Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

факультет Программной инженерии и компьютерных технологий

Лабораторная работа №2

по дисциплине

«Программирование»

Вариант - 21341

Выполнил: Студент группы P3113 Хатиб Ватан

Преподаватель: Райла Мартин.

Санкт-Петербург

2021 г.

# Задание: (21341 вар)

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.

# Pokemons:

#### **Покемоны:**

A picture containing graphical user interface

Description automatically generated

# UML-диаграмма

Diagram

Description automatically generated

# Исходный код программы

Main:

import Pokemons.\*;  
import ru.ifmo.se.pokemon.\*;  
  
  
public class Main {  
  
 public static void main(String[] args) {  
 Battle b = new Battle();  
 Pokemon hitmontop = new Hitmontop("hitmontop",2);  
 Pokemon hondege = new Honedge("hondege",1);  
 Pokemon douBlade = new Doublade("douBlade",2);  
 Pokemon aegiSlash = new Aegislash("aegiSlash",3);  
 Pokemon tapulele = new TapuLele("tapulele",1);  
 Pokemon tyrough = new Tyrogue("tyrough",1);  
 b.addAlly(aegiSlash);  
 b.addAlly(douBlade);  
 b.addAlly(tapulele);  
   
 b.addFoe(tyrough);  
 b.addFoe(hitmontop);  
 b.addFoe(hondege);  
 b.go();  
 }  
}

Moves:

BodySlam:

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class BodySlam extends PhysicalMove {  
 public BodySlam(){  
 super(Type.*NORMAL*,85,100);  
 }  
 boolean flag;  
 @Override  
 protected void applyOppEffects(Pokemon foe){  
 if (Math.*random*() <= 0.3){  
 Effect.*paralyze*(foe);  
 flag=true;  
 }  
  
 }  
 @Override  
 protected String describe(){  
 if (flag){  
 flag=false;  
 return "damage and paralyzed foe ";  
  
 }  
 return "Damage";  
 }  
}

Detect:

package Moves;  
import ru.ifmo.se.pokemon.\*;  
  
public class Detect extends StatusMove {  
 public Detect(){  
 super(Type.*FIGHTING*,0,0,4,1);  
 }  
  
 @Override  
 protected String describe() {  
 return "foe freezed this turn";  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon foe){  
 Effect freezeCurTurn=new Effect().attack(0.0D).turns(0);  
 foe.addEffect(freezeCurTurn);  
 }  
}

Flail :

package Moves;  
  
import ru.ifmo.se.pokemon.\*;

public class Flail extends PhysicalMove {  
 public Flail(){  
 super(Type.*NORMAL*,0,100);  
 }  
  
 @Override  
 protected String describe() {  
 return "deals more damage the lower the user's HP.";  
 }  
  
 @Override  
 protected void applyOppDamage(Pokemon def,double damage){  
 double maxHealth =def.getStat(Stat.*HP*);  
 double currHealth =def.getHP();  
 double n =48\*currHealth/maxHealth;  
 if (n <2){  
 def.setMod(Stat.*HP*, 200);  
 }else if (n<4){  
 def.setMod(Stat.*HP*, 150);  
 }else if (n<9){  
 def.setMod(Stat.*HP*, 100);  
 }else if (n <16){  
 def.setMod(Stat.*HP*, 80);  
 }else if (n <32){  
 def.setMod(Stat.*HP*, 40);  
 }else {  
 def.setMod(Stat.*HP*, 20);  
 }  
 }  
}

Foresight:

package Moves;  
  
import ru.ifmo.se.pokemon.\*;

public class Foresight extends StatusMove {  
 public Foresight(){  
 super(Type.*NORMAL*,0,0);  
 }  
 @Override  
 protected String describe() {  
 return "Resets opponent's Evasiveness";  
 }  
  
 @Override  
 protected boolean checkAccuracy(Pokemon att, Pokemon def){  
 return true;  
 }  
  
}

HyperBeam:

package Moves;  
  
import ru.ifmo.se.pokemon.\*;

public class HyperBeam extends SpecialMove {  
 public HyperBeam(){  
 super(Type.*NORMAL*,150,90);  
 }  
  
 @Override  
 protected String describe() {  
 return "Hyper Beam deals damage, but the user must recharge on the next turn";  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon user){  
 Effect RechargeOnNextGame = new Effect().attack(0.0D).turns(1);  
 user.addEffect(RechargeOnNextGame);  
 }  
}

LeechSeed:

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class LeechSeed extends StatusMove {  
 public LeechSeed(){  
 super(Type.*GRASS*,0,90);  
 }  
  
 @Override  
 protected String describe() {  
 return "steal hp from foe";  
 }  
  
 private Pokemon seeder=null;  
 @Override  
 protected void applySelfEffects(Pokemon self){  
 seeder=self;  
 }  
 @Override  
 protected void applyOppEffects(Pokemon foe){  
 Effect decreaseHp = (new Effect()).turns(-1);  
 double hpLost= foe.getStat(Stat.*HP*) / 8;  
 decreaseHp.stat(Stat.*HP*,(int)hpLost);  
 foe.addEffect(decreaseHp);  
 if (seeder!=null){  
 Effect increaseHp=new Effect().condition(Status.*NORMAL*).turns(-1);  
 increaseHp.stat(Stat.*HP*,(int)-hpLost);  
 seeder.addEffect(increaseHp);  
 }  
 }  
  
}

MegaPunch:

package Moves;  
  
import ru.ifmo.se.pokemon.PhysicalMove;  
import ru.ifmo.se.pokemon.Type;  
  
public class MegaPunch extends PhysicalMove {  
 public MegaPunch(){  
 super(Type.*NORMAL*,80,85);  
 }  
  
 @Override  
 protected String describe() {  
 return "Mega Punch";  
 }  
}

PainSplit:

package Moves;  
  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Stat;  
import ru.ifmo.se.pokemon.StatusMove;  
import ru.ifmo.se.pokemon.Type;  
  
public class PainSplit extends StatusMove {  
 public PainSplit(){  
 super(Type.*NORMAL*,0,0);  
 }  
 Pokemon pokSelf=null;  
 Pokemon pokFoe=null;  
  
  
 @Override  
 protected void applySelfEffects(Pokemon self) {  
 super.applySelfEffects(self);  
 pokSelf=self;  
 if (pokFoe!=null){  
 makeAverge(self,pokFoe);  
 }  
 }  
 protected void makeAverge(Pokemon user, Pokemon foe){  
  
 int average = (int) ((foe.getHP() +user.getHP())/2);  
 if (average >user.getHP()){  
 int num= (int) (average-user.getHP());  
 user.setMod(Stat.*HP*,-num);  
 foe.setMod(Stat.*HP*,num);  
 }else {  
 int num= (int) (user.getHP()-average);  
 user.setMod(Stat.*HP*,num);  
 foe.setMod(Stat.*HP*,-num);  
  
 }  
  
 }  
  
 @Override  
 protected String describe() {  
 return "average health for both foe and user";  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon foe) {  
 super.applyOppEffects(foe);  
 pokFoe=foe;  
 if (pokSelf!=null){  
 makeAverge(pokSelf,foe);  
 }  
 }  
}

SelfDestruct:

package Moves;  
  
import ru.ifmo.se.pokemon.Effect;  
import ru.ifmo.se.pokemon.PhysicalMove;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class SelfDestruct extends PhysicalMove {  
 public SelfDestruct(){  
 super(Type.*NORMAL*,200,100);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon user) {  
 Effect faint = new Effect().attack(0.0D).turns(-1);  
 user.addEffect(faint);  
  
 }  
  
 @Override  
 protected String describe(){  
 return "deals high damage but the user faints";  
 }  
  
}

SmellingSalts:

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class SmellingSalts extends PhysicalMove {  
 public SmellingSalts(){  
 super(Type.*NORMAL*,70,100);  
 }  
 private boolean flag=false;  
 @Override  
 protected void applyOppDamage(Pokemon def, double damage){  
 if (def.getCondition().equals(Status.*PARALYZE*)){  
 flag=true;  
 def.setMod(Stat.*HP*, (int) Math.*round*(damage\*2));  
 }else {  
 def.setMod(Stat.*HP*, (int) Math.*round*(damage));  
 }  
 }  
  
 @Override  
 protected String describe() {  
 if (flag){  
 return "attacked with power 140 , and unpralyzed opponent";  
 }else {  
 return "attack with power 70";  
 }  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon foe){  
 if (foe.getCondition().equals(Status.*PARALYZE*)){  
 flag=true;  
 Effect unParalyze = new Effect().condition(Status.*NORMAL*);  
 foe.setCondition(unParalyze);  
 }  
 }  
}

TriAttack:

package Moves;  
  
import ru.ifmo.se.pokemon.Effect;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.SpecialMove;  
import ru.ifmo.se.pokemon.Type;  
  
public class TriAttack extends SpecialMove {  
 public TriAttack(){  
 super(Type.*NORMAL*,80,100);  
 }  
 private boolean flag;  
  
 @Override  
 protected void applyOppEffects(Pokemon foe) {  
 if (Math.*random*() \* 100 <=20){  
 double chance = Math.*random*() \*100;  
 flag = true;  
 if (chance <= 33){  
 Effect.*paralyze*(foe);  
 }else if (chance <=66){  
 Effect.*burn*(foe);  
 }else if (chance <= 100){  
 Effect.*freeze*(foe);  
 }  
 }  
 }  
 @Override  
 protected String describe(){  
 if (flag){  
 flag=false;  
 return "damage and freeze/burn/paralyze";  
 }else {  
 return "damage";  
 }  
  
 }  
  
}

pokemons:

Aegislash:

package Pokemons;  
  
import Moves.Detect;  
import Moves.Flail;  
import Moves.HyperBeam;  
import Moves.PainSplit;  
import ru.ifmo.se.pokemon.Type;  
  
public class Aegislash extends Doublade {  
 public Aegislash(String name , int level){  
 super(name,level);  
 this.setStats(60,50,140,50,140,60);  
 this.setType(Type.*STEEL*,Type.*GHOST*);  
 this.setMove(new HyperBeam(),new Detect(),new Flail(),new PainSplit());  
 }  
}

Doublade:

package Pokemons;  
  
import Moves.Detect;  
import Moves.Flail;  
import Moves.HyperBeam;  
import ru.ifmo.se.pokemon.Type;  
  
public class Doublade extends Honedge{  
 public Doublade(String name , int level){  
 super(name,level);  
 this.setStats(59,110,150,45,49,35);  
 this.setType(Type.*STEEL*,Type.*GHOST*);  
 this.setMove(new HyperBeam(),new Detect(),new Flail());  
 }  
}

HitmonTop:

package Pokemons;  
  
import Moves.\*;  
import ru.ifmo.se.pokemon.Type;  
  
public class Hitmontop extends Tyrogue {  
 public Hitmontop(String name , int level){  
 super(name,level);  
 this.setStats(50,95,95,35,110,70);  
 this.setType(Type.*FIGHTING*);  
 this.setMove(new HyperBeam(),new SmellingSalts(),new LeechSeed(),new MegaPunch());  
 }  
}

HoneEdge:

package Pokemons;  
  
import Moves.\*;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Honedge extends Pokemon {  
 public Honedge(String name , int level){  
 super(name,level);  
 this.setStats(45,80,100,35,37,28);  
 this.setType(Type.*STEEL*,Type.*GHOST*);  
 this.setMove(new HyperBeam(),new Detect());  
 }  
}

TapuLeta:

package Pokemons;  
  
import Moves.\*;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
  
public class TapuLele extends Pokemon {  
 public TapuLele(String name , int level){  
 super(name,level);  
 this.setStats(70,85,75,130,115,95);  
 this.setType(Type.*PSYCHIC*,Type.*FAIRY*);  
 this.setMove(new TriAttack(),new BodySlam(),new SelfDestruct(),new Foresight());  
 }  
}

Tyrouga:

package Pokemons;  
  
import Moves.\*;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Tyrogue extends Pokemon {  
 public Tyrogue(String name , int level){  
 super(name,level);  
 this.setStats(35,35,35,35,35,35);  
 this.setType(Type.*FIGHTING*);  
 this.setMove(new HyperBeam(), new SmellingSalts(), new LeechSeed());  
 }  
}

# Результат работы программы

1)

Aegislash aegiSlash from the team Blue enters the battle!

Tyrogue tyrough from the team Black enters the battle!

Aegislash aegiSlash deals more damage the lower the user's HP..

Tyrogue tyrough loses 20 hit points.

Tyrogue tyrough faints.

Hitmontop hitmontop from the team Black enters the battle!

Aegislash aegiSlash misses

Hitmontop hitmontop steal hp from foe.

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash Hyper Beam deals damage, but the user must recharge on the next turn.

Hitmontop hitmontop loses 7 hit points.

Hitmontop hitmontop steal hp from foe.

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash loses 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Aegislash aegiSlash misses

Hitmontop hitmontop Mega Punch.

Aegislash aegiSlash loses 1 hit points.

Aegislash aegiSlash isn't affected by NORMAL

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash loses 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Aegislash aegiSlash Hyper Beam deals damage, but the user must recharge on the next turn.

Hitmontop hitmontop loses 6 hit points.

Hitmontop hitmontop attack with power 70.

Aegislash aegiSlash loses 1 hit points.

Aegislash aegiSlash isn't affected by NORMAL

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash loses 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Aegislash aegiSlash misses

Hitmontop hitmontop steal hp from foe.

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash loses 2 hit points.

Aegislash aegiSlash loses 2 hit points.

Doublade douBlade from the team Blue enters the battle!

Doublade douBlade misses

Hitmontop hitmontop Mega Punch.

Doublade douBlade loses 1 hit points.

Doublade douBlade isn't affected by NORMAL

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop steal hp from foe.

Doublade douBlade Hyper Beam deals damage, but the user must recharge on the next turn.

Hitmontop hitmontop loses 5 hit points.

Doublade douBlade loses 1 hit points.

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop restores 1 hit points.

Hitmontop hitmontop steal hp from foe.

Doublade douBlade Hyper Beam deals damage, but the user must recharge on the next turn.

Hitmontop hitmontop loses 7 hit points.

Doublade douBlade loses 1 hit points.

Doublade douBlade loses 1 hit points.

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop restores 2 hit points.

Hitmontop hitmontop restores 1 hit points.

Hitmontop hitmontop restores 1 hit points.

Hitmontop hitmontop attack with power 70.

Doublade douBlade loses 1 hit points.

Doublade douBlade isn't affected by NORMAL

Doublade douBlade deals more damage the lower the user's HP..

Hitmontop hitmontop loses 20 hit points.

Hitmontop hitmontop faints.

Honedge hondege from the team Black enters the battle!

Doublade douBlade deals more damage the lower the user's HP..

Honedge hondege loses 20 hit points.

Honedge hondege isn't affected by NORMAL

Honedge hondege faints.

Team Black loses its last Pokemon.

The team Blue wins the battle!

# Вывод

В процессе выполнения лабораторной работы я получил навыки использования

объектно-ориентированного подхода программирования при использовании языка Java.

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