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 ОІ-24

**Тріщук Максим**

Прийняв: Пелех Ю.М.

Львів – 2024

**Лабораторна робота № 3. Гра «Битва дроїдів»**

Напишіть свою реалізацію консольної гри “Битва дроїдів”. Лабораторна робота розрахована на 3 заняття.

**Мінімальний набір вимог до програми:**

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.

**Додаткове завдання**

**Використайте свою фантазію, щоб гра вийшла цікавою для Вас і для глядача**.  
Жодних обмежень немає.

Приклади того, що можна доробити:

− Дроїди можуть мати специфічні для свого виду характеристики – рівень енергії, регенерація, точність, здатність відновлювати здоров’я членів команди тощо.

− Деякі дроїди можуть мати додаткову зброю (утворюють окрему ієрархію класів).

− Звернути увагу на збалансованість гри (відсутність або низький рівень деяких характеристик дроїдів певного виду компенсуються наявністю або вищим рівнем інших).

− Можна створити арени різних видів, що міняють характеристики бою (наприклад, зменшують точність).

− Бій 1 на 1 може бути реалізований як бій між командами з 1 дроїда або як окремий різновид бою зі своїми правилами.

− При бої команда на команду можна застосовувати різні стратегії бою, наприклад, вибирати випадкового дроїда зі своєї команди і команди суперника, дроїдів по черзі, або атакувати дроїда з найменшим здоров’ям.

− Візуалізувати процес бою, використовуючи псевдографіку, кольори та інші можливості змін вигляду тексту у консольному вікні.

Клас Droid

package droids;  
  
public class Droid {  
 protected String name;  
 protected int health;  
 protected int damage;  
 private String maxHealth;  
  
 public Droid(String name, int health, int damage) {  
 this.name = name;  
 this.health = health;  
 this.damage = damage;  
 this.maxHealth = Integer.toString(health);  
 }  
  
 public boolean isAlive() {  
 return this.health > 0;  
 }  
  
 public int getHit(int damage) {  
  
 this.health -= damage;  
  
 if (health < 0) {  
 health = 0;  
 }  
  
 return damage;  
 }  
  
 public int getDamage() {  
 return this.damage;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
  
  
 public int getHealth() {  
 return this.health;  
 }  
  
 public void setHealth(int health) {  
 this.health = health;  
 }  
  
  
  
 @Override  
 public String toString() {  
 return this.name + " health = [" + this.health + " / " + this.maxHealth + "]";  
 }  
}

Похідні класи :

Demetra

package droids;  
  
  
  
public class Demetra extends Droid {  
  
 public int getRandomNumber(int min, int max) {  
 return (int) ((Math.random() \* (max - min)) + min);  
 }  
  
 private final int healStreng;  
  
 public Demetra(String name, int health, int damage, int healStreng) {  
 super(name, health, damage);  
 this.healStreng = healStreng;  
 }  
  
 @Override  
 public int getHit(int damage) {  
 int actualDamage = damage;  
   
 if (actualDamage < 0) {  
 actualDamage = 0;  
 }  
   
 this.health -= actualDamage;  
  
 if (health < 0) {  
 health = 0;  
 }else {  
 this.getHeald();  
 }  
  
 return actualDamage;  
  
 }  
  
 protected void getHeald() {  
 if (this.health <= 25) {  
 int healChance = 50;  
  
 if (Math.random() < (double) healChance / 100) {  
 System.out.printf("ooOOOooOOOOoOO Healing SPELL (@\_@ )\n");  
 this.health += healStreng;  
 }  
 }  
 }  
   
 @Override  
 public int getDamage() {  
 int actualDamage = this.damage;  
 return actualDamage;  
 }  
}

package droids;

Ares

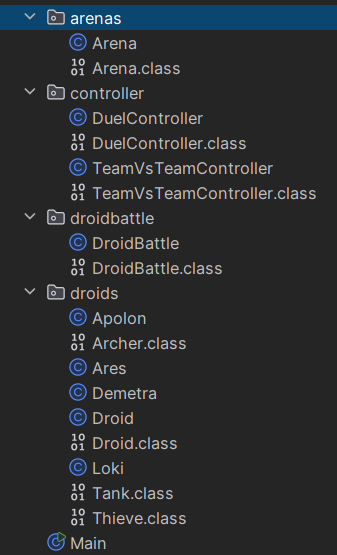
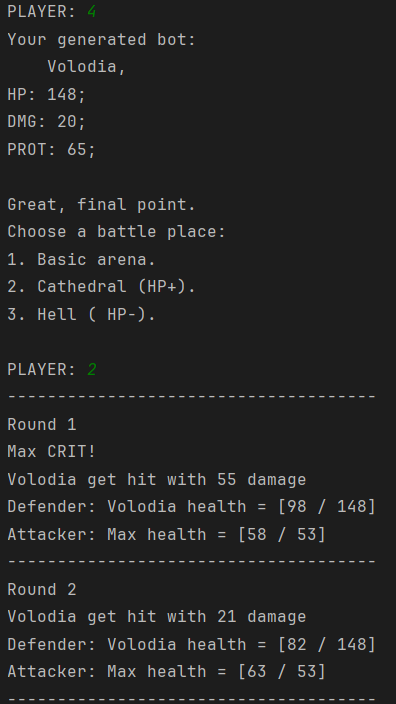
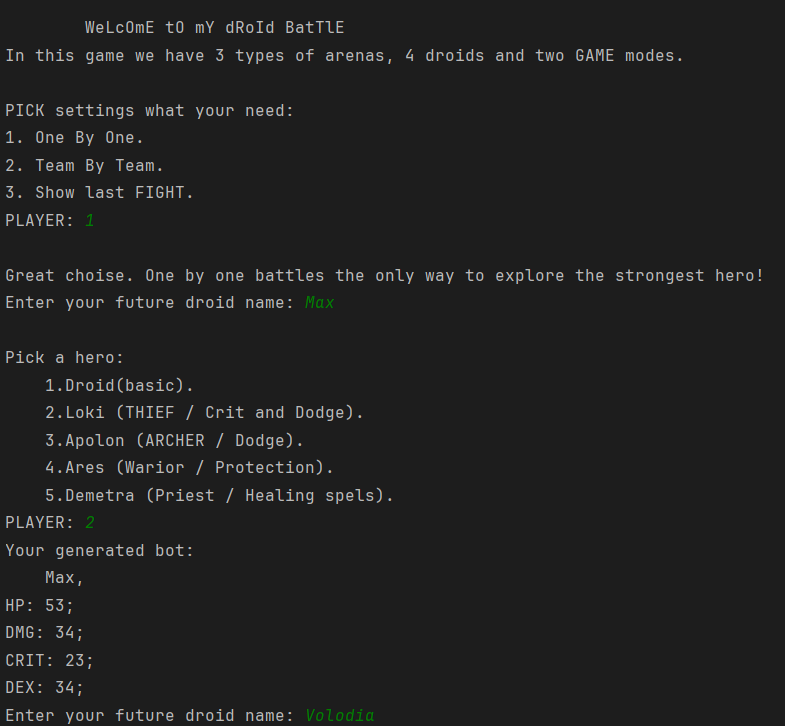
package droids;  
  
  
  
public class Ares extends Droid {  
 private int protection;  
 private final double defKoef = 0.2;  
  
 public Ares(String name, int health, int damage, int protection) {  
 super(name, health, damage);  
 this.protection = protection;  
 }  
  
 public int getHit(int damage) {  
 int actualDamage = damage - (int) (protection \* defKoef);  
  
 if (actualDamage < 0) {  
 actualDamage = 0;  
 }  
  
 this.health -= actualDamage;  
  
 if (health < 0) {  
 health = 0;  
 }  
  
 return actualDamage;  
 }  
  
 public int getDamage() {  
 return this.damage;  
 }  
  
}

Apolon

package droids;  
  
  
public class Apolon extends Droid {  
  
 private int accuracy;  
 private int dexerity;  
  
 public Apolon(String name, int health, int damage, int accuracy, int dexerity) {  
 super(name, health, damage);  
 this.accuracy = accuracy;  
 this.dexerity = dexerity;  
 }  
  
 @Override  
 public int getHit(int damage) {  
 int actualDamage = damage;  
  
 // Damage avoided  
 if (Math.random() < (double) this.dexerity / 100) {  
 System.out.printf("%s WOOW the hit was Dodged !!!!! \n", this.getName());  
 actualDamage = 0;  
 }  
  
 if (actualDamage < 0) {  
 actualDamage = 0;  
 }  
  
 this.health -= actualDamage;  
  
 if (health < 0) {  
 health = 0;  
 }  
  
 return actualDamage;  
 }  
  
 @Override  
 public int getDamage() {  
 int actualDamage = this.damage;  
  
 // Arrow missed  
 if (Math.random() >= (double) this.accuracy / 100) {  
 System.out.printf("%s oh nooooo =( a shot missed a target !!!!!\n", this.getName());  
 actualDamage = 0;  
 }  
 return actualDamage;  
 }  
}

Lokі

public class Loki extends Droid {  
  
 private int critChance;  
 private int dexerity;  
  
 public Loki(String name, int health, int damage, int critChance, int dexerity) {  
 super(name, health, damage);  
 this.critChance = critChance;  
 this.dexerity = dexerity;  
 }  
  
 @Override  
 public int getHit(int damage) {  
 int actualDamage = damage;  
  
 // Damage avoided  
 if (Math.random() < (double) this.dexerity / 100) {  
 System.out.printf("%s godged, WOOW!!\n", this.getName());  
 actualDamage = 0;  
 }  
  
 if (actualDamage < 0) {  
 actualDamage = 0;  
 }  
  
 this.health -= actualDamage;  
  
 if (health < 0) {  
 health = 0;  
 }  
  
 return actualDamage;  
 }  
  
 @Override  
 public int getDamage() {  
 int actualDamage = this.damage;  
  
 // Critical Hit  
 if (Math.random() < (double) this.critChance / 100) {  
 System.out.printf("%s CRIT!\n", this.getName());  
 actualDamage \*= 2;  
 }  
 return actualDamage;  
 }  
}



**Висновок**

Під час виконання цієї лабораторної я застосував усі принципи мови програмування Java а також покращив свої знання про них

https://github.com/itimodzi/Trishchuk\_oi-24\_2024.git