Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина: Программирование

Лабораторная работа по программированию №2

Выполнил: Конаныхина Антонина Александровна
Группа: P3115
Вариант: 311544
Преподаватель: Письмак Алексей Евгеньевич

Санкт-Петербург, 2020г

Задание:

На основе базового класса Pokemon написать свои классы для заданных видов покемонов. Каждый вид покемона должен иметь один или два типа и стандартные базовые характеристики:

- очки здоровья (HP)
- атака (attack)
- защита (defense)
- специальная атака (special attack)
- специальная защита (special defense)
- скорость (speed)

Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых классов PhysicalMove, SpecialMove и StatusMove реализовать свои классы для заданных видов атак.

Атака должна иметь стандартные тип, силу (power) и точность (accuracy). Должны быть реализованы стандартные эффекты атаки. Назначить каждому виду покемонов атаки в соответствии с вариантом. Уровень покемона выбирается минимально необходимым для всех реализованных атак.

Используя класс симуляции боя Battle, создать 2 команды покемонов (каждый покемон должен иметь имя) и запустить бой.

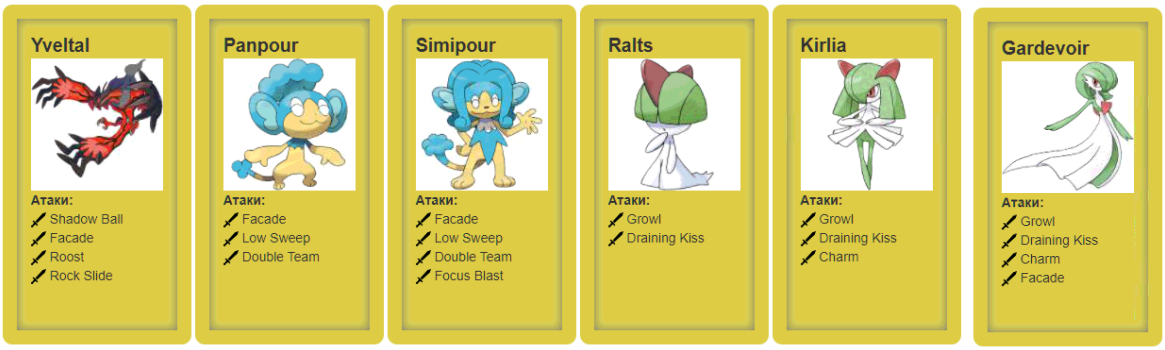
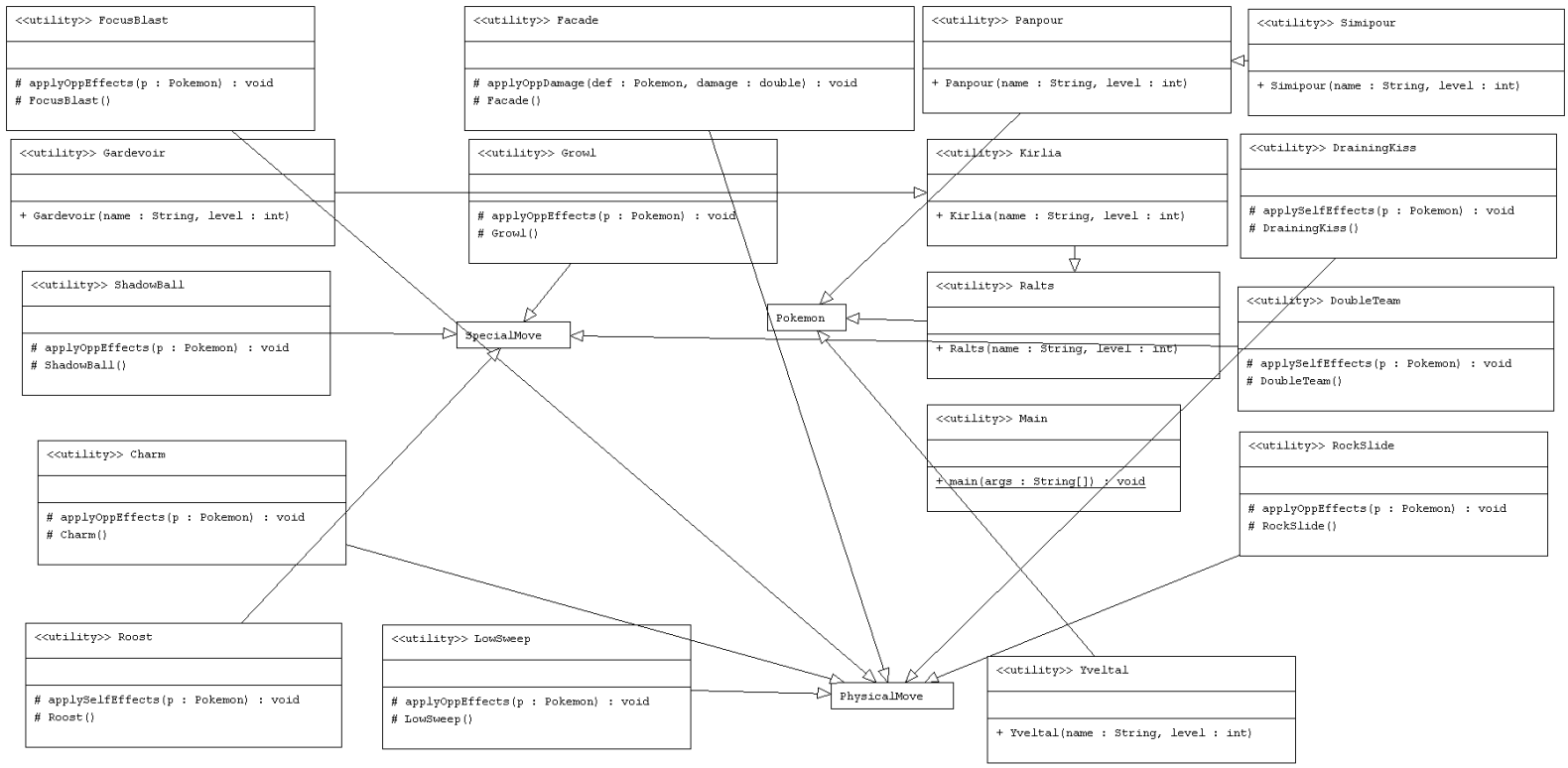


Схема классов:



Программа:

[Main.java:](#)

```
import ru.ifmo.se.pokemon.*;

public class Main {

    public static void main(String[] args) {
        Battle field = new Battle();
        field.addAlly(new Yveltal("A", 10));
        field.addAlly(new Panpour("B", 10));
        field.addAlly(new Simipour("C", 10));
        field.addFoe(new Ralts("D", 10));
        field.addFoe(new Kirlia("E", 10));
        field.addFoe(new Gardevoir("F", 10));
        field.go();
    }
}
```

[Gardevoir.java:](#)

```
import ru.ifmo.se.pokemon.*;

public class Gardevoir extends Kirlia {
    public Gardevoir(String name, int level) {
        super(name, level);
        setStats(68, 65, 65, 125, 115, 80);
```

```
        setType(Type.PSYCHIC, Type.FAIRY);
        setMove(new Growl(), new DrainingKiss(), new Charm(), new Facade());
    }
}
```

Kirlia.java:

```
import ru.ifmo.se.pokemon.*;

public class Kirlia extends Ralts {
    public Kirlia(String name, int level) {
        super(name, level);
        setStats(38, 35, 35, 65, 55, 50);
        setType(Type.PSYCHIC, Type.FAIRY);
        setMove(new Growl(), new DrainingKiss(), new Charm());
    }
}
```

Panpour.java:

```
import ru.ifmo.se.pokemon.*;

public class Panpour extends Pokemon {
    public Panpour(String name, int level) {
        super(name, level);
        setStats(50, 53, 48, 53, 48, 64);
        setType(Type.WATER);
        setMove(new Facade(), new LowSweep(), new DoubleTeam());
    }
}
```

Ralts.java:

```
import ru.ifmo.se.pokemon.*;

public class Ralts extends Pokemon {
    public Ralts(String name, int level) {
        super(name, level);
        setStats(28, 25, 25, 45, 35, 40);
        setType(Type.PSYCHIC, Type.FAIRY);
        setMove(new Growl(), new DrainingKiss());
    }
}
```

Simipour.java:

```
import ru.ifmo.se.pokemon.*;

public class Simipour extends Panpour {
    public Simipour(String name, int level) {
        super(name, level);
        setStats(75, 98, 63, 98, 63, 101);
        setType(Type.WATER);
        setMove(new Facade(), new LowSweep(), new DoubleTeam(), new
FocusBlast());
    }
}
```

Yveltal.java:

```
import ru.ifmo.se.pokemon.*;

public class Yveltal extends Pokemon {
    public Yveltal(String name, int level) {
        super(name, level);
        setStats(126, 131, 95, 131, 98, 99);
        setType(Type.FLYING, Type.DARK);
        setMove(new ShadowBall(), new Facade(), new Roost(), new
RockSlide());
    }
}
```

Attacks.java

```
import ru.ifmo.se.pokemon.*;

import java.awt.*;

class ShadowBall extends SpecialMove{
    protected ShadowBall(){
        super(Type.GHOST, 80, 100);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if (Math.random() <= 0.2) p.setMod(Stat.SPECIAL_DEFENSE, -1);
    }
}
```

```
class Growl extends SpecialMove{
    protected Growl(){
        super(Type.NORMAL, 0, 100);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        p.setMod(Stat.ATTACK, -2);
    }
}


class RockSlide extends PhysicalMove {
    protected RockSlide(){
        super(Type.ROCK, 75, 90);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if (Math.random() <= 0.3) Effect.flinch(p);
    }
}

class FocusBlast extends PhysicalMove {
    protected FocusBlast(){
        super(Type.FIGHTING , 120, 70);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if (Math.random() <= 0.1) p.setMod(Stat.SPECIAL_DEFENSE, -2);
    }
}


class DrainingKiss extends PhysicalMove {
    protected DrainingKiss(){
        super(Type.FAIRY , 50, 100);
    }
    @Override
    protected void applySelfEffects(Pokemon p){
        p.setMod(Stat.HP, (int) Math.round(50*0.75));
    }
}

class Charm extends PhysicalMove {
    protected Charm(){
        super(Type.FAIRY , 0, 100);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        p.setMod(Stat.ATTACK, -2);
    }
}


class Roost extends SpecialMove {
    protected Roost(){
```

```
        super(Type.FLYING, 0, 0);
    }
    @Override
    protected void applySelfEffects(Pokemon p){
        if (p.hasType(Type.FLYING)){
                p.setMod(Stat.HP, (int) Math.round(p.getHP()/2));
        }
    }
}

class DoubleTeam extends SpecialMove {
    protected DoubleTeam(){
        super(Type.NORMAL, 0, 0);
    }
    @Override
    protected void applySelfEffects(Pokemon p){
        p.setMod(Stat.EVASION, +2);
    }
}


class Facade extends PhysicalMove{
    protected Facade(){
        super(Type.NORMAL, 70, 100);
    }
    @Override
    protected void applyOppDamage(Pokemon def, double damage){
        Status PokCon = def.getCondition();
        if (PokCon.equals(Status.BURN) || PokCon.equals(Status.POISON) ||
PokCon.equals(Status.PARALYZE)) {
            def.setMod(Stat.HP, (int) Math.round(damage) * 2);
        }
    }
}

class LowSweep extends PhysicalMove {
    protected LowSweep(){
        super(Type.FIGHTING, 65, 100);
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        p.setMod(Stat.SPEED, -2);
    }
}
```

Результат работы:

Yveltal A из команды полосатых вступает в бой!

Ralts D из команды белых вступает в бой!

Yveltal A атакует.

Ralts D теряет 15 здоровья.

Ralts D теряет сознание.

Kirlia E из команды белых вступает в бой!

Yveltal A атакует.

Kirlia E атакует.

Yveltal A теряет 2 здоровья.

Yveltal A уменьшает атаку.

Yveltal A атакует.

Kirlia E теряет 14 здоровья.

Kirlia E уменьшает специальную защиту.

Kirlia E атакует.

Yveltal A теряет 12 здоровья.

Kirlia E теряет 38 здоровья.

Kirlia E теряет сознание.

Gardevoir F из команды белых вступает в бой!

Yveltal A промахивается

Gardevoir F атакует.

Yveltal A промахивается

Gardevoir F атакует.

Yveltal A атакует.

Gardevoir F восстанавливает 1 здоровья.

Gardevoir F атакует.

Yveltal A промахивается

Gardevoir F атакует.

Yveltal A атакует.

Gardevoir F восстанавливает 1 здоровья.

Gardevoir F атакует.

Yveltal A теряет 3 здоровья.

Yveltal A уменьшает атаку.

Yveltal A атакует.

Gardevoir F атакует.

Yveltal A теряет 2 здоровья.

Yveltal A уменьшает атаку.

Yveltal A теряет сознание.

Panpour B из команды полосатых вступает в бой!

Gardevoir F атакует.

Panpour B теряет 6 здоровья.

Gardevoir F теряет 38 здоровья.

Gardevoir F теряет сознание.

В команде белых не осталось покемонов.

Команда полосатых побеждает в этом бою!

Вывод:

В процессе выполнения лабораторной работы были получены навыки использования объектно-ориентированного подхода программирования при использовании языка Java.