Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Лабораторная работа №2**

Вариант №7346528

Группа: P3132

Выполнил: Волков Г. А.

Проверил:

Письмак Алексей Евгеньевич

г.Санкт-Петербург

2022г.

Оглавление

[Оглавление 2](#_Toc120235336)

[Текст задания 3](#_Toc120235337)

[UML-диаграмма классов 5](#_Toc120235338)

[Исходный код программы 6](#_Toc120235339)

[Результат работы программы 12](#_Toc120235340)

[Вывод 19](#_Toc120235341)

Текст задания

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.

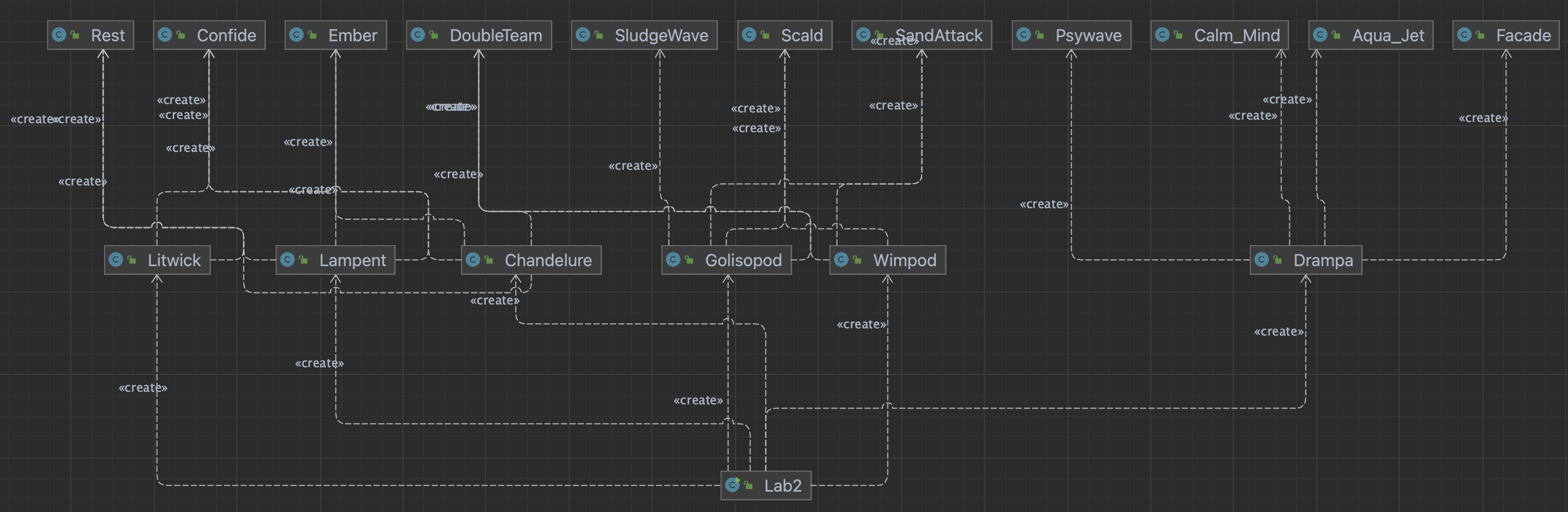
Используя класс симуляции боя Battle, создать 2 команды покемонов (каждый покемон должен иметь имя) и запустить бой.

Базовые классы и симулятор сражения находятся в jar-архиве (обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - здесь.

Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах http://poke-universe.ru, http://pokemondb.net, <http://veekun.com/dex/pokemon>



UML-диаграмма классов



Исходный код программы

Аттаки

**Aqua\_Jet.java**

package moves.physical;  
import ru.ifmo.se.pokemon.\*;  
public class Aqua\_Jet extends PhysicalMove {  
 public Aqua\_Jet() {  
 super(Type.*WATER*, 40, 100);  
 }  
 @Override  
 protected String describe() {  
 return "обрызгал";  
 }  
}

**Calm\_Mind.java**

package moves.status;  
import ru.ifmo.se.pokemon.\*;  
public class Calm\_Mind extends StatusMove{  
 public Calm\_Mind(){  
 super(Type.*PSYCHIC*, 0, 0);  
 }  
 @Override  
 public void applySelfEffects(Pokemon p){  
 p.setMod(Stat.*SPECIAL\_ATTACK*,+1);  
 p.setMod(Stat.*SPECIAL\_DEFENSE*,+1);  
 }  
 @Override  
 protected String describe(){  
 return "собирается с мыслями";  
 }  
}

**Confide.java**

package moves.status;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Confide extends StatusMove {  
 public Confide(){  
 super(Type.*NORMAL*, 0, 0);  
 }  
 @Override  
 public void applyOppEffects(Pokemon p){  
 p.setMod(Stat.*SPECIAL\_ATTACK*, -1);  
 }  
 @Override  
 protected String describe(){  
 return "делает слабее";  
 }  
}

**DoubleTeam.java**

package moves.status;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class DoubleTeam extends StatusMove {  
 public DoubleTeam(){  
 super(Type.*NORMAL*, 0, 0);  
 }  
 @Override  
 public void applySelfEffects(Pokemon p){  
 p.setMod(Stat.*EVASION*, +1);  
 }  
 @Override  
 protected String describe() {  
 return "прокачивает ловкость";  
 }  
}

**Ember.java**

package moves.special;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Ember extends SpecialMove {  
 public Ember(){  
 super(Type.*FIRE*, 40, 100);  
 }  
 private boolean flag;  
 @Override  
 public void applyOppEffects(Pokemon p){  
 if(Math.*random*() < 0.1){  
 Effect.*burn*(p);  
 flag = true;  
 }  
 }  
 @Override  
 protected String describe(){  
 if(flag) return "атакует и поджигает";  
 else return "атакует";  
 }  
  
}

**Façade.java**

package moves.physical;  
import ru.ifmo.se.pokemon.\*;  
public class Facade extends PhysicalMove {  
 public Facade(){  
 super(Type.*NORMAL*, 70, 100);  
 }  
 private boolean flag;  
 @Override  
 public void applyOppDamage(Pokemon poc, double damage){  
 Status cond = poc.getCondition();  
 flag = true;  
 if (cond.equals(Status.*POISON*) || cond.equals(Status.*BURN*) || cond.equals(Status.*PARALYZE*)) {  
 poc.setMod(Stat.*HP*, -2\*(int)Math.*round*(damage));  
 }  
 }  
 @Override  
 protected String describe() {  
 if (flag) return "сильно бьет";  
 else return "бьет";  
 }  
}

**Psywave.java**

package moves.special;  
import ru.ifmo.se.pokemon.\*;  
public class Psywave extends SpecialMove {  
 public Psywave(){  
 super(Type.*PSYCHIC*, 0, 100 );  
 }  
  
 private int lv;  
 @Override  
 public void applySelfEffects(Pokemon p){  
 lv = p.getLevel();  
 }  
 private int dam = (int)Math.*round*(-1\*(Math.*random*()+0.5) \* lv);  
 @Override  
 public void applyOppEffects(Pokemon p){  
 p.setMod(Stat.*HP*, dam);  
 }  
 protected String describe(){  
 return "волнует";  
 }  
  
}

**Rest.java**

package moves.status;  
import ru.ifmo.se.pokemon.\*;  
public class Rest extends StatusMove{  
 public Rest(){  
 super(Type.*PSYCHIC*,0,0);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.restore();  
 Effect restSleep = new Effect().turns(2).attack(0);  
 p.addEffect(restSleep);  
 }  
  
 @Override  
 protected String describe(){  
 return "отдыхает";  
 }  
}

**SandAttack.java**

package moves.status;  
import ru.ifmo.se.pokemon.\*;  
public class SandAttack extends StatusMove {  
 public SandAttack() {  
 super(Type.*GROUND*, 0, 100);  
 }  
 @Override  
 public void applyOppEffects(Pokemon p) {  
  
 p.setMod(Stat.*ACCURACY*, -1);  
 }  
 @Override  
 protected String describe() {  
 return "засыпает песком";  
 }  
}

**Scald.java**

package moves.special;  
import ru.ifmo.se.pokemon.\*;  
public class Scald extends SpecialMove {  
 public Scald() {  
 super(Type.*WATER*, 80, 100);  
 }  
 private boolean flag;  
 @Override  
 public void applyOppEffects(Pokemon p) {  
 Status cond = p.getCondition();  
 if (Math.*random*() < 0.3 && !(p.hasType(Type.*FIRE*))) {  
 Effect.*burn*(p);  
 flag = true;  
 }  
 }  
 @Override  
 protected String describe() {  
 if(flag) return "ошпаривает";  
 else return "обрызгивает";  
 }  
}

**SludgeWave.java**

package moves.special;  
import ru.ifmo.se.pokemon.\*;  
public class SludgeWave extends SpecialMove {  
 public SludgeWave() {  
 super(Type.*POISON*, 95, 100);  
 }  
 private boolean flag;  
 @Override  
 public void applyOppEffects(Pokemon p){  
 if(Math.*random*() < 0.1 & !(p.hasType(Type.*STEEL*)) & !(p.hasType(Type.*POISON*))){  
 flag = true;  
 Effect.*poison*(p);  
 }  
 }  
 @Override  
 protected String describe(){  
 if(flag) return "наносит урон и отравляет";  
 else return "наносит урон";  
 }  
}

**Покемоны**

**Chandelure.java**

package Pokes;  
import ru.ifmo.se.pokemon.\*;  
import moves.\*;  
public class Chandelure extends Pokemon{  
public Chandelure(String name, int level){  
 super(name, level);  
 setStats(60, 55, 90, 145, 90, 80);  
 setType(Type.*GHOST*, Type.*FIRE*);  
 setMove(new Rest(), new Confide(), new Ember(), new DoubleTeam());  
 }  
}

**Drampa.java**

package Pokes;  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
public class Drampa extends Pokemon{  
 public Drampa(String name, int level){  
 super(name, level);  
 setStats(78, 60, 85, 135, 91, 36);  
 setType(Type.*DRAGON*);  
 setMove(new Facade(), new Aqua\_Jet(), new Calm\_Mind(), new Psywave());  
 }  
}

**Golisopod.java**

package Pokes;  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
public class Golisopod extends Pokemon{  
public Golisopod(String name, int level) {  
 super(name, level);  
 setStats(75, 125, 140, 60, 90, 40);  
 setType(Type.*BUG*, Type.*WATER*);  
 setMove(new SandAttack(), new Scald(), new DoubleTeam(), new SludgeWave());  
 }  
}

**Lampent.java**

package Pokes;  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
public class Lampent extends Pokemon{  
public Lampent(String name, int level){  
 super(name, level);  
 setStats(60, 40, 60, 95, 60, 55);  
 setType(Type.*GHOST*, Type.*FIRE*);  
 setMove(new Rest(), new Confide(), new Ember());  
 }  
}

**Litwick.java**

package Pokes;  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
public class Litwick extends Pokemon{  
public Litwick(String name, int level) {  
 super(name, level);  
 setStats(50, 30, 55, 65, 55, 20);  
 setType(Type.*GHOST*, Type.*FIRE*);  
 setMove(new Rest(), new Confide());  
 }  
}

**Wimpod.java**

package Pokes;  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
public class Wimpod extends Pokemon{  
 public Wimpod(String name, int level){  
 super(name, level);  
 setStats(25, 35, 40, 20, 30, 80);  
 setType(Type.*BUG*, Type.*WATER*);  
 setMove(new SandAttack(), new Scald(), new DoubleTeam());  
  
 }  
}

**Main**

**Lab2.java**

import ru.ifmo.se.pokemon.Battle;  
import Pokes.\*;  
public class Lab2 {  
 public static void main(String[] args) {  
 Battle b = new Battle();  
 Drampa p1 = new Drampa("Дракоша Драги", 1);  
 Wimpod p2 = new Wimpod("Устрица", 1);  
 Golisopod p3 = new Golisopod("Дикобраз", 1);  
 Litwick p4 = new Litwick("Свечечка", 1);  
 Lampent p5 = new Lampent("Лампада", 1);  
 Chandelure p6 = new Chandelure("ЛЮСТРА", 1);  
 b.addAlly(p1);  
 b.addAlly(p2);  
 b.addAlly(p3);  
 b.addFoe(p4);  
 b.addFoe(p5);  
 b.addFoe(p6);  
 b.go();  
 }  
}

Результат работы программы

Drampa Дракоша Драги из команды желтых вступает в бой!

Litwick Свечечка из команды зеленых вступает в бой!

Drampa Дракоша Драги бьет.

Litwick Свечечка не замечает воздействие типа NORMAL

Litwick Свечечка промахивается

Drampa Дракоша Драги волнует.

Litwick Свечечка теряет 2 здоровья.

Litwick Свечечка промахивается

Drampa Дракоша Драги сильно бьет.

Litwick Свечечка не замечает воздействие типа NORMAL

Litwick Свечечка промахивается

Drampa Дракоша Драги промахивается

Litwick Свечечка промахивается

Drampa Дракоша Драги промахивается

Litwick Свечечка промахивается

Drampa Дракоша Драги обрызгал.

Litwick Свечечка теряет 10 здоровья.

Litwick Свечечка промахивается

Drampa Дракоша Драги промахивается

Litwick Свечечка промахивается

Drampa Дракоша Драги промахивается

Litwick Свечечка промахивается

Drampa Дракоша Драги обрызгал.

Litwick Свечечка теряет 6 здоровья.

Litwick Свечечка теряет сознание.

Lampent Лампада из команды зеленых вступает в бой!

Lampent Лампада атакует.

Drampa Дракоша Драги теряет 3 здоровья.

Drampa Дракоша Драги промахивается

Lampent Лампада промахивается

Drampa Дракоша Драги обрызгал.

Lampent Лампада теряет 8 здоровья.

Lampent Лампада атакует.

Drampa Дракоша Драги теряет 2 здоровья.

Drampa Дракоша Драги сильно бьет.

Lampent Лампада не замечает воздействие типа NORMAL

Lampent Лампада промахивается

Drampa Дракоша Драги сильно бьет.

Lampent Лампада не замечает воздействие типа NORMAL

Lampent Лампада промахивается

Drampa Дракоша Драги волнует.

Lampent Лампада теряет 3 здоровья.

Lampent Лампада промахивается

Drampa Дракоша Драги волнует.

Lampent Лампада теряет 3 здоровья.

Lampent Лампада теряет сознание.

Drampa Дракоша Драги из команды зеленых вступает в бой!

Drampa Дракоша Драги сильно бьет.

Drampa Дракоша Драги сильно бьет.

Drampa Дракоша Драги сильно бьет.

Drampa Дракоша Драги сильно бьет.

Drampa Дракоша Драги промахивается

Drampa Дракоша Драги промахивается

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 1 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 2 здоровья.

Drampa Дракоша Драги промахивается

Drampa Дракоша Драги промахивается

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 1 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 1 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 2 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 2 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 2 здоровья.

Drampa Дракоша Драги обрызгал.

Drampa Дракоша Драги теряет 2 здоровья.

Оба покемона теряют сознание.

В команде зеленых не осталось покемонов.

Команда желтых побеждает в этом бою!

Вывод

В ходе выполнения лабораторной работы я получил навыки использования

объектно-ориентированного подхода программирования при использовании языка Java.

Научился работать с классами, конструкторами, полями и модификаторами доступа.