Федеральное государственное автономное образовательное учреждение высшего образования национальный исследовательский   
университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки: 09.03.01

Дисциплина «Программирование»

Отчет

По лабораторной работе №2

Вариант №1938626

Студент

Бушмелев Константин Алексеевич,   
группа P3132

Преподаватель

Письмак Алексей Евгеньевич

г. Санкт-Петербург,

2022 г.

Оглавление

[Оглавление 2](#_Toc120269397)

[Задание 2](#_Toc120269398)

[Код программы 2](#_Toc120269399)

[UML диаграмма 2](#_Toc120269400)

[Заключение 2](#_Toc120269401)

Задание

Ознакомиться с документацией, обращая особое внимание на классы Pokemon и Move. При дальнейшем выполнении лабораторной работы читать документацию еще несколько раз.

Скачать файл Pokemon.jar. Его необходимо будет использовать как для компиляции, так и для запуска программы. Распаковывать его не надо! Нужно научиться подключать внешние jar-файлы к своей программе.

Написать минимально работающую программу и посмотреть как она работает.

Battle b = new Battle();

Pokemon p1 = new Pokemon("Чужой", 1);

Pokemon p2 = new Pokemon("Хищник", 1);

b.addAlly(p1);

b.addFoe(p2);

b.go();

Создать один из классов покемонов для своего варианта. Класс должен наследоваться от базового класса Pokemon. В конструкторе нужно будет задать типы покемона и его базовые характеристики. После этого попробуйте добавить покемона в сражение.

Создать один из классов атак для своего варианта (лучше всего начать с физической или специальной атаки). Класс должен наследоваться от класса PhysicalMove или SpecialMove. В конструкторе нужно будет задать тип атаки, ее силу и точность. После этого добавить атаку покемону и проверить ее действие в сражении. Не забудьте переопределить метод describe, чтобы выводилось нужное сообщение.

Если действие атаки отличается от стандартного, например, покемон не промахивается, либо атакующий покемон также получает повреждение, то в классе атаки нужно дополнительно переопределить соответствующие методы (см. документацию). При реализации атак, которые меняют статус покемона (наследники StatusMove), скорее всего придется разобраться с классом Effect. Он позволяет на один или несколько ходов изменить состояние покемона или модификатор его базовых характеристик.

Доделать все необходимые атаки и всех покемонов, распределить покемонов по командам, запустить сражение.

Код программы

import ru.ifmo.se.pokemon.Battle;  
import Pokemons.Castform;  
import Pokemons.Buizel;  
import Pokemons.Floatzel;  
import Pokemons.Porygon;  
import Pokemons.Porygon2;  
import Pokemons.PorygonZ;  
  
public class Main {  
 public static void main(String[] args) {  
 Battle b = new Battle();  
 var castform = new Castform("R2-D2", 10);  
 var buizel = new Buizel("Чубакка", 10);  
 var floatzel = new Floatzel("Йода", 10);  
 var porygon = new Porygon("Дарт Мол", 10);  
 var porygon2 = new Porygon2("Дарт Вейдер", 10);  
 var porygonZ = new PorygonZ("Палпатин", 10);  
 b.addAlly(castform);  
 b.addAlly(buizel);  
 b.addAlly(floatzel);  
 b.addFoe(porygon);  
 b.addFoe(porygon2);  
 b.addFoe(porygonZ);  
 b.go();  
 }  
}

package Pokemons;  
  
import Moves.AquaTail;  
import Moves.Facade;  
import Moves.Growl;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Buizel extends Pokemon{  
 public Buizel(String name, int ...level){  
 super(name, level[0]);  
 setStats(55, 65, 35, 60, 30, 85);  
 setType(Type.*WATER*);  
 var aquatail = new AquaTail();  
 var facade = new Facade();  
 var growl = new Growl();  
 setMove(aquatail, facade, growl);  
 }  
}

package Pokemons;  
import Moves.Blizzard;  
import Moves.PowderSnow;  
import Moves.Thunder;  
import Moves.CosmicPower;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Castform extends Pokemon{  
 public Castform(String name, int level){  
 super(name, level);  
 setStats(70, 70, 70, 70, 70, 70);  
 setType(Type.*NORMAL*);  
 var thunder = new Thunder();  
 var blizzard = new Blizzard();  
 var powdersnow = new PowderSnow();  
 var cosmicpower = new CosmicPower();  
 setMove(thunder, blizzard, powdersnow, cosmicpower);  
 }  
}

package Pokemons;  
  
import Moves.FocusBlast;  
  
public class Floatzel extends Buizel {  
 public Floatzel(String name, int level){  
 super(name, level);  
 setStats(85, 105, 55, 85, 50, 115);  
 var focusblast = new FocusBlast();  
 addMove(focusblast);  
 }  
}

package Pokemons;  
  
import Moves.Blizzard;  
import Moves.IceBeam;  
import ru.ifmo.se.pokemon.Pokemon;  
import ru.ifmo.se.pokemon.Type;  
  
public class Porygon extends Pokemon {  
 public Porygon(String name, int level){  
 super(name, level);  
 setStats(65, 60, 70, 85, 75, 40);  
 setType(Type.*NORMAL*);  
 var icebeam = new IceBeam();  
 var blizzard = new Blizzard();  
 setMove(icebeam, blizzard);  
 }  
}

package Pokemons;  
  
import Moves.DefenseCurl;  
  
public class Porygon2 extends Porygon {  
 public Porygon2(String name, int level){  
 super(name, level);  
 setStats(85, 80, 90, 105, 95, 60);  
 var defensecurl = new DefenseCurl();  
 addMove(defensecurl);  
 }  
}

package Pokemons;  
  
import Moves.Swagger;  
  
  
public class PorygonZ extends Porygon2 {  
 public PorygonZ(String name, int level){  
 super(name, level);  
 setStats(85, 80, 70, 135, 75, 90);  
 var swagger = new Swagger();  
 addMove(swagger);  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class AquaTail extends PhysicalMove {  
 public AquaTail(){  
 super(Type.*WATER*, 90, 90);  
 }  
 @Override  
 public String describe() {  
 return "агрессивно брызгается";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Blizzard extends SpecialMove {  
 public Blizzard(){  
 super(Type.*ICE*, 110, 70);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 if (!pokemon.hasType(Type.*ICE*)) {  
 var effect = new Effect();  
 effect.condition(Status.*FREEZE*).attack(0.0).chance(0.1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 }  
 @Override  
 public String describe() {  
 return "призывает ледяной ветер!";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class CosmicPower extends StatusMove {  
 public CosmicPower(){  
 super(Type.*PSYCHIC*, 0,0);  
 }  
 @Override  
 public void applySelfEffects(Pokemon pokemon) {  
 Effect effect = new Effect();  
 effect.stat(Stat.*DEFENSE*, 1).stat(Stat.*SPECIAL\_DEFENSE*, 1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 @Override  
 public String describe() {return "повышает свою защиту!";}  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class DefenseCurl extends StatusMove {  
 public DefenseCurl(){  
 super(Type.*NORMAL*, 0,0);  
 }  
 @Override  
 public void applySelfEffects(Pokemon pokemon) {  
 Effect effect = new Effect();  
 effect.stat(Stat.*DEFENSE*, 1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 @Override  
 public String describe() {  
 return "прячет голову в песок.";  
 }  
}

package Moves;  
  
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 pokemon, double damage){  
 Status effect = pokemon.getCondition();  
 flag = false;  
 if (effect == Status.*BURN* || effect == Status.*POISON* || effect == Status.*PARALYZE*){  
 flag = true;  
 pokemon.setMod(Stat.*HP*, -2\*(int)Math.*round*(damage));  
 }  
 }  
 @Override  
 public String describe() {  
 if (flag){return "зачаровывает";}  
 else {return "сильно зачаровывает";}  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class FocusBlast extends SpecialMove {  
 public FocusBlast(){  
 super(Type.*FIGHTING*, 120, 70);  
 }  
 public void applyOppEffects(Pokemon pokemon) {  
 var effect = new Effect();  
 effect.stat(Stat.*SPECIAL\_DEFENSE*, -1).chance(0.1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 @Override  
 public String describe() {  
 return "Применяет силу";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Growl extends StatusMove {  
 public Growl(){  
 super(Type.*NORMAL*, 0,100);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 Effect effect = new Effect();  
 effect.stat(Stat.*ATTACK*, -1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 @Override  
 public String describe() {  
 return "громко вопит";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class IceBeam extends SpecialMove {  
 public IceBeam(){  
 super(Type.*ICE*, 90, 100);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 if (!pokemon.hasType(Type.*ICE*)) {  
 var effect = new Effect();  
 effect.condition(Status.*FREEZE*).attack(0.0).chance(0.1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 }  
 @Override  
 public String describe() {  
 return "кастует ледяной луч!";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class PowderSnow extends SpecialMove {  
 public PowderSnow(){  
 super(Type.*ICE*, 40, 100);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 if (!pokemon.hasType(Type.*ICE*)) {  
 var effect = new Effect();  
 effect.condition(Status.*FREEZE*).attack(0.0).chance(0.1).turns(-1);  
 pokemon.setCondition(effect);  
 }  
 }  
 @Override  
 public String describe() {  
 return "кидается снежками!";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Swagger extends StatusMove {  
 public Swagger(){  
 super(Type.NORMAL, 0,85);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 pokemon.confuse();  
 Effect effect = new Effect();  
 effect.stat(Stat.ATTACK, 2);  
 pokemon.setCondition(effect);  
 }  
 @Override  
 public String describe() {  
 return "использует секретную технику \"тысячелетия боли\".";  
 }  
}

package Moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Thunder extends SpecialMove {  
 public Thunder(){  
 super(Type.*ELECTRIC*, 110, 70);  
 }  
 @Override  
 public void applyOppEffects(Pokemon pokemon) {  
 if (!pokemon.hasType(Type.*ELECTRIC*)) {  
 var effect = new Effect();  
 effect.condition(Status.*PARALYZE*).attack(0.75).chance(0.3).turns(-1);  
 effect.stat(Stat.*SPEED*, -2);  
 pokemon.setCondition(effect);  
 }  
 }  
 @Override  
 public String describe() {  
 return "применяет раскаты грома!";  
 }  
}

UML диаграмма

Изображение выглядит как текст, монитор, снимок экрана

Автоматически созданное описание

Заключение

В ходе лабораторной работы я изучил основы ООП, научился работать с классами, конструкторами, методами, полями, модификаторами доступа.