Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій
Кафедра автоматизованих систем управління

**Звіт**

до лабораторної роботи № 3

з дисципліни

*Прикладне програмування*

на тему:

# “Гра «Битва дроїдів»”

Виконав: студент КН-105
**Руденко Руслан**

Прийняв:
**Мельник Р.В.**

Львів – 2022

# *Лабораторна робота № 3*

## **Тема роботи:** “**Гра «Битва дроїдів»**”

**Мета роботи:** ознайомитись з основними компонентами інтегрованого середовища програмування мовою Java; навчитись створювати, відлагоджувати і виконувати програмні проекти;

## **Завдання лабораторної роботи:**

1. Створіть базовий клас Droid, від якого будуть походити інші підкласи (види дроїдів), які
будуть відрізнятися різними характеристиками. Мінімальний набір характеристик: name,
health, damage.
2. Додайте можливість різних видів бою: 1 на 1, або команда на команду.
3. Класи потрібно грамотно розкласти по пакетах.
4. У програмі має бути консольне меню. Мінімальний набір команд:
– створити дроїда (обраного виду);
– показати список створених дроїдів;
– запустити бій 1 на 1 (вибрати дроїдів, які будуть змагатися);
– запустити бій команда на команду (сформувати команди суперників з дроїдів, яких
ви створили у першому пункті);
– записати проведений бій у файл;
– відтворити проведений бій зі збереженого файлу;
– вийти з програми.

## Текст програми:

**Arena.java**

```java
package lab3.arena;

public interface Arena {
    void simulateBattle(boolean record);
}
```

**OneToOneArena.java**

```java
package lab3.arena;

import lab3.droid.Droid;
import lab3.output.Output;
import lab3.writer.ArenaIO;

import java.io.Serializable;

public class OneToOneArena implements Arena, Serializable {
    Droid firstDroid;
    Droid secondDroid;

    public OneToOneArena(Droid firstDroid, Droid secondDroid)
{
        this.firstDroid = firstDroid;
        this.secondDroid = secondDroid;
    }

    @Override
    public void simulateBattle(boolean record) {
        if (record) ArenaIO.write(this, "record.obj");

        int roundCount = 0;
        while (firstDroid.isAlive() && secondDroid.isAlive())
{
            secondDroid.receiveAttack(firstDroid.attack());

            if (!secondDroid.isAlive()) break;

            System.out.print(Output.ANSI_BLACK_BACKGROUND);
            firstDroid.receiveAttack(secondDroid.attack());
            System.out.print(Output.ANSI_RESET);

            roundCount++;
            if (roundCount > 100) {
                break;
            }
        }
        printResults();
    }

    private void printResults() {
        if (firstDroid.isAlive() && secondDroid.isAlive()) {
            System.out.println("Draw!");
        } else if (firstDroid.isAlive()) {
            System.out.println(firstDroid.getName() + " droid 
wins!");
        } else if (secondDroid.isAlive()) {
            System.out.println(secondDroid.getName() + " droid 
wins!");
        }
    }
}
```

**TeamArena.java**

```
package lab3.arena;

import lab3.droid.Droid;
import lab3.output.Output;
import lab3.writer.ArenaIO;

import java.io.Serializable;
import java.util.List;
import java.util.Random;

public class TeamArena implements Arena, Serializable {
    final Random random;
    final List<Droid> firstTeam;
    final List<Droid> secondTeam;

    public TeamArena(List<Droid> firstTeam, List<Droid> 
secondTeam) {
        this.firstTeam = firstTeam;
        this.secondTeam = secondTeam;
        random = new Random();
    }

    @Override
    public void simulateBattle(boolean record) {
        if (record) ArenaIO.write(this, "record.obj");

        int roundCount = 0;
        while (!firstTeam.isEmpty() && !secondTeam.isEmpty()) 
{
            System.out.println("First team attacks: ");
            teamAttack(firstTeam, secondTeam);
            System.out.println();

            if (secondTeam.isEmpty()) break;


            System.out.print(Output.ANSI_BLACK_BACKGROUND);
            System.out.println("Second team attacks: ");
            teamAttack(secondTeam, firstTeam);
            System.out.print(Output.ANSI_RESET + "\n");

            roundCount++;
            if (roundCount > 100) {
                break;
            }
        }

        printResults();
    }

    private void teamAttack(List<Droid> attackers, List<Droid>
defenders) {
        for (int i = 0; i < attackers.size(); i++) {
            int target = random.nextInt(defenders.size());


defenders.get(target).receiveAttack(attackers.get(i).attack())
;

            if (!defenders.get(target).isAlive())
defenders.remove(target);
            if (!attackers.get(i).isAlive())
attackers.remove(i);
            if (attackers.isEmpty() || defenders.isEmpty())
break;
        }
    }

    private void printResults() {
        if (!firstTeam.isEmpty() && secondTeam.isEmpty()) {
            System.out.println("First team wins!");
        } else if (!secondTeam.isEmpty() &&
firstTeam.isEmpty()) {
            System.out.println("Second team wins!");
        } else {
            System.out.println("Draw!");
        }
    }
}
```

**Attack.java**

```java
package lab3.droid;

public class Attack {
    Effect effect;
    Integer damage;

    public Attack() {
    }

    public Attack(int damage) {
        this.damage = damage;
    }

    public Attack(Effect effect) {
        this.effect = effect;
    }

    public Attack(Effect effect, int damage) {
        this.effect = effect;
        this.damage = damage;
    }

    @Override
    public String toString() {
        return "(damage: " + damage + ")";
    }
}
```

**Attacker.java**

```java
package lab3.droid;

import java.util.Scanner;

public class Attacker extends HealthDroid {
    private final int damage;

    public Attacker(String name, int health, int defense, int
damage) {
        super(name, defense, health);
        this.damage = damage;
    }

    public Attacker() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter droid damage: ");
        this.damage = input.nextInt();
        System.out.println("Droid created!\n");
    }

    @Override
    public Attack attack() {
        return new Attack(damage);
    }

    @Override
    public boolean isAlive() {
        return health > 0;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Name: " + name + "\nHealth: " + health +
"\nDefence: " + defense + "\ndamage: " + damage;
    }
}
```

**BaseDroid.java**

```java
package lab3.droid;

import java.io.Serializable;
import java.util.Scanner;

abstract class BaseDroid implements Droid, Serializable,
Cloneable {
    String name;

    protected BaseDroid(String name) {
        this.name = name;
    }

    protected BaseDroid() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter droid name: ");
        this.name = input.nextLine();
    }

    @Override
    public String getName() {
        return name;
    }
    public Droid copy() throws CloneNotSupportedException {
        return (Droid) super.clone();
    }
}
```

**Defender.java**

```java
package lab3.droid;

public class Defender extends HealthDroid {
    public Defender(String name, int defence, int health) {
        super(name, defence, health);
    }

    public Defender() {
        System.out.println("Droid created!\n");
    }

    @Override
    public Attack attack() {
        return new Attack();
    }

    @Override
    public String toString() {
        return "Name: " + name + "\nHealth: " + health +
"\nDefence: " + defense;
    }
}
```

**Droid.java**

```java
package lab3.droid;

public interface Droid {
    Attack attack();

    void receiveAttack(Attack attack);

    boolean isAlive();

    String getName();

    Droid copy() throws CloneNotSupportedException;
}
```

**DroidsManager.java**

```java
package lab3.droid;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;

public class DroidsManager {
    private final List<Droid> droids;

    public DroidsManager() {
        droids = new ArrayList<>();
        droids.add(new Attacker("first", 100, 10 ,10));
        droids.add(new Attacker("second", 100, 0 ,20));
        droids.add(new Attacker("third", 100, 0 ,20));
    }

    public List<Droid> getAllDroids() {
        return droids;
    }

    public Droid[] getAttackDroids() {
        return filterDroids((droid) -> droid instanceof
Attacker);
    }

    private Droid[] filterDroids(Predicate<Droid> compare) {
        return
droids.stream().filter(compare).toArray(Droid[]::new);
    }

    public void addDroid(Droid droid) {
        droids.add(droid);
    }

    private void showDroids(Predicate<Droid> compare) {
        Droid[] showDroids = filterDroids(compare);

        if (showDroids.length == 0) {
            System.out.println("No droids!");
        }

        for (int i = 1; i <= showDroids.length; i++) {
            System.out.println("#" + i + "\n" + showDroids[i -
1] + "\n");
        }
    }

    public void showAllDroids() {
        showDroids((droid) -> true);
    }

    public void showAttackerDroids() {
        showDroids((droid) -> droid instanceof Attacker);
    }
}
```

**Effect.java**

```java
package lab3.droid;

import java.io.Serializable;
import java.util.Scanner;

public class Effect implements Serializable {
    protected final int energyConsumption;
    protected final int healthReduction;
    protected final int defenseReduction;
    protected final int energyReduction;

    public Effect(int energyConsumption, int healthReduction,
int defenseReduction, int energyReduction) {
        this.energyConsumption = energyConsumption;
        this.healthReduction = healthReduction;
        this.defenseReduction = defenseReduction;
        this.energyReduction = energyReduction;
    }

    public Effect() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter energy consumption: ");
        this.energyConsumption = input.nextInt();
        System.out.println("Enter health reduction: ");
        this.healthReduction = input.nextInt();
        System.out.println("Enter defense reduction: ");
        this.defenseReduction = input.nextInt();
        System.out.println("Enter energy reduction: ");
        this.energyReduction = input.nextInt();
    }

    @Override
    public String toString() {
        return "(Energy consumption: " + energyConsumption +
                "\nhealth reduction: " + healthReduction +
                "\ndefense reduction: " + defenseReduction +
                "\nenergy reduction: " + defenseReduction +
                ")";
    }
}
```

**EnergyDroid.java**

```java
package lab3.droid;

import java.util.Scanner;

public abstract class EnergyDroid extends BaseDroid {
    protected int energy;
    protected final int initialEnergy;

    protected EnergyDroid(String name, int energy) {
        super(name);
        this.energy = energy;
        this.initialEnergy = energy;
    }

    protected EnergyDroid() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter droid energy: ");
        this.energy = input.nextInt();
        this.initialEnergy = energy;
    }

    @Override
    public boolean isAlive() {
        return energy > 0;
    }
}
```

**Hacker.java**

```java
package lab3.droid;

import lab3.output.Output;

public class Hacker extends EnergyDroid {
    private final Effect effect;

    public Hacker(String name, int energy, Effect effect) {
        super(name, energy);
        this.effect = effect;
    }

    public Hacker() {
        this.effect = new Effect();
    }

    @Override
    public Attack attack() {
        energy -= effect.energyConsumption;
        Attack attack = new Attack();
        if (energy >= 0) {
            attack = new Attack(effect);
        }

        energy = Math.max(energy, 0);
        if (!isAlive()) {
            System.out.println(name + " run out of energy!");
        }

        return attack;
    }

    @Override
    public void receiveAttack(Attack attack) {
        if (attack.effect != null) {
            energy -= attack.effect.energyReduction;

            System.out.println(name + "health: " +
Output.health(energy, initialEnergy) + ", after effect" +
effect + Output.ANSI_RESET);
        } else if (attack.damage != null) {
            System.out.println("Attack missed on " + name);
        }
    }

    @Override
    public String toString() {
        return "Name: " + name + "\nEnergy: \n" + energy +
"\nEffect: \n" + effect;
    }
}
```

**HealthDroid.java**

```java
package lab3.droid;

import lab3.output.Output;

import java.util.Scanner;

public abstract class HealthDroid extends BaseDroid {
    protected int defense;
    protected int health;
    protected final int initialHealth;

    protected HealthDroid(String name, int defense, int
health) {
        super(name);
        this.defense = defense;
        this.health = health;
        this.initialHealth = health;
    }

    protected HealthDroid() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter droid health: ");
        health = input.nextInt();
        initialHealth = health;
        System.out.println("Enter droid defence: ");
        defense = input.nextInt();
    }

    @Override
    public void receiveAttack(Attack attack) {
        if (attack.damage != null) {
            health -= Math.max(0, attack.damage - defense);
            health = Math.max(health, 0);

            double healthPercent = (double) health /
initialHealth * 100.0;
            System.out.println(name + "(defense: " + defense +
") health: " + Output.health(health, healthPercent) + ", after
attack" + attack);
        }

        if (attack.effect != null) {
            health -= attack.effect.healthReduction;
            defense -= attack.effect.defenseReduction;
            health = Math.max(health, 0);
            defense = Math.max(defense, 0);
            double healthPercent = (double) health /
initialHealth * 100.0;
            System.out.println(name + "(defense: " + defense +
") health: " + Output.health(health, healthPercent) + ", after
effect\n" + attack.effect);
        }
    }

    @Override
    public boolean isAlive() {
        return health > 0;
    }
}
```

**Output.java**

```java
package lab3.output;

public class Output {
    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_RED = "\u001B[31m";
    public static final String ANSI_GREEN = "\u001B[32m";
    public static final String ANSI_YELLOW = "\u001B[33m";
    public static final String ANSI_WHITE = "\u001B[97m";

    public static final String ANSI_BLACK_BACKGROUND =
"\u001B[48;2;50;61;61m";

    public static String health(int health, double persent) {
        if (persent >= 80.0)
            return ANSI_GREEN + health + ANSI_WHITE;
        if (persent >= 30.0)
            return ANSI_YELLOW + health + ANSI_WHITE;
        return ANSI_RED + health + ANSI_WHITE;
    }
}
```

**ArenaIO.java**

```
package lab3.writer;

import lab3.arena.Arena;

import java.io.*;

public class ArenaIO {
    public static void write(Arena arena, String fileName) {
        try {
            FileOutputStream fileOut = new
FileOutputStream(fileName);
            ObjectOutputStream objWriter = new
ObjectOutputStream(fileOut);
            objWriter.writeObject(arena);
            objWriter.close();

            System.out.println("Battle was saved to file:
record.obj");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public static Arena read(String fileName) throws
IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(fileName);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Arena arena = (Arena) ois.readObject();
        ois.close();

        System.out.println("Reading battle from file:");
        return arena;
    }
}
```

**Lab3.java**

```
package lab3;

import lab3.arena.Arena;
import lab3.arena.OneToOneArena;
import lab3.arena.TeamArena;
import lab3.droid.*;
import lab3.writer.ArenaIO;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Lab3 {
    DroidsManager manager = new DroidsManager();

    public static void main(String[] args) {
        new Lab3().run();
    }

    void run() {
        Scanner input = new Scanner(System.in);

        while (true) {
            printMainMenu();

            int menuItem = input.nextInt();
            boolean exit = false;

            switch (menuItem) {
                case 1:
                    createDroid();
                    break;
                case 2:
                    showAllDroids();
                    break;
                case 3:
                    playBattleFromFile();
                    break;
                case 4:
                    runOneToOneBattle();
                    break;
                case 5:
                    runTeamBattle();
```

```java
                    break;
                case 6:
                    exit = true;
                    break;
            }

            if (exit) break;
        }
    }

    void createDroid() {
        printDroidMenu();
        System.out.println("Enter droid type number: ");
        int type = new Scanner(System.in).nextInt();
        switch (type) {
            case 1:
                manager.addDroid(new Attacker());
                break;
            case 2:
                manager.addDroid(new Defender());
                break;
            case 3:
                manager.addDroid(new Hacker());
                break;
            default:
                System.out.println("No such droid type!");
        }

    }

    void showAllDroids() {
        manager.showAllDroids();
        System.out.println("press enter to continue..");
        new Scanner(System.in).nextLine();
    }

    void playBattleFromFile() {
        System.out.println("Enter file path: ");
        String filePath = new Scanner(System.in).nextLine();
        try {
            Arena arena = ArenaIO.read(filePath);

            arena.simulateBattle(false);
        } catch (Exception e) {
            System.out.println("File error!");
        }
    }

    void runOneToOneBattle() {
        manager.showAttackerDroids();
        System.out.println("Enter two droid numbers to battle: 
");
        Scanner in = new Scanner(System.in);
        int first = in.nextInt();
        int second = in.nextInt();

        if (first == second) {
            System.out.println("Droids can't be the same");
            return;
        }

        Droid[] droids = manager.getAttackDroids();

        try {
            Arena arena = new OneToOneArena(droids[first - 
1].copy(), droids[second - 1].copy());

            System.out.println("Record battle?(y/n)");
            String record = in.next();

arena.simulateBattle(record.equalsIgnoreCase("y"));
        } catch (Exception e) {
            System.out.println("Invalid data!");
        }
    }

    void runTeamBattle() {
        manager.showAllDroids();
        Scanner in = new Scanner(System.in);
        System.out.println("Enter first team numbers: ");
        int[] firstTeamIndices = 
Arrays.stream(in.nextLine().split(" 
")).mapToInt(Integer::parseInt).toArray();
        System.out.println("Enter second team numbers:");
        int[] secondTeamIndices = 
Arrays.stream(in.nextLine().split(" 
")).mapToInt(Integer::parseInt).toArray();

        List<Droid> firstTeam = new ArrayList<>();
        List<Droid> secondTeam = new ArrayList<>();

        try {
            for (int i : firstTeamIndices) {
                firstTeam.add(manager.getAllDroids().get(i - 
1).copy());
            }

            for (int i : secondTeamIndices) {
                secondTeam.add(manager.getAllDroids().get(i - 
1).copy());
            }
            Arena arena = new TeamArena(firstTeam,
secondTeam);

            System.out.println("Record battle?(y/n)");
            String record = in.next();

arena.simulateBattle(record.equalsIgnoreCase("y"));
        } catch (Exception e) {
            System.out.println("Invalid data!");
        }
    }

    private void printDroidMenu() {
        System.out.println(" 1. Attacker\n 2. Defender\n 3.
Hacker\n");
    }

    private void printMainMenu() {
        System.out.print("===========Menu===========\n" +
                "1. Create droid\n" +
                "2. Show droids\n" +
                "3. Play battle from file\n" +
                "4. Run 1 vs 1 battle\n" +
                "5. Run team battle\n" +
                "6. Exit\n");
    }
}
```

**Результати роботи програми:**

```
===========Menu===========
1. Create droid
2. Show droids
3. Play battle from file
4. Run 1 vs 1 battle
5. Run team battle
6. Exit
4
#1
Name: first
Health: 100
Defence: 10
damage: 10

#2
Name: second
Health: 100
Defence: 0
damage: 20

#3
Name: third
Health: 100
Defence: 0
damage: 20

Enter two droid numbers to battle:
1 2
Record battle?(y/n)
y
Battle was saved to file: record.obj
second(defense: 0) health: 90, after attack(damage: 10)
first(defense: 10) health: 90, after attack(damage: 20)
second(defense: 0) health: 80, after attack(damage: 10)
first(defense: 10) health: 80, after attack(damage: 20)
second(defense: 0) health: 70, after attack(damage: 10)
first(defense: 10) health: 70, after attack(damage: 20)
second(defense: 0) health: 60, after attack(damage: 10)
```

```
first(defense: 10) health: 60, after attack(damage: 20)
second(defense: 0) health: 50, after attack(damage: 10)
first(defense: 10) health: 50, after attack(damage: 20)
second(defense: 0) health: 40, after attack(damage: 10)
first(defense: 10) health: 40, after attack(damage: 20)
second(defense: 0) health: 30, after attack(damage: 10)
first(defense: 10) health: 30, after attack(damage: 20)
second(defense: 0) health: 20, after attack(damage: 10)
first(defense: 10) health: 20, after attack(damage: 20)
second(defense: 0) health: 10, after attack(damage: 10)
first(defense: 10) health: 10, after attack(damage: 20)
second(defense: 0) health: 0, after attack(damage: 10)
first droid wins!
===========Menu===========
1. Create droid
2. Show droids
3. Play battle from file
4. Run 1 vs 1 battle
5. Run team battle
6. Exit
```

## Висновок

В ході лабораторної роботи я ознайомився з основними компонентами інтегрованого середовища **Intellij Idea** для програмування мовою Java. Навчився створювати, відлагоджувати і виконувати програмні проекти, вивчив принципи оголошення класу та об’єктів, рівні захисту елементів класу та застосував ці знання на практиці виконавши індивідуальне завдання.