Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до лабораторної роботи № 1

з дисципліни

​*Прикладне програмування*  ​

на тему:

**“Гра «Битва дроїдів»”**

Виконав: студент гр. ОІ-21Ш

**Пелех Денис**

Прийняв: асистент каф. АСУ

**Чорненький В.Я.**

Львів – 2025

Індивідуальне завдання:

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.

**Код програми**

**Клас Droid.java**

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Абстрактний базовий клас для всіх типів бойових дроїдів.

\* Містить основні характеристики: здоров'я, шкоду, а також систему бафів.

\*/

public abstract class Droid {

protected String name;

protected int maxHealth;

protected int health;

protected int damage;

protected boolean isAlive;

protected List<Buff> activeBuffs;

public Droid(String name, int health, int damage) {

this.name = name;

this.maxHealth = health;

this.health = health;

this.damage = damage;

this.isAlive = true;

this.activeBuffs = new ArrayList<>();

}

// ═══ СИСТЕМА БАФІВ ═══

public int getModifiedDamage(int baseDamage) {

int finalDamage = baseDamage;

for (Buff buff : activeBuffs) {

if (buff.getType() == BuffType.DOUBLE\_DAMAGE) {

finalDamage \*= 2;

} else if (buff.getType() == BuffType.CRITICAL\_CHARGE) {

finalDamage \*= 3;

}

}

return finalDamage;

}

public void takeDamage(int damage) {

int finalDamage = damage;

// Застосування щита

for (Buff buff : activeBuffs) {

if (buff.getType() == BuffType.SHIELD) {

finalDamage = (int)(finalDamage \* 0.5); // 50% зменшення

break;

}

}

this.health -= finalDamage;

if (this.health <= 0) {

this.health = 0;

this.isAlive = false;

}

}

public void heal(int amount) {

this.health += amount;

if (this.health > maxHealth) {

this.health = maxHealth;

}

}

public String addBuff(Buff buff) {

activeBuffs.removeIf(b -> b.getType() == buff.getType());

activeBuffs.add(buff);

if (buff.getType() == BuffType.FULL\_HEAL) {

heal(maxHealth);

return String.format("%s %s отримує %s - здоров'я відновлено до %d!",

Colors.BRIGHT\_GREEN, name, buff.toString(), health);

}

return String.format("%s %s отримує %s!",

Colors.BRIGHT\_CYAN, name, buff.toString());

}

public void updateBuffs() {

activeBuffs.forEach(Buff::decreaseDuration);

activeBuffs.removeIf(buff -> !buff.isActive());

}

public boolean hasBuff(BuffType type) {

return activeBuffs.stream().anyMatch(b -> b.getType() == type);

}

// ═══ АБСТРАКТНІ МЕТОДИ ═══

public abstract String attack(Droid target);

public abstract String getType();

public abstract String getColor();

// ═══ ГЕТТЕРИ ═══

public String getName() { return name; }

public int getHealth() { return health; }

public int getMaxHealth() { return maxHealth; }

public int getDamage() { return damage; }

public boolean isAlive() { return isAlive; }

public List<Buff> getActiveBuffs() { return activeBuffs; }

// ═══ ВІЗУАЛІЗАЦІЯ ═══

public String getHealthBar() {

int barLength = 20;

double healthPercent = (double)health / maxHealth;

int filled = (int)(barLength \* healthPercent);

String color;

if (healthPercent > 0.6) {

color = Colors.BRIGHT\_GREEN;

} else if (healthPercent > 0.3) {

color = Colors.BRIGHT\_YELLOW;

} else {

color = Colors.BRIGHT\_RED;

}

StringBuilder bar = new StringBuilder(color + "[");

for (int i = 0; i < barLength; i++) {

if (i < filled) {

bar.append("█");

} else {

bar.append("░");

}

}

bar.append("]" + Colors.RESET);

return bar.toString();

}

@Override

public String toString() {

StringBuilder result = new StringBuilder();

result.append(String.format("%s%s%s (%s) - Здоров'я: %d/%d, Шкода: %d",

getColor(), name, Colors.RESET, getType(), health, maxHealth, damage));

if (!activeBuffs.isEmpty()) {

result.append(Colors.CYAN).append(" [");

for (int i = 0; i < activeBuffs.size(); i++) {

if (i > 0) result.append(", ");

result.append(activeBuffs.get(i).getType().getIcon());

}

result.append("]").append(Colors.RESET);

}

return result.toString();

}

}

**Клас Enum BuffType.java**

/\*\*

\* Перелік можливих бафів для дроїдів під час бою.

\*/

public enum BuffType {

FULL\_HEAL("Повне зцілення", "💚"), // Повне відновлення HP

DOUBLE\_DAMAGE("Подвійна шкода", "⚔️"), // 2x шкоди

SHIELD("Енергетичний щит", "🛡️"), // -50% шкоди

RAPID\_FIRE("Швидка стрільба", "⚡"), // Атака 2 рази

VAMPIRISM("Вампіризм", "🩸"), // +30% HP від шкоди

CRITICAL\_CHARGE("Критичний заряд", "💥"); // 3x шкоди одноразово

private final String name;

private final String icon;

BuffType(String name, String icon) {

this.name = name;

this.icon = icon;

}

public String getName() { return name; }

public String getIcon() { return icon; }

}

**Клас Buff.java**

/\*\*

\* Клас, що представляє активний баф на дроїді.

\*/

public class Buff {

private BuffType type;

private int duration;

private int initialDuration;

public Buff(BuffType type, int duration) {

this.type = type;

this.duration = duration;

this.initialDuration = duration;

}

public void decreaseDuration() {

if (duration > 0) {

duration--;

}

}

public boolean isActive() {

return duration > 0;

}

public BuffType getType() { return type; }

public int getDuration() { return duration; }

public int getInitialDuration() { return initialDuration; }

@Override

public String toString() {

return String.format("%s %s (%d %s)",

type.getIcon(),

type.getName(),

duration,

duration == 1 ? "раунд" : "раунди");

}

}

**Клас AssaultDroid.java**

public class AssaultDroid extends Droid {

private static final int ASSAULT\_HEALTH = 120;

private static final int ASSAULT\_DAMAGE = 25;

private int criticalChance = 20; // 20% шанс

public AssaultDroid(String name) {

super(name, ASSAULT\_HEALTH, ASSAULT\_DAMAGE);

}

@Override

public String attack(Droid target) {

int actualDamage = damage;

boolean isCritical = Math.random() \* 100 < criticalChance;

if (isCritical) {

actualDamage = (int)(damage \* 1.5);

target.takeDamage(actualDamage);

return String.format("%s наносить КРИТИЧНИЙ УДАР по %s на %d шкоди!",

name, target.getName(), actualDamage);

} else {

target.takeDamage(actualDamage);

return String.format("%s атакує %s на %d шкоди!",

name, target.getName(), actualDamage);

}

}

@Override

public String getType() { return "Штурмовик"; }

@Override

public String getColor() { return Colors.BRIGHT\_YELLOW; }

}

**Клас OneOnOneBattle.java**

import java.util.Random;

public class OneOnOneBattle extends Battle {

private Droid droid1;

private Droid droid2;

private int roundNumber;

private Random random;

private static final int BUFF\_CHANCE = 20; // 20% шанс бафу

public OneOnOneBattle(Droid droid1, Droid droid2) {

super();

this.droid1 = droid1;

this.droid2 = droid2;

this.roundNumber = 0;

this.random = new Random();

}

private void tryApplyRandomBuff() {

if (random.nextInt(100) < BUFF\_CHANCE) {

Droid target = random.nextBoolean() ? droid1 : droid2;

BuffType[] types = BuffType.values();

BuffType randomType = types[random.nextInt(types.length)];

int duration = (randomType == BuffType.FULL\_HEAL) ? 0 : (random.nextInt(2) + 1);

Buff buff = new Buff(randomType, duration);

String message = target.addBuff(buff);

log(message + Colors.RESET);

}

}

@Override

public void start() {

clearLog();

log(Colors.BOLD + Colors.CYAN + "\n⚔️ === БІЙ 1 НА 1 === ⚔️" + Colors.RESET);

log("\n" + droid1.toString());

log(" " + droid1.getHealthBar());

log("\n" + Colors.BOLD + Colors.YELLOW + " ⚡ ПРОТИ ⚡" + Colors.RESET);

log("\n" + droid2.toString());

log(" " + droid2.getHealthBar());

log("\n" + Colors.CYAN + "===================" + Colors.RESET + "\n");

while (droid1.isAlive() && droid2.isAlive()) {

roundNumber++;

log(Colors.BOLD + Colors.PURPLE + "\n➤ Раунд " + roundNumber + Colors.RESET);

log(Colors.PURPLE + "──────────" + Colors.RESET);

// Випадковий баф

tryApplyRandomBuff();

// Атака дроїда 1

if (droid1.isAlive()) {

String attackResult = droid1.attack(droid2);

log(attackResult);

log(" " + droid2.getName() + " " + droid2.getHealthBar() +

" " + droid2.getHealth() + "/" + droid2.getMaxHealth());

}

if (!droid2.isAlive()) {

log("\n" + Colors.RED + "☠️ " + droid2.getName() + " переможений!" + Colors.RESET);

log(Colors.BOLD + Colors.GREEN + "\n🏆 ПЕРЕМОЖЕЦЬ: " + droid1.getName() + "! 🏆" + Colors.RESET);

break;

}

// Атака дроїда 2

if (droid2.isAlive()) {

String attackResult = droid2.attack(droid1);

log(attackResult);

log(" " + droid1.getName() + " " + droid1.getHealthBar() +

" " + droid1.getHealth() + "/" + droid1.getMaxHealth());

}

if (!droid1.isAlive()) {

log("\n" + Colors.RED + "☠️ " + droid1.getName() + " переможений!" + Colors.RESET);

log(Colors.BOLD + Colors.GREEN + "\n🏆 ПЕРЕМОЖЕЦЬ: " + droid2.getName() + "! 🏆" + Colors.RESET);

break;

}

// Оновлення бафів

droid1.updateBuffs();

droid2.updateBuffs();

log("");

}

}

public Droid getDroid1() { return droid1; }

public Droid getDroid2() { return droid2; }

}

**Клас Main.java**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

private static List<Droid> droids = new ArrayList<>();

private static Scanner scanner = new Scanner(System.in);

private static Battle lastBattle = null;

public static void main(String[] args) {

boolean running = true;

while (running) {

printMenu();

int choice = getIntInput("Ваш вибір: ");

switch (choice) {

case 1: createDroid(); break;

case 2: showDroids(); break;

case 3: startOneOnOneBattle(); break;

case 4: startTeamBattle(); break;

case 5: saveBattle(); break;

case 6: replayBattle(); break;

case 7:

running = false;

System.out.println("Вихід з програми. До побачення!");

break;

default:

System.out.println("Невірний вибір. Спробуйте ще раз.");

}

System.out.println();

}

scanner.close();

}

private static void printMenu() {

System.out.println("====== АРЕНА БИТВИ ДРОЇДІВ ======");

System.out.println("1. Створити дроїда");

System.out.println("2. Показати список дроїдів");

System.out.println("3. Почати бій 1 на 1");

System.out.println("4. Почати командний бій");

System.out.println("5. Зберегти останній бій у файл");

System.out.println("6. Відтворити бій з файлу");

System.out.println("7. Вийти");

System.out.println("==================================");

}

private static void createDroid() {

System.out.println("\n" + Colors.BOLD + "Оберіть тип дроїда:" + Colors.RESET);

System.out.println(Colors.BRIGHT\_YELLOW + "1. Штурмовик" + Colors.RESET + " - Критичні удари");

System.out.println(Colors.BRIGHT\_BLUE + "2. Танк" + Colors.RESET + " - Броня");

System.out.println(Colors.BRIGHT\_CYAN + "3. Медик" + Colors.RESET + " - Самолікування");

System.out.println(Colors.BRIGHT\_RED + "4. Снайпер" + Colors.RESET + " - Висока шкода");

System.out.println(Colors.RED + "5. Берсерк" + Colors.RESET + " - Лютість (до 2x шкоди)");

System.out.println(Colors.PURPLE + "6. Ніндзя" + Colors.RESET + " - Ухилення + Сплеш");

System.out.println(Colors.GREEN + "7. Регенератор" + Colors.RESET + " - Регенерація");

int type = getIntInput("Введіть тип: ");

System.out.print("Введіть ім'я дроїда: ");

String name = scanner.nextLine();

Droid droid = null;

switch (type) {

case 1: droid = new AssaultDroid(name); break;

case 2: droid = new TankDroid(name); break;

case 3: droid = new MedicDroid(name); break;

case 4: droid = new SniperDroid(name); break;

case 5: droid = new BerserkerDroid(name); break;

case 6: droid = new NinjaDroid(name); break;

case 7: droid = new RegeneratorDroid(name); break;

default:

System.out.println("Невірний тип. Створюємо Штурмовика.");

droid = new AssaultDroid(name);

}

droids.add(droid);

System.out.println("\n" + Colors.BRIGHT\_GREEN + "✓ Дроїда створено!" + Colors.RESET);

System.out.println(droid.toString());

}

private static void showDroids() {

if (droids.isEmpty()) {

System.out.println("\n" + Colors.YELLOW + "Ще не створено дроїдів." + Colors.RESET);

return;

}

System.out.println("\n" + Colors.BOLD + Colors.CYAN + "=== СПИСОК ДРОЇДІВ ===" + Colors.RESET);

for (int i = 0; i < droids.size(); i++) {

Droid d = droids.get(i);

System.out.println("\n" + Colors.BOLD + (i + 1) + "." + Colors.RESET + " " + d.toString());

System.out.println(" " + d.getHealthBar());

}

}

private static void startOneOnOneBattle() {

if (droids.size() < 2) {

System.out.println("\nПотрібно мінімум 2 дроїди для бою.");

return;

}

showDroids();

int index1 = getIntInput("\nОберіть першого дроїда (номер): ") - 1;

int index2 = getIntInput("Оберіть другого дроїда (номер): ") - 1;

if (index1 < 0 || index1 >= droids.size() ||

index2 < 0 || index2 >= droids.size()) {

System.out.println("Невірний вибір дроїда.");

return;

}

if (index1 == index2) {

System.out.println("Не можна битися з самим собою.");

return;

}

Droid droid1Copy = copyDroid(droids.get(index1));

Droid droid2Copy = copyDroid(droids.get(index2));

lastBattle = new OneOnOneBattle(droid1Copy, droid2Copy);

lastBattle.start();

}

private static Droid copyDroid(Droid original) {

if (original instanceof AssaultDroid) {

return new AssaultDroid(original.getName());

} else if (original instanceof TankDroid) {

return new TankDroid(original.getName());

} else if (original instanceof MedicDroid) {

return new MedicDroid(original.getName());

} else if (original instanceof SniperDroid) {

return new SniperDroid(original.getName());

} else if (original instanceof BerserkerDroid) {

return new BerserkerDroid(original.getName());

} else if (original instanceof NinjaDroid) {

return new NinjaDroid(original.getName());

} else if (original instanceof RegeneratorDroid) {

return new RegeneratorDroid(original.getName());

}

return null;

}

private static int getIntInput(String prompt) {

System.out.print(prompt);

while (!scanner.hasNextInt()) {

System.out.print("Невірне введення. " + prompt);

scanner.next();

}

int value = scanner.nextInt();

scanner.nextLine();

return value;

}

// saveBattle() та replayBattle() використовують BattleManager

private static void saveBattle() {

if (lastBattle == null) {

System.out.println("\nНемає бою для збереження.");

return;

}

System.out.print("Введіть назву файлу: ");

String filename = scanner.nextLine();

BattleManager.saveBattleToFile(lastBattle, filename);

}

private static void replayBattle() {

System.out.print("Введіть назву файлу: ");

String filename = scanner.nextLine();

BattleManager.replayBattleFromFile(filename);

}

}

**Процес гри:**

|  |  |
| --- | --- |
| **A black screen with white text  AI-generated content may be incorrect.** | **A screen shot of a computer  AI-generated content may be incorrect.** |
| **A screen shot of a computer  AI-generated content may be incorrect.** | **A screenshot of a computer  AI-generated content may be incorrect.** |
| **A screenshot of a computer  AI-generated content may be incorrect.** |  |

**Посилання на GitHub:** <https://github.com/Denius88/Educ.git>

**Висновок**: під час виконання лабораторної роботи, я поглибив знання мови програмування Java. Написав функціонуючу гру з своїми фішками.