlive());  
 System.*out*.println("Потік у стані TERMINATED: " + myThread.getName());  
 }  
}





***Завдання 3.*** Створити клас, що реалізує інтерфейс Runnable для

виводу в консоль чисел від 0 до 10000, що діляться на 10 без залишку:

• Створити клас MyRunnable, який реалізує інтерфейс Runnable.

• Імплементувати метод run().

• Визначити умову, якщо потік хочуть перервати, то завершити роботу

потоку та вивести повідомлення «Розрахунок завершено!!!»

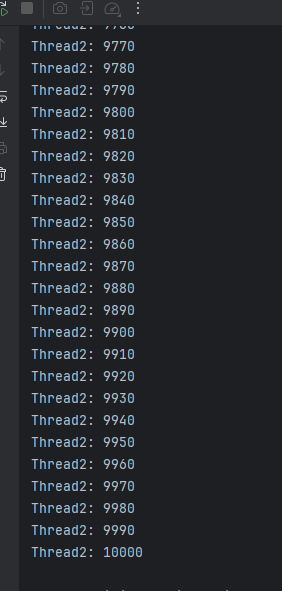
• Створити три потоки, які виконують завдання друку значень.

• Використовуємо статичний метод Thread.sleep(), щоб зробити паузу на

2 секунди для головного потоку, а після цього викликати для створених

потоків метод interrupt().

package com.education.ztu;  
  
// Завдання 3: Створити клас, що реалізує інтерфейс Runnable  
class MyRunnable implements Runnable {  
 @Override  
 public void run() {  
 for (int i = 0; i <= 10000; i++) {  
 if (Thread.*currentThread*().isInterrupted()) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": Розрахунок завершено!!!");  
 return;  
 }  
 if (i % 10 == 0) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + i);  
 }  
 }  
 }  
}  
  
public class MyRunnableDemo {  
 public static void main(String[] args) {  
 // Створення і запуск трьох потоків  
 MyRunnable myRunnable = new MyRunnable();  
 Thread thread1 = new Thread(myRunnable, "Thread1");  
 Thread thread2 = new Thread(myRunnable, "Thread2");  
 Thread thread3 = new Thread(myRunnable, "Thread3");  
  
 thread1.start();  
 thread2.start();  
 thread3.start();  
  
 try {  
 // Зробити паузу на 2 секунди  
 Thread.*sleep*(2000);  
 } catch (InterruptedException e) {  
 System.*out*.println("Головний потік було перервано.");  
 }  
  
 // Перервати потоки  
 thread1.interrupt();  
 thread2.interrupt();  
 thread3.interrupt();  
 }  
}



***Завдання 4.*** Створити клас, що реалізує інтерфейс Runnable для

вививедення арифметичної прогресії від 1 до 100 з кроком 1:

• Створити клас, який реалізує інтерфейс Runnable.

• Створити об'єкт зі статичною змінною result для збереження значення

арифметичної прогресії.

• Перевизначити метод run(). Створити цикл for. У циклі виводимо через

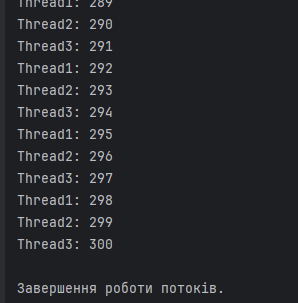
пробіл значення змінної result. Та додаємо наступне значення до змінної

result та чекаємо 0,2 секунду.

• Забезпечити корректну роботу використовуючи синхронізований метод.

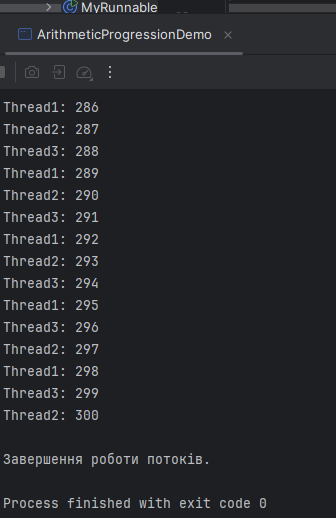
• Створити три потоки, які виконують завдання друку значень.

package com.education.ztu;  
  
class ArithmeticProgression implements Runnable {  
 private static int *result* = 1;  
  
 // Синхронізований метод для забезпечення коректної роботи  
 private synchronized void printProgression() {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + *result*);  
 *result*++;  
 }  
  
 @Override  
 public void run() {  
 for (int i = 1; i <= 100; i++) {  
 printProgression();  
 try {  
 Thread.*sleep*(200); // Чекати 0,2 секунди  
 } catch (InterruptedException e) {  
 System.*out*.println(Thread.*currentThread*().getName() + " було перервано.");  
 return;  
 }  
 }  
 }  
}  
  
public class ArithmeticProgressionDemo {  
 public static void main(String[] args) {  
 // Створення об'єкта Runnable  
 ArithmeticProgression progressionTask = new ArithmeticProgression();  
  
 // Створення трьох потоків  
 Thread thread1 = new Thread(progressionTask, "Thread1");  
 Thread thread2 = new Thread(progressionTask, "Thread2");  
 Thread thread3 = new Thread(progressionTask, "Thread3");  
  
 // Запуск потоків  
 thread1.start();  
 thread2.start();  
 thread3.start();  
  
 try {  
 thread1.join();  
 thread2.join();  
 thread3.join();  
 } catch (InterruptedException e) {  
 System.*out*.println("Головний потік було перервано.");  
 }  
  
 System.*out*.println("\nЗавершення роботи потоків.");  
 }  
}



***Завдання 5***. Переробити 4 завдання використовуючи блок синхронізації.

package com.education.ztu;  
  
class ArithmeticProgression implements Runnable {  
 private static int *result* = 1; // Статична змінна для збереження результату прогресії  
  
 @Override  
 public void run() {  
 for (int i = 1; i <= 100; i++) {  
 synchronized (ArithmeticProgression.class) { // Використання блоку синхронізації  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + *result*);  
 *result*++;  
 }  
 try {  
 Thread.*sleep*(200); // Чекати 0,2 секунди  
 } catch (InterruptedException e) {  
 System.*out*.println(Thread.*currentThread*().getName() + " було перервано.");  
 return;  
 }  
 }  
 }  
}  
  
public class ArithmeticProgressionDemo {  
 public static void main(String[] args) {  
 // Створення об'єкта Runnable  
 ArithmeticProgression progressionTask = new ArithmeticProgression();  
  
 // Створення трьох потоків  
 Thread thread1 = new Thread(progressionTask, "Thread1");  
 Thread thread2 = new Thread(progressionTask, "Thread2");  
 Thread thread3 = new Thread(progressionTask, "Thread3");  
  
 // Запуск потоків  
 thread1.start();  
 thread2.start();  
 thread3.start();  
  
 try {  
 thread1.join();  
 thread2.join();  
 thread3.join();  
 } catch (InterruptedException e) {  
 System.*out*.println("Головний потік було перервано.");  
 }  
  
 System.*out*.println("\nЗавершення роботи потоків.");  
 }  
}



***Завдання 6.*** Створити два потоки Reader та Printer. Reader зчитує

введені дані з консолі та записує в змінну. Після цього інформує потік

Printer та засипає на 1 секунду, а потік Reader виводить дотриманий

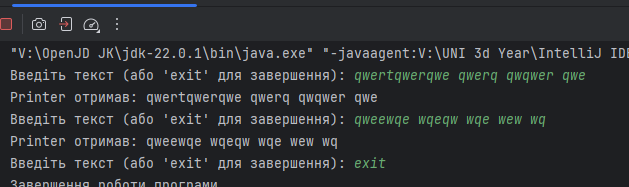
рядок. І так повторюється знову, поки користувач не завершить роботу

програми.

• Змінну треба використати як об'єкт для синхронізації.

• Тут необхідно використати wait() i notify().

package com.education.ztu;  
  
import java.util.Scanner;  
  
class SharedData {  
 private String data;  
 private boolean hasData = false;  
  
 public synchronized void writeData(String input) {  
 while (hasData) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Writer потік було перервано.");  
 }  
 }  
 data = input;  
 hasData = true;  
 notify();  
 }  
  
 public synchronized String readData() {  
 while (!hasData) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Reader потік було перервано.");  
 }  
 }  
 hasData = false;  
 notify();  
 return data;  
 }  
}  
  
class Reader implements Runnable {  
 private final SharedData sharedData;  
  
 public Reader(SharedData sharedData) {  
 this.sharedData = sharedData;  
 }  
  
 @Override  
 public void run() {  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.print("Введіть текст (або 'exit' для завершення): ");  
 String input = scanner.nextLine();  
 if (input.equalsIgnoreCase("exit")) {  
 sharedData.writeData("exit");  
 break;  
 }  
 sharedData.writeData(input);  
 try {  
 Thread.*sleep*(1000); // Заснути на 1 секунду  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Reader потік було перервано.");  
 }  
 }  
 scanner.close();  
 }  
}  
  
class Printer implements Runnable {  
 private final SharedData sharedData;  
  
 public Printer(SharedData sharedData) {  
 this.sharedData = sharedData;  
 }  
  
 @Override  
 public void run() {  
 while (true) {  
 String data = sharedData.readData();  
 if (data.equalsIgnoreCase("exit")) {  
 break;  
 }  
 System.*out*.println("Printer отримав: " + data);  
 }  
 }  
}  
  
public class ReaderPrinterDemo {  
 public static void main(String[] args) {  
 SharedData sharedData = new SharedData();  
  
 Thread readerThread = new Thread(new Reader(sharedData), "Reader");  
 Thread printerThread = new Thread(new Printer(sharedData), "Printer");  
  
 readerThread.start();  
 printerThread.start();  
  
 try {  
 readerThread.join();  
 printerThread.join();  
 } catch (InterruptedException e) {  
 System.*out*.println("Головний потік було перервано.");  
 }  
  
 System.*out*.println("Завершення роботи програми.");  
 }  
}



***Завдання 7.*** Створити програму для знаходження суми цифр в масиві

на 1 000 000 елементів:

• Заповнити масив числами використовуючи клас Random.

• Реалізувати задачу в однопоточному та багатопоточному середовищі.

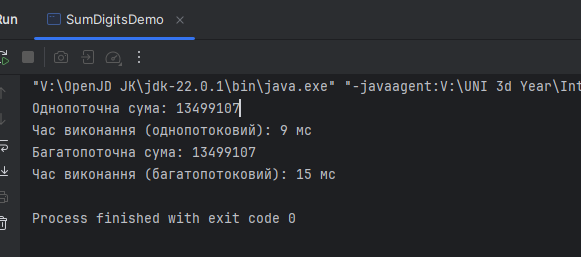
• Для багатопоточного середовища використати ExecutorService на 5

потоків та об'єкти потоків, що імплементують інтерфейси Runnable або

Callable.

• Заміряти час виконання обох варіантів завдання використовуючи

System.currentTimeMillis() та вивести результати в консоль. package com.education.ztu;  
  
import java.util.Random;  
import java.util.concurrent.\*;  
  
public class SumDigitsDemo {  
  
 private static final int *ARRAY\_SIZE* = 1\_000\_000;  
 private static final int *THREAD\_COUNT* = 5;  
  
 public static void main(String[] args) throws InterruptedException, ExecutionException {  
 // Заповнення масиву випадковими числами  
 int[] numbers = new int[*ARRAY\_SIZE*];  
 Random random = new Random();  
 for (int i = 0; i < *ARRAY\_SIZE*; i++) {  
 numbers[i] = random.nextInt(1000); // Числа від 0 до 999  
 }  
  
 // Однопоточне виконання  
 long singleThreadStart = System.*currentTimeMillis*();  
 long singleThreadSum = *calculateSumSingleThread*(numbers);  
 long singleThreadEnd = System.*currentTimeMillis*();  
  
 System.*out*.println("Однопоточна сума: " + singleThreadSum);  
 System.*out*.println("Час виконання (однопотоковий): " + (singleThreadEnd - singleThreadStart) + " мс");  
  
 // Багатопоточне виконання  
 long multiThreadStart = System.*currentTimeMillis*();  
 long multiThreadSum = *calculateSumMultiThread*(numbers);  
 long multiThreadEnd = System.*currentTimeMillis*();  
  
 System.*out*.println("Багатопоточна сума: " + multiThreadSum);  
 System.*out*.println("Час виконання (багатопотоковий): " + (multiThreadEnd - multiThreadStart) + " мс");  
 }  
  
 private static long calculateSumSingleThread(int[] numbers) {  
 long sum = 0;  
 for (int number : numbers) {  
 sum += *sumDigits*(number);  
 }  
 return sum;  
 }  
  
 private static long calculateSumMultiThread(int[] numbers) throws InterruptedException, ExecutionException {  
 ExecutorService executor = Executors.*newFixedThreadPool*(*THREAD\_COUNT*);  
 int chunkSize = *ARRAY\_SIZE* / *THREAD\_COUNT*;  
 Future<Long>[] futures = new Future[*THREAD\_COUNT*];  
  
 for (int i = 0; i < *THREAD\_COUNT*; i++) {  
 int start = i \* chunkSize;  
 int end = (i == *THREAD\_COUNT* - 1) ? *ARRAY\_SIZE* : (i + 1) \* chunkSize;  
 futures[i] = executor.submit(new SumTask(numbers, start, end));  
 }  
  
 long totalSum = 0;  
 for (Future<Long> future : futures) {  
 totalSum += future.get();  
 }  
  
 executor.shutdown();  
 return totalSum;  
 }  
  
 private static int sumDigits(int number) {  
 int sum = 0;  
 while (number != 0) {  
 sum += number % 10;  
 number /= 10;  
 }  
 return sum;  
 }  
  
 static class SumTask implements Callable<Long> {  
 private final int[] numbers;  
 private final int start;  
 private final int end;  
  
 public SumTask(int[] numbers, int start, int end) {  
 this.numbers = numbers;  
 this.start = start;  
 this.end = end;  
 }  
  
 @Override  
 public Long call() {  
 long sum = 0;  
 for (int i = start; i < end; i++) {  
 sum += *sumDigits*(numbers[i]);  
 }  
 return sum;  
 }  
 }  
}



**Лабораторна робота №8**

**Тема:** Лямбда вирази. Функціональні інтерфейси. Посилання на

методи. Stream API.

**Мета роботи:** практика роботи з лямбда виразами, функціональними

інтерфейсами; використання посилань на методи та Stream API при

розробці програм на Java.

**Завдання на лабораторну роботу**

***Завдання 1***. Створити консольний Java проект java\_lab\_8 з пакетом

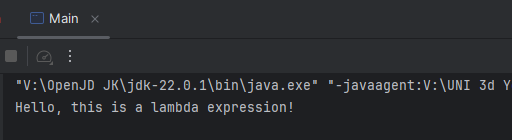
com.education.ztu.

***Завдання 2.*** Описати власний функціональний інтерфейс Printable з

методом void print() та написати лямбда вираз цього інтерфейсу.

package com.education.ztu;  
  
public interface Printable {  
 void print();  
}

package com.education.ztu;  
  
public class Main {  
 public static void main(String[] args) {  
 Printable printable = () -> System.out.println("Hello, this is a lambda expression!");  
  
 printable.print();  
 }  
}



***Завдання 3.*** Написати лямбда вирази для вбудованих

функціональних інтерфейсів:

a) Створити лямбда вираз, який повертає значення true, якщо рядок

можна привести до числа, використовуючи функціональний інтерфейс

Predicate.

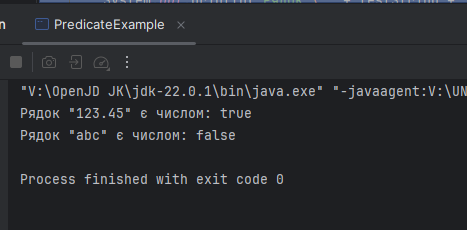
Створити вираз лямбда, який перевіряє, що рядок можна привести до

числа, використовуючи функціональний інтерфейс Predicate.

Написати програму, яка перевіряє, що рядок можна привести до числа,

використовуючи метод and() функціонального інтерфейсу Predicate.

package com.education.ztu;  
  
import java.util.function.Predicate;  
  
public class PredicateExample {  
 public static void main(String[] args) {  
 // Лямбда вираз для перевірки, чи рядок можна привести до числа  
 Predicate<String> isNumeric = str -> {  
 try {  
 Double.*parseDouble*(str);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 };  
  
 // Використання Predicate і методу and()  
 Predicate<String> notEmpty = str -> !str.isEmpty();  
 Predicate<String> validNumber = isNumeric.and(notEmpty);  
  
 // Перевірка рядків  
 String testString = "123.45";  
 System.*out*.println("Рядок \"" + testString + "\" є числом: " + validNumber.test(testString));  
  
 testString = "abc";  
 System.*out*.println("Рядок \"" + testString + "\" є числом: " + validNumber.test(testString));  
 }  
}

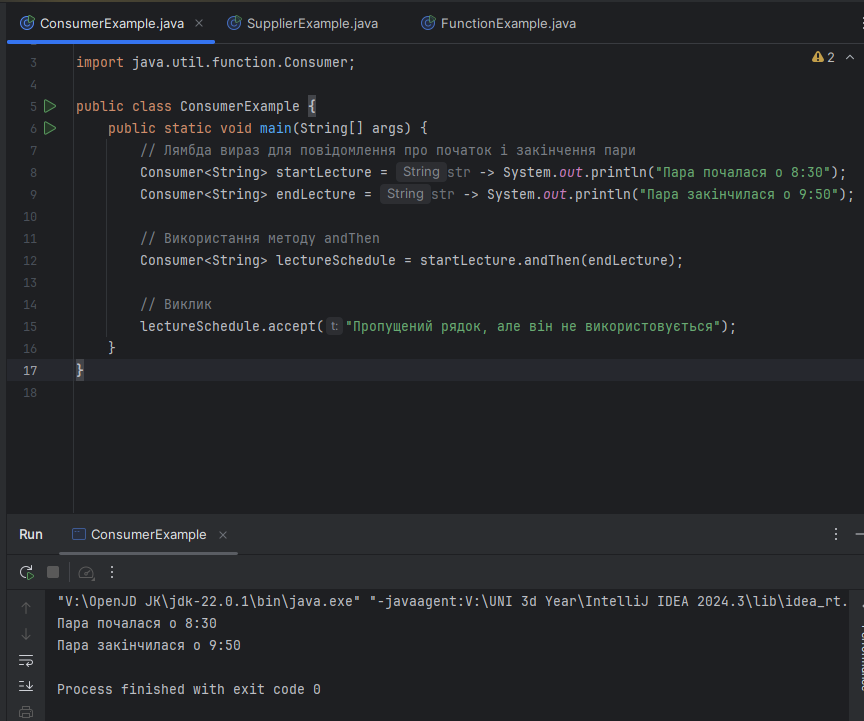


b) Написати лямбда вираз, який приймає на вхід рядок і виводить на

консоль повідомлення "Пара почалася о 8:30", "Пара закінчилася о

9:50". Використовуємо функціональний інтерфейс Consumer і метод за

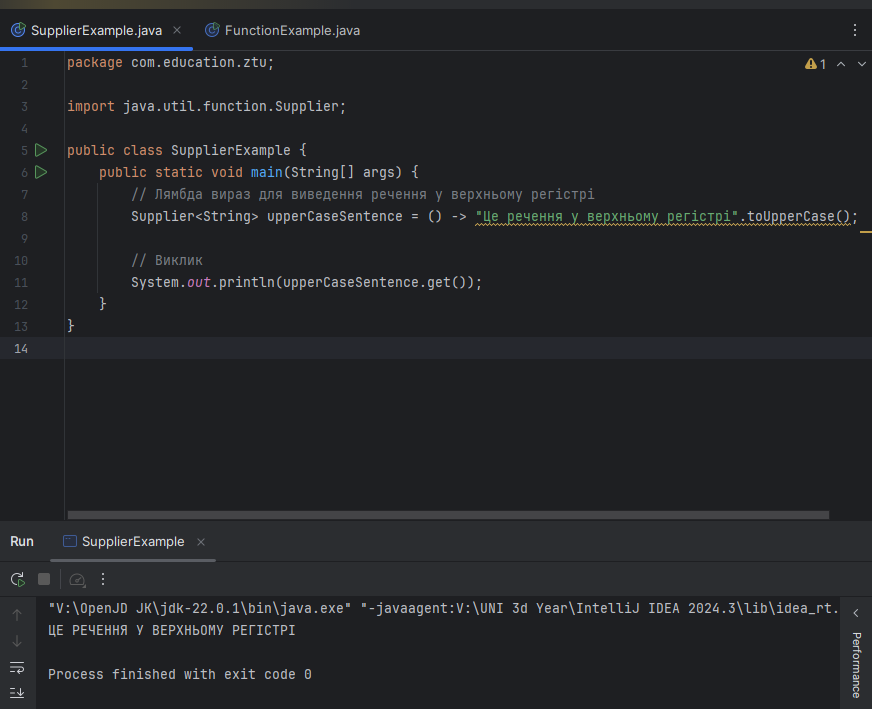
замовчуванням andThen.



c) Написати лямбда вираз, який виводить в консоль речення в з

літерами у верхньому регістрі. Використовуємо функціональний

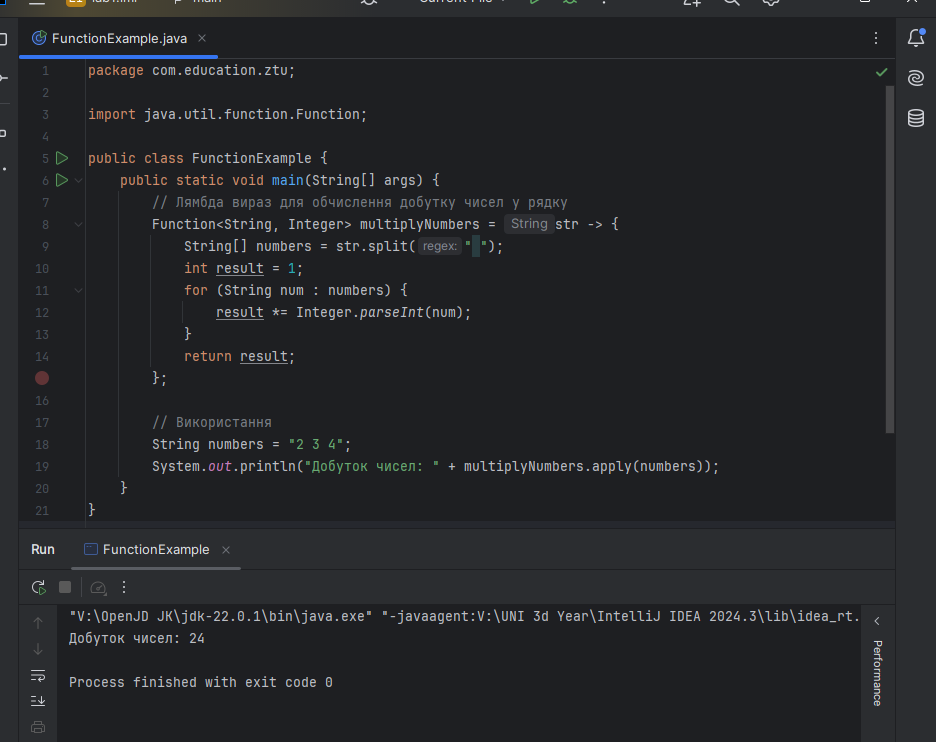
інтерфейс Supplier.



d) Написати лямбда вираз, який приймає на вхід рядок з набором чисел

через пробіл та повертає добуток цих чисел. Використовуємо

функціональний інтерфейс Function<String, Integer>.



***Завдання 4.*** Stream API.

• Створити стрім з масиву Product з полями name, brand, price, count.

• Отримати всі бренди та вивести в консоль. (map)

• Отримати 2 товари ціна яких менше тисячі. (filter, limit)

• Отримати суму всіх видів товарів, що є на складі. (reduce)

• Згрупувати товари по бренду (Collectors.groupingBy())

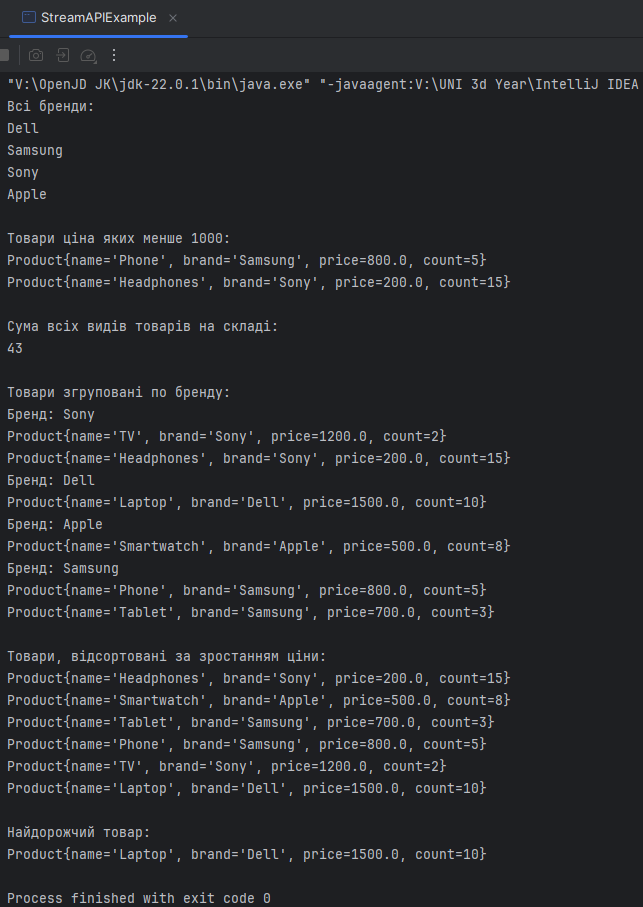
• Відсортувати товари за зростанням ціни та повернути масив (sorted,

Collectors)

• За бажанням дописати функціонал, що використовує інші методи

стрімів.

package com.education.ztu;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
class Product {  
 String name;  
 String brand;  
 double price;  
 int count;  
  
 public Product(String name, String brand, double price, int count) {  
 this.name = name;  
 this.brand = brand;  
 this.price = price;  
 this.count = count;  
 }  
  
 @Override  
 public String toString() {  
 return "Product{" +  
 "name='" + name + '\'' +  
 ", brand='" + brand + '\'' +  
 ", price=" + price +  
 ", count=" + count +  
 '}';  
 }  
}  
  
public class StreamAPIExample {  
 public static void main(String[] args) {  
 List<Product> products = Arrays.*asList*(  
 new Product("Laptop", "Dell", 1500, 10),  
 new Product("Phone", "Samsung", 800, 5),  
 new Product("TV", "Sony", 1200, 2),  
 new Product("Headphones", "Sony", 200, 15),  
 new Product("Smartwatch", "Apple", 500, 8),  
 new Product("Tablet", "Samsung", 700, 3)  
 );  
  
 // всі бренди та вивести в консоль (map)  
 System.*out*.println("Всі бренди:");  
 products.stream()  
 .map(product -> product.brand)  
 .distinct()  
 .forEach(System.*out*::println);  
  
 // отримати 2 товари ціна яких менше тисячі (filter, limit)  
 System.*out*.println("\nТовари ціна яких менше 1000:");  
 products.stream()  
 .filter(product -> product.price < 1000)  
 .limit(2)  
 .forEach(System.*out*::println);  
  
 // отримати суму всіх видів товарів що є на складі (reduce)  
 System.*out*.println("\nСума всіх видів товарів на складі:");  
 int totalCount = products.stream()  
 .map(product -> product.count)  
 .reduce(0, Integer::*sum*);  
 System.*out*.println(totalCount);  
  
 // згрупувати товари по бренду  
 System.*out*.println("\nТовари згруповані по бренду:");  
 Map<String, List<Product>> groupedByBrand = products.stream()  
 .collect(Collectors.*groupingBy*(product -> product.brand));  
 groupedByBrand.forEach((brand, productList) -> {  
 System.*out*.println("Бренд: " + brand);  
 productList.forEach(System.*out*::println);  
 });  
  
 // Відсортувати товари за зростанням ціни та повернути масив  
 System.*out*.println("\nТовари, відсортовані за зростанням ціни:");  
 List<Product> sortedByPrice = products.stream()  
 .sorted(Comparator.*comparingDouble*(product -> product.price))  
 .collect(Collectors.*toList*());  
 sortedByPrice.forEach(System.*out*::println);  
  
 // знайти найдорожчий товар  
 System.*out*.println("\nНайдорожчий товар:");  
 products.stream()  
 .max(Comparator.*comparingDouble*(product -> product.price))  
 .ifPresent(System.*out*::println);  
 }  
}



***Завдання 5***. Посилання на методи чи конструктори.

В попередньому завданні, де це можливо, виклики переробити на

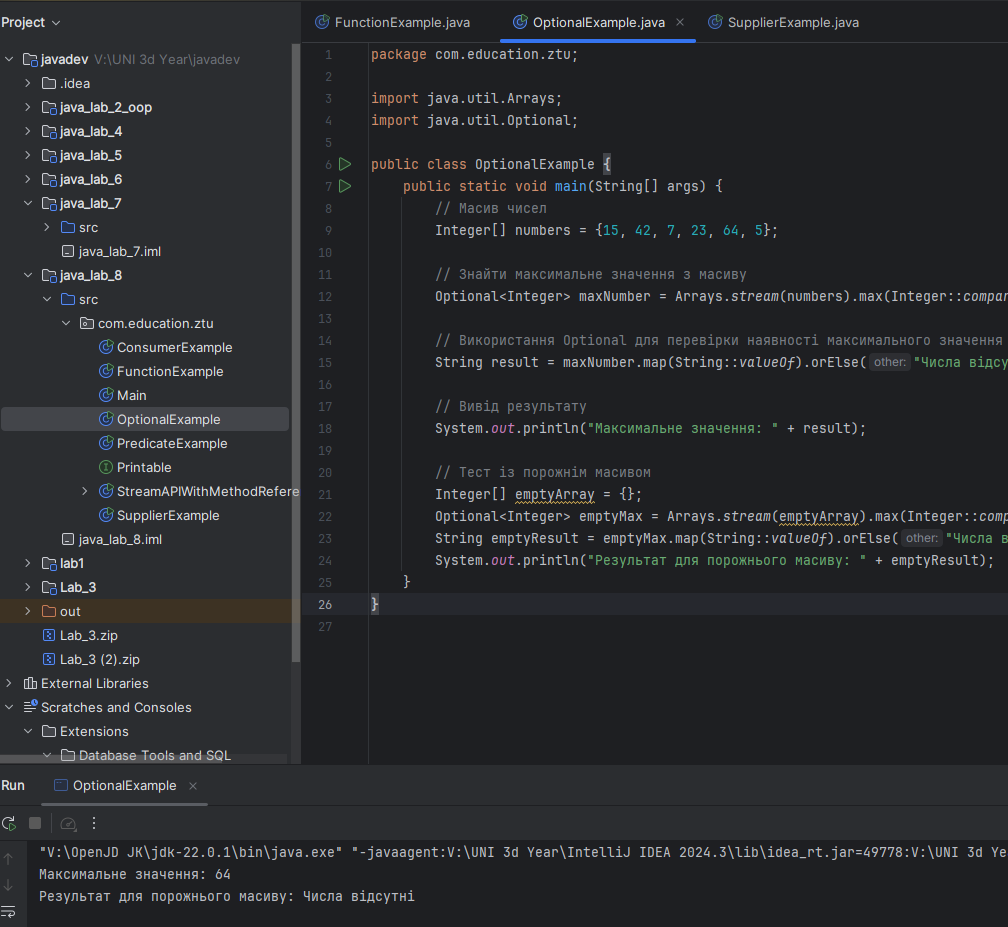
посилання на методи чи конструктори

package com.education.ztu;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
class Product {  
 private String name;  
 private String brand;  
 private double price;  
 private int count;  
  
 public Product(String name, String brand, double price, int count) {  
 this.name = name;  
 this.brand = brand;  
 this.price = price;  
 this.count = count;  
 }  
  
 // Гетери для полів  
 public String getName() {  
 return name;  
 }  
  
 public String getBrand() {  
 return brand;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public int getCount() {  
 return count;  
 }  
  
 @Override  
 public String toString() {  
 return "Product{" +  
 "name='" + name + '\'' +  
 ", brand='" + brand + '\'' +  
 ", price=" + price +  
 ", count=" + count +  
 '}';  
 }  
}  
public class StreamAPIWithMethodReferences {  
 public static void main(String[] args) {  
 // Створення масиву продуктів  
 List<Product> products = Arrays.*asList*(  
 new Product("Laptop", "Dell", 1500, 10),  
 new Product("Phone", "Samsung", 800, 5),  
 new Product("TV", "Sony", 1200, 2),  
 new Product("Headphones", "Sony", 200, 15),  
 new Product("Smartwatch", "Apple", 500, 8),  
 new Product("Tablet", "Samsung", 700, 3)  
 );  
  
 // Отримати всі бренди та вивести в консоль (map)  
 System.*out*.println("Всі бренди:");  
 products.stream()  
 .map(Product::getBrand) // Використання гетера  
 .distinct()  
 .forEach(System.*out*::println);  
  
 // Отримати 2 товари ціна яких менше 1000 (filter, limit)  
 System.*out*.println("\nТовари ціна яких менше 1000:");  
 products.stream()  
 .filter(product -> product.getPrice() < 1000) // Використання гетера  
 .limit(2)  
 .forEach(System.*out*::println);  
  
 // Отримати суму всіх видів товарів, що є на складі (reduce)  
 System.*out*.println("\nСума всіх видів товарів на складі:");  
 int totalCount = products.stream()  
 .map(Product::getCount) // Використання гетера  
 .reduce(0, Integer::*sum*);  
 System.*out*.println(totalCount);  
  
 // Згрупувати товари по бренду (Collectors.groupingBy())  
 System.*out*.println("\nТовари згруповані по бренду:");  
 Map<String, List<Product>> groupedByBrand = products.stream()  
 .collect(Collectors.*groupingBy*(Product::getBrand)); // Використання гетера  
 groupedByBrand.forEach((brand, productList) -> {  
 System.*out*.println("Бренд: " + brand);  
 productList.forEach(System.*out*::println);  
 });  
  
 // Відсортувати товари за зростанням ціни та повернути масив (sorted, Collectors)  
 System.*out*.println("\nТовари, відсортовані за зростанням ціни:");  
 List<Product> sortedByPrice = products.stream()  
 .sorted(Comparator.*comparingDouble*(Product::getPrice)) // Використання гетера  
 .collect(Collectors.*toList*());  
 sortedByPrice.forEach(System.*out*::println);  
  
 // Додатково: знайти найдорожчий товар  
 System.*out*.println("\nНайдорожчий товар:");  
 products.stream()  
 .max(Comparator.*comparingDouble*(Product::getPrice)) // Використання гетера  
 .ifPresent(System.*out*::println);  
 }  
}

***Завдання 6.*** Використання Optional та його методів.

Знайти максимальне значення з масиву чисел, в іншому випадку

повернути рядок «Числа відсутні».



**Лабораторна робота №9**

**Тема:** Регулярні вирази. Рефлексія. Анотації.

**Мета роботи:** практика роботи з регулярними виразами, використання

рефлексії, створення власних анотацій.

**Завдання на лабораторну роботу**

***Завдання 1.*** Створити консольний Java проект java\_lab\_9 з пакетом

com.education.ztu.

***Завдання 2.*** Робота з регулярними виразами:

• Використати власний текст, що містить дані 5-10 співробітників компанії

(ПІБ, вік, посада, досвід роботи, адреса, емайл, телефон і т. д.)

• Знайти в тексті всі номери телефонів та емайли.

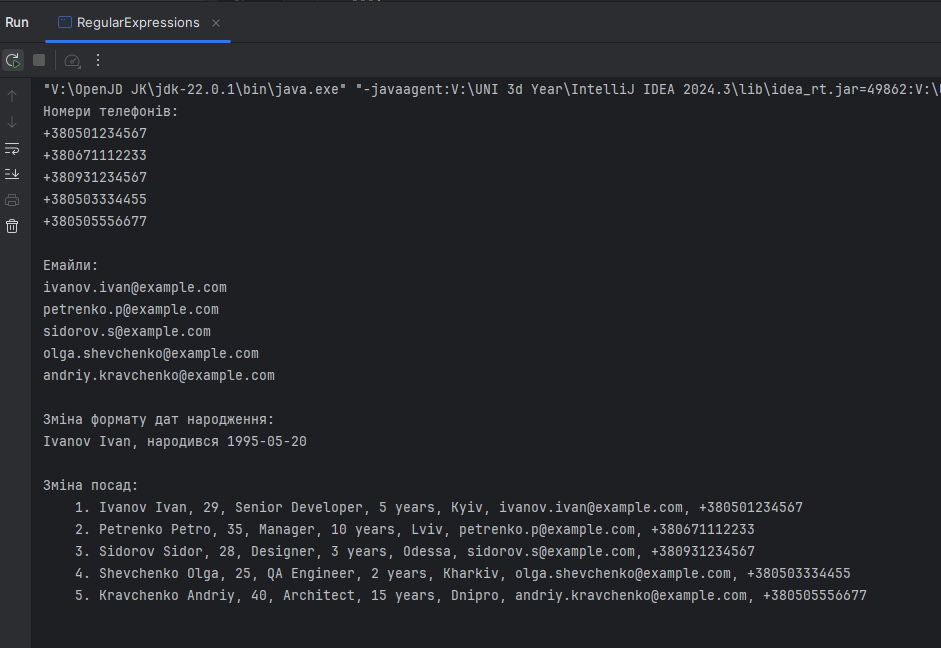
• Змінити формати відображення дат народження (наприклад: 20.05.1995

на 1995-05-20)

• Змінити посади кільком співробітникам.

• Результати роботи відобразити в консолі.

package com.education.ztu;  
  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class RegularExpressions {  
 public static void main(String[] args) {  
 // Вхідний текст із даними співробітників  
 String employees = """  
 1. Ivanov Ivan, 29, Developer, 5 years, Kyiv, ivanov.ivan@example.com, +380501234567  
 2. Petrenko Petro, 35, Manager, 10 years, Lviv, petrenko.p@example.com, +380671112233  
 3. Sidorov Sidor, 28, Designer, 3 years, Odessa, sidorov.s@example.com, +380931234567  
 4. Shevchenko Olga, 25, Tester, 2 years, Kharkiv, olga.shevchenko@example.com, +380503334455  
 5. Kravchenko Andriy, 40, Architect, 15 years, Dnipro, andriy.kravchenko@example.com, +380505556677  
 """;  
  
 // Знаходимо номери телефонів  
 System.*out*.println("Номери телефонів:");  
 Pattern phonePattern = Pattern.*compile*("\\+380\\d{9}");  
 Matcher phoneMatcher = phonePattern.matcher(employees);  
 while (phoneMatcher.find()) {  
 System.*out*.println(phoneMatcher.group());  
 }  
  
 // Знаходимо емейли  
 System.*out*.println("\nЕмайли:");  
 Pattern emailPattern = Pattern.*compile*("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}");  
 Matcher emailMatcher = emailPattern.matcher(employees);  
 while (emailMatcher.find()) {  
 System.*out*.println(emailMatcher.group());  
 }  
  
 // Змінюємо формат дат народження (приклад: замінюємо вигадану дату)  
 System.*out*.println("\nЗміна формату дат народження:");  
 String textWithDates = "Ivanov Ivan, народився 20.05.1995";  
 String updatedDates = textWithDates.replaceAll("(\\d{2})\\.(\\d{2})\\.(\\d{4})", "$3-$2-$1");  
 System.*out*.println(updatedDates);  
  
 // Змінюємо посади кільком співробітникам  
 System.*out*.println("\nЗміна посад:");  
 String updatedPositions = employees.replaceAll("Developer", "Senior Developer")  
 .replaceAll("Tester", "QA Engineer");  
 System.*out*.println(updatedPositions);  
 }  
}



***Завдання 3***. Робота з користувацьким класом методами Reflection

API:

• Створити власний клас в якому міститимуться публічні та приватні поля,

коструктори і методи з аргументами та без.

• Отримати об'єкт класу Class для користувацького класу трьома

способами.

• Отримати всі поля, методи, конструктори, що визначені тільки в цьому

класі (не враховувати наслідування) та вивести ці значення в консоль

(назву, типи параметрів та значення, що повертається).

• Створити екземпляр класу.

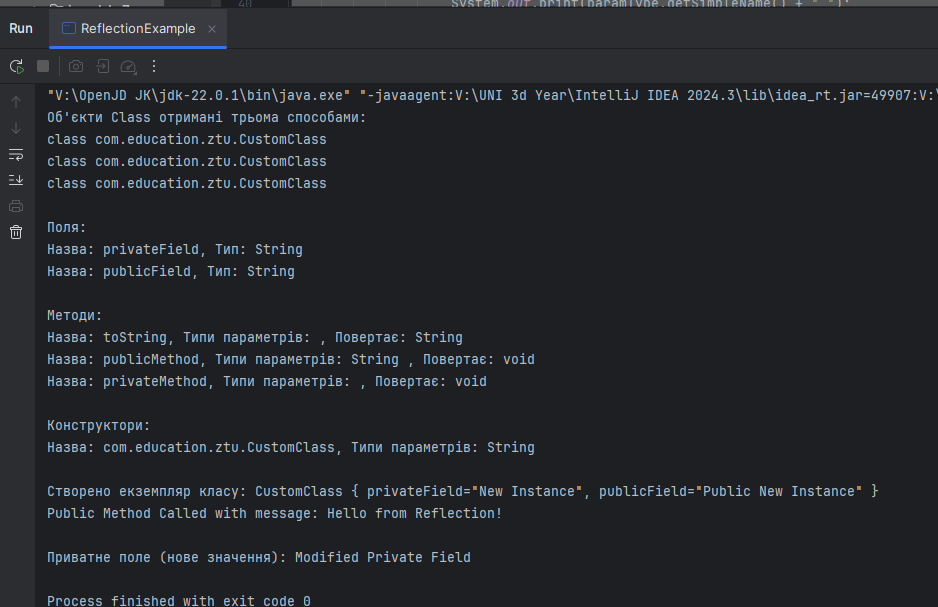
• Викликати метод класу.

• Попрацювати з приватним полем (встановити та отримати значення)

• Результати роботи відобразити в консолі.

package com.education.ztu;  
import java.lang.reflect.\*;  
public class ReflectionExample {  
 public static void main(String[] args) {  
 try {  
 //три способи  
 Class<?> customClass1 = CustomClass.class;  
 Class<?> customClass2 = Class.*forName*("com.education.ztu.CustomClass");  
 CustomClass instance = new CustomClass("Example");  
 Class<?> customClass3 = instance.getClass();  
  
 System.*out*.println("Об'єкти Class отримані трьома способами:");  
 System.*out*.println(customClass1);  
 System.*out*.println(customClass2);  
 System.*out*.println(customClass3);  
  
 System.*out*.println("\nПоля:");  
 Field[] fields = customClass1.getDeclaredFields();  
 for (Field field : fields) {  
 System.*out*.println("Назва: " + field.getName() + ", Тип: " + field.getType().getSimpleName());  
 }  
  
 System.*out*.println("\nМетоди:");  
 Method[] methods = customClass1.getDeclaredMethods();  
 for (Method method : methods) {  
 System.*out*.print("Назва: " + method.getName() + ", Типи параметрів: ");  
 Class<?>[] parameterTypes = method.getParameterTypes();  
 for (Class<?> paramType : parameterTypes) {  
 System.*out*.print(paramType.getSimpleName() + " ");  
 }  
 System.*out*.println(", Повертає: " + method.getReturnType().getSimpleName());  
 }  
  
 System.*out*.println("\nКонструктори:");  
 Constructor<?>[] constructors = customClass1.getDeclaredConstructors();  
 for (Constructor<?> constructor : constructors) {  
 System.*out*.print("Назва: " + constructor.getName() + ", Типи параметрів: ");  
 Class<?>[] parameterTypes = constructor.getParameterTypes();  
 for (Class<?> paramType : parameterTypes) {  
 System.*out*.print(paramType.getSimpleName() + " ");  
 }  
 System.*out*.println();  
 }  
  
 Constructor<?> constructor = customClass1.getDeclaredConstructor(String.class);  
 Object customInstance = constructor.newInstance("New Instance");  
 System.*out*.println("\nСтворено екземпляр класу: " + customInstance);  
  
 Method method = customClass1.getDeclaredMethod("publicMethod", String.class);  
 method.invoke(customInstance, "Hello from Reflection!");  
  
 Field privateField = customClass1.getDeclaredField("privateField");  
 privateField.setAccessible(true);  
 privateField.set(customInstance, "Modified Private Field");  
 System.*out*.println("\nПриватне поле (нове значення): " + privateField.get(customInstance));  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

package com.education.ztu;  
  
class CustomClass {  
 private String privateField;  
 public String publicField;  
  
 public CustomClass(String value) {  
 this.privateField = value;  
 this.publicField = "Public " + value;  
 }  
  
 public void publicMethod(String message) {  
 System.*out*.println("Public Method Called with message: " + message);  
 }  
  
 private void privateMethod() {  
 System.*out*.println("Private Method Called");  
 }  
  
 @Override  
 public String toString() {  
 return "CustomClass { privateField=\"" + privateField + "\", publicField=\"" + publicField + "\" }";  
 }  
}



***Завдання 4***. Створення власної анотації:

• Створити власну анотацію, задати їй необхідні поля та значення за

замовчуванням для них.

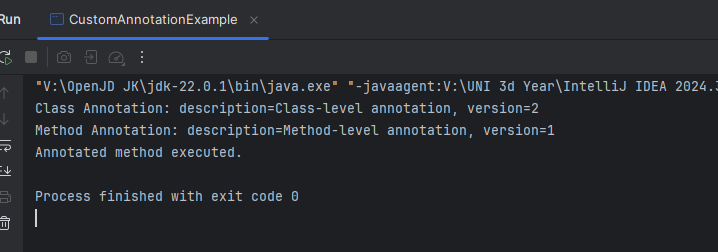
• Встановити їй обмеження застосування через анотацію @Target

• Встановити їй політику утримання через анотацію @Retention

• Додати анотацію до відповідного об'єкту в коді.

• Отримати дані анотації з об'єкту та вивести в консоль.

package com.education.ztu;  
  
import java.lang.annotation.\*;  
import java.lang.reflect.\*;  
  
// анотація(власна)  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target({ElementType.*TYPE*, ElementType.*METHOD*})  
@interface CustomAnnotation {  
 String description() default "Default description";  
 int version() default 1;  
}  
  
// клас з анотацією  
@CustomAnnotation(description = "Class-level annotation", version = 2)  
public class CustomAnnotationExample {  
  
 @CustomAnnotation(description = "Method-level annotation", version = 1)  
 public void annotatedMethod() {  
 System.*out*.println("Annotated method executed.");  
 }  
  
 public static void main(String[] args) {  
 try {  
 // отримуємо анотацію з класу  
 Class<?> clazz = CustomAnnotationExample.class;  
 if (clazz.isAnnotationPresent(CustomAnnotation.class)) {  
 CustomAnnotation annotation = clazz.getAnnotation(CustomAnnotation.class);  
 System.*out*.println("Class Annotation: description=" + annotation.description() + ", version=" + annotation.version());  
 }  
  
 // з методу  
 Method method = clazz.getDeclaredMethod("annotatedMethod");  
 if (method.isAnnotationPresent(CustomAnnotation.class)) {  
 CustomAnnotation annotation = method.getAnnotation(CustomAnnotation.class);  
 System.*out*.println("Method Annotation: description=" + annotation.description() + ", version=" + annotation.version());  
 }  
  
 // викликаємо анотований метод  
 CustomAnnotationExample instance = new CustomAnnotationExample();  
 method.invoke(instance);  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}



**Лабораторна робота №10**

**Тема:** Серіалізація. Логування. Документування коду.

XML та JSON парсери.

**Мета роботи:** практика роботи з XML та JSON парсерами, використання

серіалізації, логування та докусентування коду.

**Завдання на лабораторну роботу**

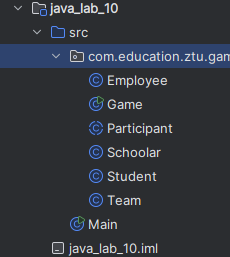
***Завдання 1.*** Створити maven Java проект java\_lab\_10 з пакетом

com.education.ztu. Додати в проект код з лабораторної роботи №3 з пакету

game. Для реалізації завдань додати необхідні залежності в файл

pom.xml.

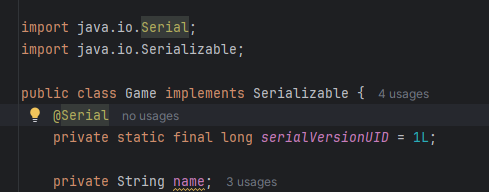
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.education.ztu</groupId>  
 <artifactId>java\_lab\_10</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>8</maven.compiler.source>  
 <maven.compiler.target>8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 5 for testing -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.0</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- SLF4J for logging -->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>2.0.7</version>  
 </dependency>  
  
 <!-- Simple implementation for SLF4J -->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>2.0.7</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Compiler Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.11.0</version>  
 <configuration>  
 <source>8</source>  
 <target>8</target>  
 </configuration>  
 </plugin>  
  
 <!-- Maven Exec Plugin for running the project -->  
 <plugin>  
 <groupId>org.codehaus.mojo</groupId>  
 <artifactId>exec-maven-plugin</artifactId>  
 <version>3.1.0</version>  
 <configuration>  
 <mainClass>com.education.ztu.Main</mainClass>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

******

***Завдання 2.*** Серіалізація:

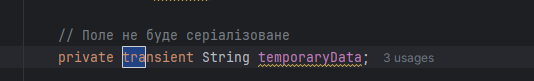
• Додати до сутностей в пакеті game serialVersionUID (згенерувати за

допомогою Inteliij IDEA)

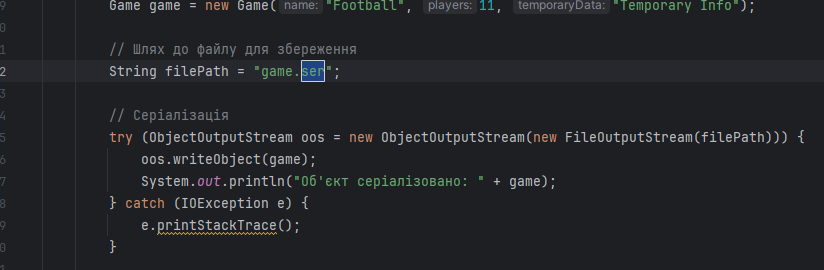


• Виключити деякі поля з серіалізації на власний розсуд (використати

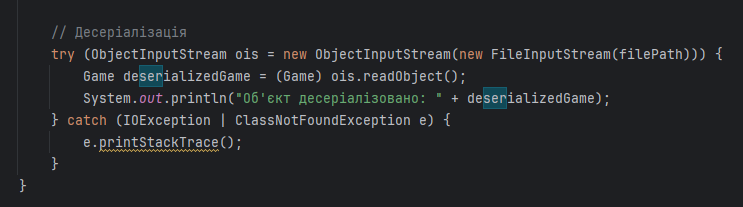
ключове слово transient)

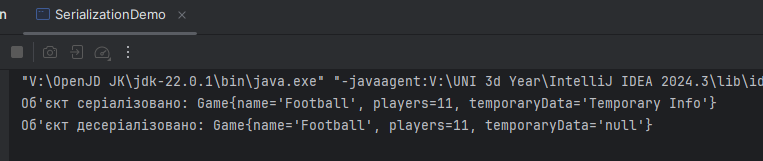


• Серіалізувати

та

десеріалізувати сутності



******

***Завдання 3.*** Логування:

• Додати логування до коду в пакеті game. Використати бібліотеки Log4J,

SLF4J.



• Вивести логи в консоль та в файл.

• Використати різні рівні логування (trace, debug, info, warn, error, fatal)

***Завдання 4.*** Документування коду:

• Додати документаційні коментарі до коду в пакеті game

• Згенерувати документацію (щоб згенерувати JavaDoc у Inteliij IDEA

необхідно натиснути Tools –> Generate JavaDoc –> вказати шлях, куди

зберегти документацію)

***Завдання 5.*** XML парсери:

• Реалізувати читання та збереження XML файлу використовуючи DOM

парсер.

• XML файл використати будь який за бажанням.

Завдання 6. JSON парсер:

• Провести перетворення сутностей з Java в JSON і навпаки з JSON в

Java (використайте бібліотеки Gson або Jackson) Сутності для

перетворень виберіть на власний розсуд.