**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине

«ПРОГРАММИРОВАНИЕ»

Вариант № 373453

**Выполнил**:

Студент группы P3112

*Файзиев Фаридун Равшанович*

**Преподаватель**:

Гаврилов Антон Валерьевич

Санкт-Петербург, 2022 г.

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:

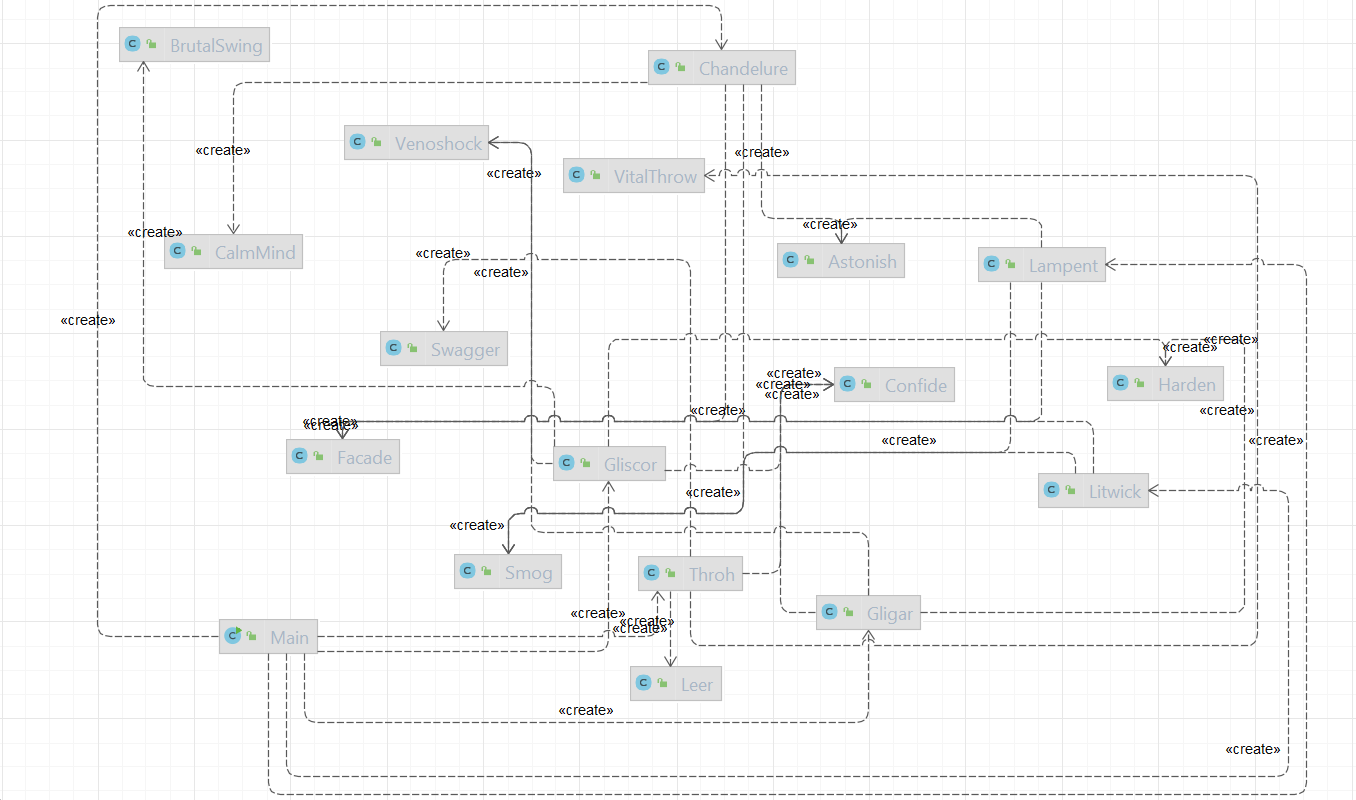
* очки здоровья (HP)
* атака (attack)
* защита (defense)
* специальная атака (special attack)
* специальная защита (special defense)
* скорость (speed)

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.

Покемоны:

UML-диаграмма классов

Код программы:

*Атаки:*

package MyMove;  
  
import Lab.Main;  
import ru.ifmo.se.pokemon.\*;  
  
public class Astonish extends PhysicalMove {  
 public Astonish(double pow, double acc) {  
 super(Type.*GHOST*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon pokemon) {  
 if(Main.*chance*(0.3)) {  
 Effect.*flinch*(pokemon);  
 }  
 }  
  
 @Override  
 protected String describe(){  
 return "Astonish";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class BrutalSwing extends PhysicalMove {  
 public BrutalSwing (double pow, double acc) {  
 super(Type.*DARK*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon pokemon) {  
 Effect effect = new Effect().stat(Stat.*ATTACK*, 1);  
 pokemon.addEffect(effect);  
 super.applyOppEffects(pokemon);  
 }  
  
 @Override  
 protected String describe(){  
 return "BrutalSwing";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class CalmMind extends StatusMove {  
 public CalmMind (double pow, double acc) {  
 super(Type.*PSYCHIC*, pow, acc);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon pokemon) {  
 Effect effect = new Effect().stat(Stat.*SPECIAL\_ATTACK*, +1);  
 pokemon.addEffect(effect);  
  
 Effect effect2 = new Effect().stat(Stat.*SPECIAL\_DEFENSE*, +1);  
 pokemon.addEffect(effect2);  
 }  
  
 @Override  
 protected String describe(){  
 return "CalmMind";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Confide extends StatusMove {  
 public Confide (double pow, double acc) {  
 super(Type.*NORMAL*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 Effect effect = new Effect().stat(Stat.*SPECIAL\_ATTACK*, -1);  
 p.addEffect(effect);  
 }  
  
 @Override  
 protected String describe(){  
 return "Confide";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.PhysicalMove;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Status;  
import ru.ifmo.se.pokemon.Type;  
  
public class Facade extends PhysicalMove {  
 public Facade (double pow, double acc) {  
 super(Type.*NORMAL*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppDamage(Pokemon pokemon, double v) {  
 Status status = pokemon.getCondition();  
  
 super.applyOppDamage(pokemon, 1);  
  
 if(status == Status.*BURN* || status == Status.*POISON* || status == Status.*PARALYZE*) {  
 super.applyOppDamage(pokemon, 140);  
 }  
 }  
  
 @Override  
 protected String describe(){  
 return "Facade";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Harden extends StatusMove {  
 public Harden(double pow, double acc) {  
 super(Type.*NORMAL*, pow, acc);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 Effect effect = new Effect().stat(Stat.*DEFENSE*, 1);  
 p.addEffect(effect);  
 }  
  
 @Override  
 protected String describe(){  
 return "Harden";  
 }  
}

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

package MyMove;  
  
import Lab.Main;  
import ru.ifmo.se.pokemon.\*;  
  
public class Smog extends SpecialMove {  
 public Smog (double pow, double acc) {  
 super(Type.POISON, pow, acc);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon pokemon) {  
 if(Main.chance(0.4)) {  
 Effect.poison(pokemon);  
 }  
 }  
  
 @Override  
 protected String describe(){  
 return "Smog";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Swagger extends StatusMove {  
 public Swagger(double pow, double acc) {  
 super(Type.*NORMAL*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.confuse();  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 Effect effect = new Effect().stat(Stat.*ATTACK*, 2);  
 p.addEffect(effect);  
 }  
  
 @Override  
 protected String describe(){  
 return "Swagger";  
 }  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Venoshock extends SpecialMove {  
 public Venoshock(double pow, double acc) {  
 super(Type.*POISON*, pow, acc);  
 }  
  
 @Override  
 protected void applyOppDamage(Pokemon pokemon, double v) {  
 if (pokemon.getCondition() == Status.*POISON*) {  
 super.applyOppDamage(pokemon, 2);  
 }  
  
 }  
  
 @Override  
 protected String describe(){  
 return "Venoshock";  
 }  
  
  
}

package MyMove;  
  
import ru.ifmo.se.pokemon.PhysicalMove;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class VitalThrow extends PhysicalMove {  
 public VitalThrow(double pow, double acc) {  
 super(Type.*FIGHTING*, pow, acc);  
 }  
 @Override  
 protected void applySelfDamage(Pokemon def, double damage){  
 super.applySelfDamage(def, damage);  
  
 }  
  
 @Override  
 protected String describe(){  
 return "VitalThrow";  
 }  
}

*Покемоны:*

package MyPokemons;  
  
import MyMove.Astonish;  
import MyMove.CalmMind;  
import MyMove.Facade;  
import MyMove.Smog;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Chandelure extends Pokemon {  
 public Chandelure(String name, int level) {  
 super(name, level);  
 super.setType(Type.*GHOST*, Type.*FIRE*);  
 super.setStats(60, 55, 90, 145, 90, 80);  
  
 Smog smog = new Smog(30, 70);  
 Facade facade = new Facade(70, 100);  
 Astonish astonish = new Astonish(30, 100);  
 CalmMind calmMind = new CalmMind(0, 0);  
  
 super.setMove(smog, facade, astonish, calmMind);  
 }  
}

package MyPokemons;  
  
import MyMove.Confide;  
import MyMove.Harden;  
import MyMove.Venoshock;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Gligar extends Pokemon {  
 public Gligar(String name, int level) {  
 super(name, level);  
 super.setType(Type.*GROUND*, Type.*FLYING*);  
 super.setStats(65, 75, 105, 35, 65, 85);  
  
 Harden harden = new Harden(0, 0);  
 Confide confide = new Confide(0, 0);  
 Venoshock venoshock = new Venoshock(65, 100);  
  
 super.setMove(harden, confide, venoshock);  
 }  
}

package MyPokemons;  
  
import MyMove.BrutalSwing;  
import MyMove.Confide;  
import MyMove.Harden;  
import MyMove.Venoshock;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Gliscor extends Pokemon {  
 public Gliscor(String name, int level) {  
 super(name, level);  
 super.setType(Type.*GROUND*, Type.*FLYING*);  
 super.setStats(75, 95, 125, 45, 75, 95);  
  
 Harden harden = new Harden(0, 0);  
 Confide confide = new Confide(0, 0);  
 Venoshock venoshock = new Venoshock(65, 100);  
 BrutalSwing brutalSwing = new BrutalSwing(60, 100);  
  
 super.setMove(harden, confide, venoshock, brutalSwing);  
 }  
}

package MyPokemons;  
  
import MyMove.Astonish;  
import MyMove.Facade;  
import MyMove.Smog;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Lampent extends Pokemon {  
 public Lampent(String name, int level) {  
 super(name, level);  
 super.setType(Type.GHOST, Type.FIRE);  
 super.setStats(60, 40, 60, 95, 60, 55);  
  
 Smog smog = new Smog(30, 70);  
 Facade facade = new Facade(70, 100);  
 Astonish astonish = new Astonish(30, 100);  
  
 super.setMove(smog, facade, astonish);  
 }  
}

package MyPokemons;  
  
import MyMove.\*;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Litwick extends Pokemon {  
 public Litwick(String name, int level) {  
 super(name, level);  
 super.setType(Type.GHOST, Type.FIRE);  
 super.setStats(50, 30, 55, 65, 55, 20);  
  
 Smog smog = new Smog(30, 70);  
 Facade facade = new Facade(70, 100);  
  
 super.setMove(smog, facade);  
 }  
}

package MyPokemons;  
  
import MyMove.Confide;  
import MyMove.Leer;  
import MyMove.Swagger;  
import MyMove.VitalThrow;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Throh extends Pokemon {  
 public Throh(String name, int level) {  
 super(name, level);  
  
 super.setType(Type.FIGHTING);  
 super.setStats(120, 100, 85, 30, 85, 45);  
 Leer leer = new Leer(0, 100);  
 Swagger swagger = new Swagger(0, 85);  
 Confide confide = new Confide(0, 0);  
 VitalThrow vitalThrow = new VitalThrow(70, Double.POSITIVE\_INFINITY);  
  
 super.setMove(leer, swagger, confide, vitalThrow);  
 }  
}

*main.java*

package Lab;  
  
import MyPokemons.\*;  
import ru.ifmo.se.pokemon.Battle;  
  
public class Main {  
 public static void main(String[] args) {  
 Battle b = new Battle();  
 Throh throh = new Throh("Тро", 1);  
 Gligar gligar = new Gligar("Хихик", 1);  
 Gliscor gliscor = new Gliscor("Глискор", 1);  
 Litwick litwick = new Litwick("Литвик", 1);  
 Lampent lampent = new Lampent("Фонарь", 1);  
 Chandelure chandelure = new Chandelure("Люстра", 1);  
  
 b.addAlly(throh);  
 b.addAlly(gliscor);  
 b.addAlly(lampent);  
  
 b.addFoe(gligar);  
 b.addFoe(litwick);  
 b.addFoe(chandelure);  
  
 b.go();  
 }  
  
 public static boolean chance(double c){  
 return (c > Math.*random*());  
 }  
}

*Результат работы программы:*

Registeel Круглоголовый из команды красных вступает в бой!

Dratini Драк из команды белых вступает в бой!

Registeel Круглоголовый усыпляет.

Dratini Драк теряет 1 здоровья.

Dratini Драк засыпает

Dratini Драк флексит.

Registeel Круглоголовый теряет 2 здоровья.

Registeel Круглоголовый увеличивает атаку.

Registeel Круглоголовый парализует.

Dratini Драк теряет 1 здоровья.

Dratini Драк парализован

Dratini Драк бьет.

Критический удар!

Registeel Круглоголовый растерянно попадает по себе.

Registeel Круглоголовый теряет 6 здоровья.

Dratini Драк сильно бьет.

Критический удар!

Registeel Круглоголовый усыпляет.

Dratini Драк теряет 1 здоровья.

Dratini Драк засыпает

Registeel Круглоголовый парализует.

Dratini Драк теряет 1 здоровья.

Dratini Драк парализован

Dratini Драк сильно бьет.

Критический удар!

Dratini Драк флексит.

Критический удар!

Registeel Круглоголовый теряет 3 здоровья.

Registeel Круглоголовый увеличивает атаку.

Registeel Круглоголовый парализует.

Dratini Драк теряет 1 здоровья.

Dratini Драк флексит.

Критический удар!

Registeel Круглоголовый теряет 3 здоровья.

Registeel Круглоголовый увеличивает атаку.

Registeel Круглоголовый адыхает.

Registeel Круглоголовый засыпает

Dratini Драк сильно бьет.

Критический удар!

Registeel Круглоголовый растерянно попадает по себе.

Registeel Круглоголовый теряет 6 здоровья.

Dratini Драк сильно бьет.

Критический удар!

Registeel Круглоголовый толкнул.

Dratini Драк теряет 4 здоровья.

Dratini Драк сильно бьет.

Критический удар!

Registeel Круглоголовый растерянно попадает по себе.

Registeel Круглоголовый теряет 4 здоровья.

Registeel Круглоголовый растерянно попадает по себе.

Registeel Круглоголовый теряет 5 здоровья.

Registeel Круглоголовый теряет сознание.

Mismagius Шляпник из команды красных вступает в бой!

Mismagius Шляпник морозит.

Dratini Драк теряет 4 здоровья.

Dratini Драк флексит.

Критический удар!

Mismagius Шляпник теряет 1 здоровья.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник усилил атаку.

Dratini Драк теряет 2 здоровья.

Dratini Драк теряет сознание.

Dragonite ДракКрутой из команды белых вступает в бой!

Mismagius Шляпник усилил атаку.

Dragonite ДракКрутой теряет 3 здоровья.

Dragonite ДракКрутой дамажит.

Mismagius Шляпник теряет 8 здоровья.

Mismagius Шляпник морозит.

Dragonite ДракКрутой теряет 4 здоровья.

Dragonite ДракКрутой бьет.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник флексит.

Dragonite ДракКрутой теряет 2 здоровья.

Dragonite ДракКрутой увеличивает атаку.

Dragonite ДракКрутой флексит.

Mismagius Шляпник теряет 1 здоровья.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник усилил атаку.

Dragonite ДракКрутой теряет 4 здоровья.

Dragonite ДракКрутой сильно бьет.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник флексит.

Dragonite ДракКрутой теряет 3 здоровья.

Dragonite ДракКрутой увеличивает атаку.

Dragonite ДракКрутой теряет сознание.

Dragonair ДракПокруче из команды белых вступает в бой!

Mismagius Шляпник адыхает.

Dragonair ДракПокруче флексит.

Mismagius Шляпник теряет 1 здоровья.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник морозит.

Dragonair ДракПокруче теряет 4 здоровья.

Dragonair ДракПокруче флексит.

Mismagius Шляпник теряет 1 здоровья.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник адыхает.

Dragonair ДракПокруче бьет.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник флексит.

Dragonair ДракПокруче теряет 3 здоровья.

Dragonair ДракПокруче увеличивает атаку.

Dragonair ДракПокруче флексит.

Mismagius Шляпник теряет 1 здоровья.

Mismagius Шляпник не замечает воздействие типа NORMAL

Mismagius Шляпник усилил атаку.

Критический удар!

Dragonair ДракПокруче теряет 8 здоровья.

Dragonair ДракПокруче теряет сознание.

В команде белых не осталось покемонов.

Команда красных побеждает в этом бою!

Выводы:

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.