Санкт-Петербургский Национальный Исследовательский  
Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

По программированию

Вариант 2623

Выполнил:

Студент группы P3106

Рубцов Арсений Дмитриевич

Проверил(а):

Харитонова Анастасия Евгеньевна

Санкт – Петербург 2023

Оглавление

[Текст задания 3](#_Toc147815450)

[Мои покемоны 3](#_Toc147815451)

[UML диаграмма 4](#_Toc147815452)

[Основные этапы вычисления 4](#_Toc147815453)

[Main.java 4](#_Toc147815454)

[Package pokemons 4](#_Toc147815455)

[1) Ninjask 4](#_Toc147815456)

[2) Skitty 4](#_Toc147815457)

[3) Delcatty 5](#_Toc147815458)

[4) Wartortle 5](#_Toc147815459)

[5) Blastoise 5](#_Toc147815460)

[6) Victreebel 5](#_Toc147815461)

[Package attacks 7](#_Toc147815462)

[1) Trick 7](#_Toc147815463)

[2) Thunder Shock 7](#_Toc147815464)

[3) Shadow Punch 7](#_Toc147815465)

[4) Reversal 8](#_Toc147815466)

[5) Peck 8](#_Toc147815467)

[6) Morning Sun 8](#_Toc147815468)

[7) Light Screen 9](#_Toc147815469)

[8) Leer 9](#_Toc147815470)

[9) Hydro Pump 9](#_Toc147815471)

[10) Amnesia 10](#_Toc147815472)

[11) Air Cutter 10](#_Toc147815473)

[Результат работы программы 11](#_Toc147815474)

[Вывод 13](#_Toc147815475)

[Список литературы 14](#_Toc147815476)

# 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.

## Мои покемоны



# [C:\Users\Арсений\Documents\Программирование\Лабораторная работа 2\uml.png](uml.png)UML диаграмма

# Основные этапы вычисления

## Main.java

package program;  
import pokemons.\*;  
import ru.ifmo.se.pokemon.Battle;  
  
public class Main {  
 public static void main(String[] args) {  
 Battle b = new Battle();  
 Ninjask p1 = new Ninjask("ubuntu", 1);  
 Blastoise p2 = new Blastoise("windows", 1);  
 Delcatty p3 = new Delcatty("minty", 1);  
 Skitty p4 = new Skitty("sweety", 1);  
 Victreebel p5 = new Victreebel("iphone", 1);  
 Wartortle p6 = new Wartortle("android", 1);  
 b.addAlly(p1);  
 b.addFoe(p2);  
 b.addAlly(p3);  
 b.addFoe(p4);  
 b.addAlly(p5);  
 b.addFoe(p6);  
 b.go();  
 }  
}

## Package pokemons

1. Ninjask

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Ninjask extends Pokemon {  
 public Ninjask(String name, int level) {  
 super(name, level);  
 setStats(61, 90, 45, 50, 50, 160);  
 setType(Type.*BUG*, Type.*FLYING*);  
 setMove(new Reversal(), new Trick(), new MorningSun(), new Peck());  
 }  
}

1. Skitty

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
у  
public class Skitty extends Pokemon{  
 public Skitty(String name, int level) {  
 super(name, level);  
 setType(Type.*NORMAL*);  
 setStats(70,50, 45, 45, 35, 50);  
 setMove(new ShadowPunch(), new LightScreen(), new HydroPump());  
 }  
}

1. Delcatty

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Delcatty extends Skitty{  
 public Delcatty(String name, int level) {  
 super(name, level);  
 setType(Type.*NORMAL*);  
 setStats(70, 65, 65, 55, 55, 90);  
 setMove(new ShadowPunch(), new LightScreen(), new HydroPump(), new Leer());  
 }  
}

1. Wartortle

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Wartortle extends Pokemon {  
 public Wartortle(String name, int level) {  
 super(name, level);  
 setType(Type.*WATER*);  
 setStats(59, 63, 80, 65, 80, 58);  
 setMove(new ThunderShock(), new Amnesia());  
 }  
}

1. Blastoise

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Blastoise extends Wartortle {  
 public Blastoise(String name, int level) {  
 super(name, level);  
 setType(Type.*WATER*);  
 setStats(79, 83, 100, 85, 105, 78);  
 setMove(new ThunderShock(), new Amnesia(), new AirCutter());  
 }  
}

1. Victreebel

package pokemons;  
import attacks.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Victreebel extends Pokemon {  
 public Victreebel(String name, int level) {  
 super(name, level);  
 setType(Type.*GRASS*, Type.*POISON*);  
 setStats(80, 105, 65, 100, 70, 70);  
 setMove(new ThunderShock(), new Amnesia(), new AirCutter(), new ShadowPunch());  
 }  
}

Package attacks

1. Trick

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class Trick extends SpecialMove {  
 public Trick() {  
 super(Type.*PSYCHIC*, 0, 100);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 *//no such effect :(* }  
 @Override  
 protected String describe() {  
 return "использует Trick";  
 }  
}

1. Thunder Shock

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class ThunderShock extends SpecialMove{  
 public ThunderShock() {  
 super(Type.*ELECTRIC*, 40, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 if (Math.*random*() <= 0.1)  
 Effect.*paralyze*(p);  
 }  
 @Override  
 protected String describe() {  
 return "использует Thunder Shock";  
 }  
}

1. Shadow Punch

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class ThunderShock extends SpecialMove{  
 public ThunderShock() {  
 super(Type.*ELECTRIC*, 40, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 if (Math.*random*() <= 0.1)  
 Effect.*paralyze*(p);  
 }  
 @Override  
 protected String describe() {  
 return "использует Thunder Shock";  
 }  
}

1. Reversal

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class Reversal extends PhysicalMove {  
 public Reversal() {  
  
 super(Type.*FIGHTING*, 200, 100);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 if (p.getStat(Stat.*HP*) == 1)  
 p.setMod(Stat.*ATTACK*, 6);  
 else if (p.getStat(Stat.*HP*) >= 2 && p.getStat(Stat.*HP*) <= 5)  
 p.setMod(Stat.*ATTACK*, 4);  
 else if (p.getStat(Stat.*HP*) >= 6 && p.getStat(Stat.*HP*) <= 12)  
 p.setMod(Stat.*ATTACK*, 3);  
 else if (p.getStat(Stat.*HP*) >= 13 && p.getStat(Stat.*HP*) <= 21)  
 p.setMod(Stat.*ATTACK*, 2);  
 else if (p.getStat(Stat.*HP*) >= 22 && p.getStat(Stat.*HP*) <= 42)  
 p.setMod(Stat.*ATTACK*, 2);  
 else  
 p.setMod(Stat.*ATTACK*, 1);  
 }  
  
 @Override  
 protected String describe() {  
 return "использует Reversal";  
 }  
}

1. Peck

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class Peck extends PhysicalMove {  
 public Peck() {  
 super(Type.*FLYING*, 35, 100);  
 }  
 @Override  
 protected void applyOppDamage(Pokemon def, double damage) {  
 super.applyOppDamage(def, damage);  
 }  
 @Override  
 protected String describe() {  
 return "использует Peck";  
 }  
}

1. Morning Sun

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class MorningSun extends StatusMove {  
 public MorningSun() {  
  
 super(Type.*NORMAL*, 0, 0);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 Effect e1 = new Effect().turns(-1).stat(Stat.*HP*, -(int) p.getStat(Stat.*HP*));  
 p.addEffect(e1);  
 }  
 @Override  
 protected String describe() {  
 return "использует Morning sun";  
 }  
}

1. Light Screen

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class LightScreen extends StatusMove {  
 public LightScreen() {  
 super(Type.*PSYCHIC*, 0, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 Effect e = new Effect().turns(5).stat(Stat.*SPECIAL\_ATTACK*, -2);  
 p.addEffect(e);  
 }  
 @Override  
 protected String describe() {  
 return "использует Light Screen";  
 }  
}

1. Leer

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class Leer extends StatusMove{  
 public Leer() {  
 super(Type.*NORMAL*, 0, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 Effect e = new Effect().stat(Stat.*DEFENSE*, -1);  
 p.addEffect(e);  
 }  
 @Override  
 protected String describe() {  
 return "использует Leer";  
 }  
}

1. Hydro Pump

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class HydroPump extends SpecialMove{  
 public HydroPump() {  
 super(Type.*WATER*, 110, 50);  
 }  
 @Override  
 protected void applyOppDamage(Pokemon def, double damage) {  
 super.applyOppDamage(def, damage);  
 }  
 @Override  
 protected String describe() {  
 return "использует Hydro Pump";  
 }  
}

1. Amnesia

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class Amnesia extends StatusMove{  
 public Amnesia() {  
 super(Type.*PSYCHIC*, 0, 0);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p){  
 p.setMod(Stat.*SPECIAL\_DEFENSE*, 2);  
 }  
 @Override  
 protected String describe() {  
 return "использует Amnesia";  
 }  
}

1. Air Cutter

package attacks;  
import ru.ifmo.se.pokemon.\*;  
  
public class AirCutter extends SpecialMove {  
 public AirCutter() {  
 super(Type.*FLYING*, 60, 95);  
 }  
 @Override  
 protected double calcCriticalHit(Pokemon att, Pokemon def) {  
 if (Math.*random*() < (att.getStat(Stat.*SPEED*) / 256.0))  
 return 2.0;  
 else  
 return 1.0;  
 }  
 @Override  
 protected String describe() {  
 return "использует Air Cutter";  
 }  
}

# Результат работы программы

Ninjask ubuntu из команды красных вступает в бой!

Blastoise windows из команды синих вступает в бой!

Ninjask ubuntu использует Reversal.

Blastoise windows теряет 9 здоровья.

Ninjask ubuntu увеличивает атаку.

Blastoise windows промахивается

Ninjask ubuntu использует Peck.

Blastoise windows теряет 5 здоровья.

Blastoise windows теряет сознание.

Skitty sweety из команды синих вступает в бой!

Ninjask ubuntu использует Reversal.

Skitty sweety теряет 15 здоровья.

Ninjask ubuntu увеличивает атаку.

Skitty sweety теряет сознание.

Wartortle android из команды синих вступает в бой!

Ninjask ubuntu использует Reversal.

Wartortle android теряет 7 здоровья.

Ninjask ubuntu увеличивает атаку.

Wartortle android использует Thunder Shock.

Ninjask ubuntu теряет 7 здоровья.

Ninjask ubuntu промахивается

Wartortle android промахивается

Ninjask ubuntu использует Trick.

Wartortle android теряет 2 здоровья.

Wartortle android промахивается

Ninjask ubuntu промахивается

Wartortle android использует Thunder Shock.

Ninjask ubuntu теряет 8 здоровья.

Ninjask ubuntu теряет сознание.

Delcatty minty из команды красных вступает в бой!

Delcatty minty использует Shadow Punch.

Wartortle android теряет 3 здоровья.

Wartortle android использует Thunder Shock.

Delcatty minty теряет 4 здоровья.

Delcatty minty парализован

Wartortle android использует Thunder Shock.

Delcatty minty теряет 4 здоровья.

Delcatty minty использует Leer.

Wartortle android использует Thunder Shock.

Delcatty minty теряет 5 здоровья.

Delcatty minty теряет сознание.

Victreebel iphone из команды красных вступает в бой!

Victreebel iphone использует Shadow Punch.

Wartortle android теряет 5 здоровья.

Wartortle android теряет сознание.

В команде синих не осталось покемонов.

Команда красных побеждает в этом бою!

# Вывод

Во время выполнения лабораторной работы я познакомился с такими понятиями, как объекты, классы, наследование. Создал свои классы на основе наследования. Выяснил, как упаковывать множество классов в один jar архив.

# Список литературы

1. <https://se.ifmo.ru/~tony/doc/>
2. <https://books.ifmo.ru/file/pdf/3231.pdf>
3. <https://pokemondb.net/>