Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет ПИиКТ

Лабораторная работа №2

по дисциплине

«Программирование»

Вариант 353

Выполнил: Киселёв Сергей Владимирович

Проверил: Письмак Алексей Евгеньевич

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:

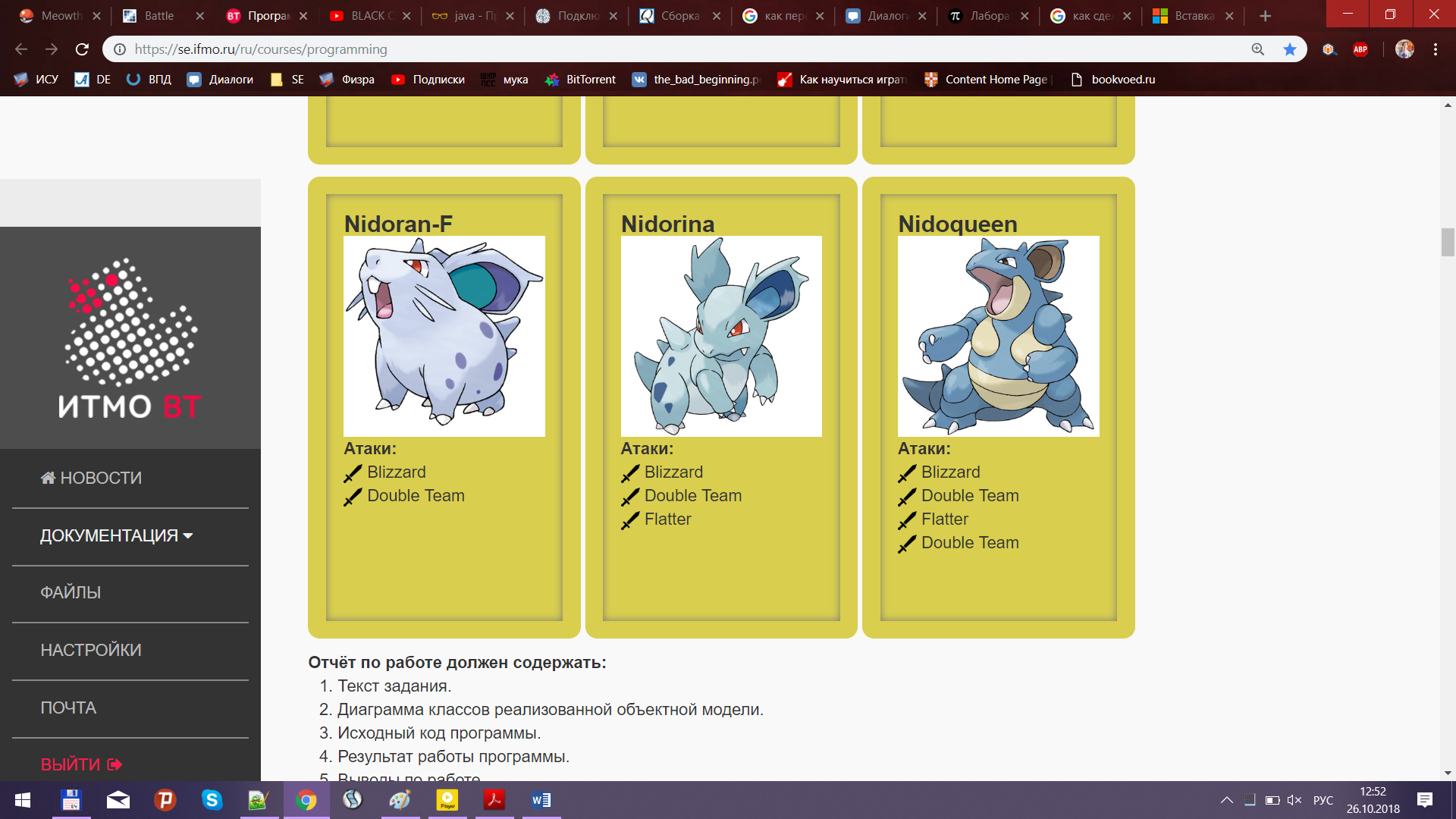
* очки здоровья (HP)
* атака (attack)
* защита (defense)
* специальная атака (special attack)
* специальная защита (special defense)
* скорость (speed)

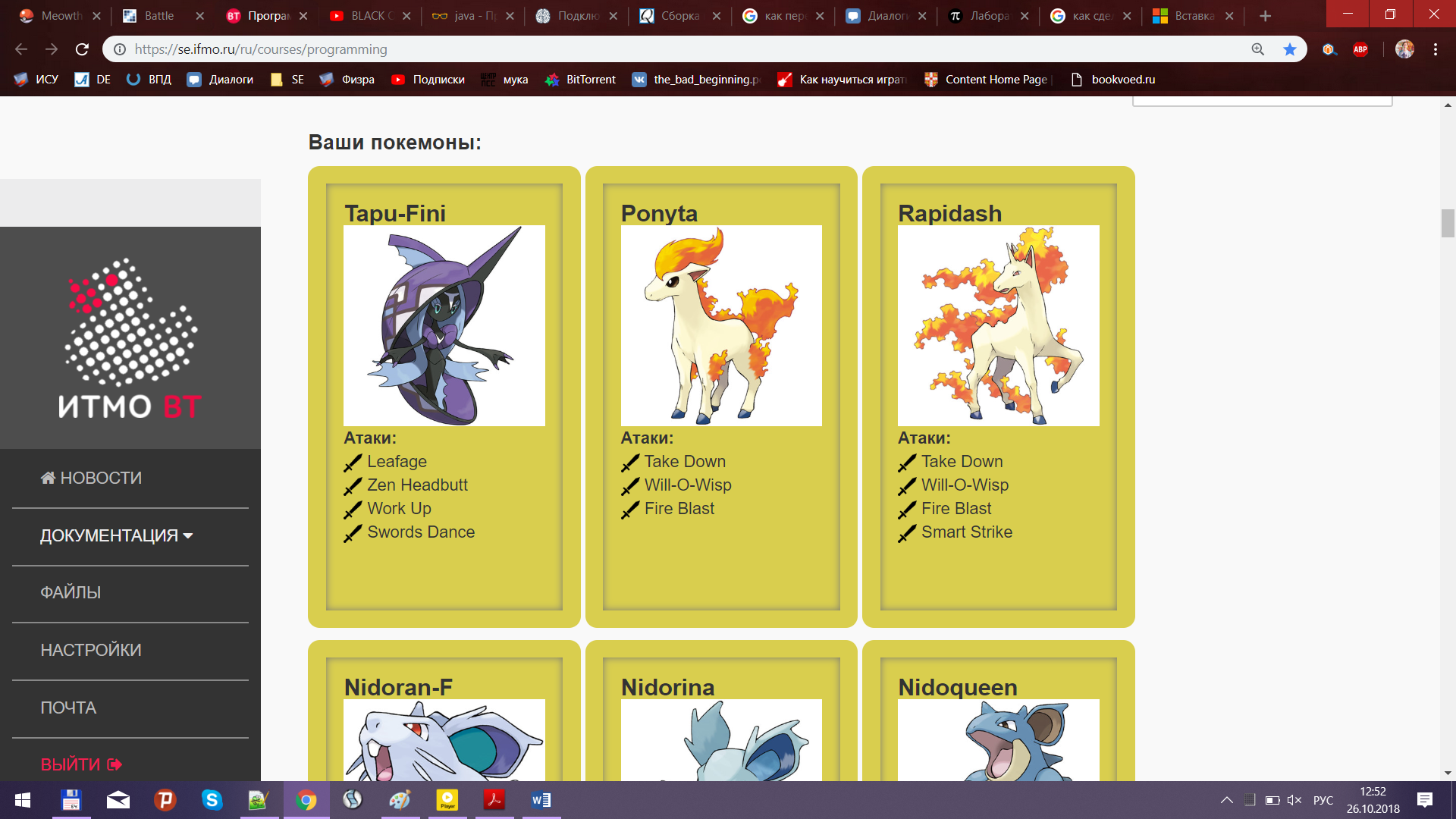
Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых классов PhysicalMove, SpecialMove и StatusMove реализовать свои классы для заданных видов атак.

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.

Вариант 353:





|  |
| --- |
| **+Tapu\_Fini** |
|  |
| +Tapu\_Fini(String, int) |
|  |
| **+Ponyta** |
|  |
| + Ponyta(String, int) |
|  |
| **+Rapidash** |
|  |
| + Rapidash(String, int) |
|  |
| **+Nidoran\_F** |
|  |
| + Nidoran\_F(String, int) |
|  |
| **+Nidorina** |
|  |
| + Nidorina(String, int) |
|  |
| **+Nidoqueen** |
|  |
| + Nidoqueen (String, int) |

|  |
| --- |
| **+Pokemon** |
| #name: String  -types: Type[]  -moves: Move[]  -preparedMove: Move  -stage: Efffect  -condition: Effect  -effects: List<Effect>  -confusion: int  #level: int  -base: double[] |
| +Pokemon(String, int)  +Pokemon()  +addEffect(Effect): void {final}  #addMove(Move): void {final}  #addType(Type): void {final}  +attack(Pokemon): Boolean {final}  +confuse(): void {final}  +getCondition(): Status {final}  +getHP(): double {final}  +getLevel(): int{final}  #getPreparedMove(): Move {final}  +getStat(Stat): double {final}  +getTypes(): Type[] {final}  +hasType(Type): Boolean {final}  +isAlive(): boolean {final}  +prepareMove(): void {final}  +restore(): void {final}  +setCondition(Effect): void {final}  +setLevel(int): void{final}  +setMod(Stat, int): void {final}  #setMove(Move…): void {final}  +setStats(double, double, double, double, double, double): void {final}  #setType(Type…): void{final}  +toString(): String {final}  +turn(): void {final} |

|  |
| --- |
| **+Battle** |
|  |
| +Battle()  +addAlly(Pokemon): void  +addFoe(Pokemon): void  +go(): void  +main(Strings[]): void {static} |

|  |
| --- |
| **+go** |
|  |
| +main(Strings[]): void {static} |

|  |
| --- |
| **+DamageMove** |
|  |
| +DamageMove()  +DamageMove(Type, double, double)  +DamageMove(Type, double, double, int, int)  #applyOppDamage(Pokemon, double): void  #applySelfDamage(Pokemon, double): void  +attack(Pokemon, Pokemon): void{final}  *#calcAttDefFactor(Pokemon, Pokemon):double*  #calcBaseDamage(Pokemon, Pokemon):double  #calcCriticalHit(Pokemon, Pokemon):double  #calcRandomDamage(Pokemon, Pokemon):double  #calcSameTypeAttackBonus(Pokemon, Pokemon): double  #calcTypeEffects(Pokemon, Pokemon):double |

|  |
| --- |
| ***+Move*** |
| #accuracy: double  #hits: int  #power: double  #priority: double  #type: Type |
| +Move()  +Move(Type,double, double)  +Move(Type, double, double, int, int)  #applyOppEffects(Pokemon): void  #applySelfEffects(Pokemon): void  *#attack(Pokemon, Pokemon): void*  #checkAccuracy(Pokemon, Pokemon): boolean  #describe(): String  +getConfusionMove(): Move {static final}  +getNoMove(): Move {static final}  +getPriority(): int {final}  +getStruggleMove(): Move {static final} |

|  |
| --- |
| **+PhysicalMove** |
|  |
| +PhysicalMove()  +PhysicalMove(Type, double, double)  +PhysicalMove(Type,double, double, int, int)  #calcAttDefFactor(Pokemon, Pokemon): double {final} |

|  |
| --- |
| **+SpecialMove** |
|  |
| +SpecialMove()  +SpecialMove(Type, double, double)  +SpecialMove(Type, double, double, int, int)  #calcAttDefFactor(Pokemon, Pokemon): double {final} |

|  |
| --- |
| **+StatusMove** |
|  |
| +StatusMove()  +StatusMove(Type, double, double)  +StatusMove(Type, double, double, int, int)  #applyOppEffects(Pokemon): void  #applySelfEffects(Pokemon): void  +attack(Pokemon, Pokemon): void{final} |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **~TakeDown** |  | **~Blizzard** |  | **~DoubleTeam** |
|  |  |  |  |  |
| +TakeDown()  #applySelfDamage(Pokemon, double):void |  | +Blizzard()  #applyOppEffects(Pokemon): void |  | +DoubleTeam()  #checkAccuracy(Pokemon, Pokemon): boolean  #applySelfEffects(Pokemon): void |
|  |  |  |  |  |
| **~SmartStrike** |  | **~FireBlast** |  | **~Flatter** |
|  |  |  |  |  |
| +SmartStrike()  #checkAccuracy(Pokemon, Pokemon): boolean |  | +FireBlast()  #applyOppEffects(Pokemon): void |  | +Flatter()  #applyOppEffects(Pokemon): void |
|  |  |  |  |  |
| **~Leafage** |  |  |  | **~Will\_O\_Wisp** |
|  |  |  |  |  |
| +Leafage() |  |  |  | +Will\_O\_Wisp()  #applyOppEffects(Pokemon): void |
|  |  |  |  |  |
| **~ZenHeadbutt** |  |  |  | **~WorkUp** |
|  |  |  |  |  |
| +ZenHeadbutt()  #applyOppEffects(Pokemon): void |  |  |  | +WorkUp()  #checkAccuracy(Pokemon, Pokemon): boolean  #applySelfEffects(Pokemon):void |
|  |  |  |  |  |
|  |  |  |  | **~SwordsDance** |
|  |  |  |  |  |
|  |  |  |  | +SwordsDance()  #checkAccuracy(Pokemon, Pokemon): boolean  #applySelfEffects(Pokemon):void |

**Tapu\_Fini.java**

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

public class Tapu\_Fini extends Pokemon

{

public Tapu\_Fini(String name, int lvl)

{

super(name, lvl);

setStats(70, 75, 115, 95, 130, 85);

setType(Type.WATER, Type.FAIRY);

setMove(new Leafage(), new ZenHeadbutt(), new WorkUp(), new SwordsDance());

}

}

class Leafage extends PhysicalMove // Strikes opponent with leaves.

{

public Leafage()

{

super(Type.GRASS, 40, 1.0);

}

}

class ZenHeadbutt extends PhysicalMove // Zen Headbutt deals damage and has a 20% chance of causing the target

{ // to flinch.

public ZenHeadbutt()

{

super(Type.PSYCHIC , 80, 0.9);

} @Override

protected void applyOppEffects(Pokemon p)

{

if(Math.random() < 0.2)

Effect.flinch(p);

}

}

class WorkUp extends StatusMove // Work Up raises the user's Attack and Special Attack by one stage each.

{ // Stats can be raised to a maximum of +6 stages each.

public WorkUp()

{

super(Type.NORMAL, 0, 0.0);

}

@Override

protected boolean checkAccuracy(Pokemon att, Pokemon def)

{

return true;

} @Override

protected void applySelfEffects(Pokemon p)

{

if (p.getStat(Stat.ATTACK) < 6) p.setMod(Stat.ATTACK, +1);

if (p.getStat(Stat.SPECIAL\_ATTACK) < 6) p.setMod(Stat.SPECIAL\_ATTACK, +1);

}

}

class SwordsDance extends StatusMove // Swords Dance raises the user's Attack by two stages.

{ // Stats can be raised to a maximum of +6 stages each.

public SwordsDance()

{

super(Type.NORMAL, 0, 0.0);

}

@Override

protected boolean checkAccuracy(Pokemon att, Pokemon def)

{

return true;

} @Override

protected void applySelfEffects(Pokemon p)

{

if (p.getStat(Stat.ATTACK) < 6)

p.setMod(Stat.ATTACK, +1);

if (p.getStat(Stat.ATTACK) < 6)

p.setMod(Stat.ATTACK, +1);

}

}

**Ponyta.java**

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

public class Ponyta extends Pokemon

{

public Ponyta(String name, int lvl)

{

super(name, lvl);

setStats(50, 85, 55, 65, 65, 90);

setType(Type.FIRE);

setMove(new TakeDown(), new Will\_O\_Wisp(), new FireBlast());

}

}

class TakeDown extends PhysicalMove //Take Down deals damage, but the user receives 1⁄4 of the damage it

{ // inflicted in recoil.

public TakeDown()

{

super(Type.NORMAL, 90, 0.85);

}

@Override

protected void applySelfDamage(Pokemon att, double damage)

{

att.setMod(Stat.HP, (int) Math.round(damage/4));

}

}

class Will\_O\_Wisp extends StatusMove // Will-O-Wisp causes the target to become burned, if it hits.

{

public Will\_O\_Wisp()

{

super(Type.FIRE, 0, 0.85);

}

@Override

protected void applyOppEffects(Pokemon p)

{

Effect.burn(p);

}

}

class FireBlast extends SpecialMove // Fire Blast deals damage and has a 10% chance of burning the target.

{

public FireBlast()

{

super(Type.FIRE, 110, 0.85);

}

@Override

protected void applyOppEffects(Pokemon p)

{

if (Math.random() < 0.1)

Effect.burn(p);

}

}

**Rapidash.java**

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

public class Rapidash extends Ponyta

{

public Rapidash(String name, int lvl)

{

super(name, lvl);

setStats(65, 100, 70, 80, 80, 105);

addMove(new SmartStrike());

}

}

class SmartStrike extends PhysicalMove // The user stabs the target with a sharp horn. This attack never misses.

{

public SmartStrike()

{

super(Type.STEEL, 70, 0);

} @Override

protected boolean checkAccuracy(Pokemon att, Pokemon def)

{

return true;

}

}

**Nidoran\_F.java**

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

public class Nidoran\_F extends Pokemon

{

public Nidoran\_F(String name, int lvl)

{

super(name, lvl);

setStats(55, 47, 52, 40, 40, 41);

setType(Type.POISON);

setMove(new Blizzard(), new DoubleTeam());

}

}

class Blizzard extends SpecialMove // Blizzard deals damage and has a 10% chance of freezing the target.

{

public Blizzard()

{

super(Type.ICE, 110, 0.7);

}

@Override

protected void applyOppEffects(Pokemon p)

{

if (Math.random() < 0.1)

Effect.freeze(p);

}

}

class DoubleTeam extends StatusMove // Double Team raises the user's Evasiveness by one stage, thus making

{ // the user more difficult to hit

public DoubleTeam() // Stats can be raised to a maximum of +6 stages each.

{

super(Type.NORMAL, 0, 0.0);

}

@Override

protected boolean checkAccuracy(Pokemon att, Pokemon def)

{

return true;

}

@Override

protected void applySelfEffects(Pokemon p)

{

if (p.getStat(Stat.EVASION) < 6) p.setMod(Stat.EVASION, +1);

}

}

**Nidorina.java**

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

public class Nidorina extends Nidoran\_F

{

public Nidorina(String name, int lvl)

{

super(name, lvl);

setStats(70, 62, 67, 55, 55, 56);

addMove(new Flatter());

}

}

class Flatter extends StatusMove // Confuses opponent, but raises its Special Attack by two stages.

{

public Flatter()

{

super(Type.DARK, 0, 1.0);

}

@Override

protected void applyOppEffects(Pokemon p)

{

Effect.confuse(p);

if (p.getStat(Stat.SPECIAL\_ATTACK) < 6)

p.setMod(Stat.SPECIAL\_ATTACK, +1);

if (p.getStat(Stat.SPECIAL\_ATTACK) < 6)

p.setMod(Stat.SPECIAL\_ATTACK, +1);

}

}

**Nidoqueen.java**

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

public class Nidoqueen extends Nidorina

{

public Nidoqueen(String name, int lvl)

{

super(name, lvl);

setStats(90, 92, 87, 75, 85, 76);

addMove(new DoubleTeam());

}

}

**Go.java**

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

public class go

{

public static void main(String[] args)

{

Battle b = new Battle();

b.addAlly(new Tapu\_Fini("Письмак", 1));

b.addAlly(new Rapidash("Алексей", 1));

b.addAlly(new Nidoran\_F("Евгеньевич", 1));

b.addFoe(new Nidorina("Самый", 1));

b.addFoe(new Ponyta("Мемный", 1));

b.addFoe(new Nidoqueen("Препод", 1));

b.go();

}

}

Tapu\_Fini из команды черных вступает в бой!

Nidorina из команды полосатых вступает в бой!

Tapu\_Fini атакует.

Nidorina атакует.

Tapu\_Fini теряет 3 здоровья.

Tapu\_Fini атакует.

Nidorina атакует.

Nidorina увеличивает уклоняемость.

Tapu\_Fini атакует.

Nidorina теряет 2 здоровья.

Nidorina атакует.

Tapu\_Fini растерянно попадает по себе.

Tapu\_Fini теряет 3 здоровья.

Nidorina атакует.

Nidorina увеличивает уклоняемость.

Tapu\_Fini атакует.

Nidorina атакует.

Nidorina увеличивает уклоняемость.

Tapu\_Fini атакует.

Nidorina теряет 11 здоровья.

Nidorina теряет сознание.

Ponyta из команды полосатых вступает в бой!

Ponyta атакует.

Tapu\_Fini теряет 4 здоровья.

Tapu\_Fini растерянно попадает по себе.

Tapu\_Fini теряет 5 здоровья.

Tapu\_Fini теряет сознание.

Rapidash из команды черных вступает в бой!

Rapidash атакует.

Ponyta атакует.

Rapidash теряет 6 здоровья.

Ponyta теряет 2 здоровья.

Rapidash промахивается

Ponyta атакует.

Rapidash атакует.

Ponyta теряет 5 здоровья.

Ponyta промахивается

Rapidash атакует.

Ponyta теряет 2 здоровья.

Ponyta атакует.

Rapidash теряет 4 здоровья.

Ponyta теряет 1 здоровья.

Rapidash атакует.

Ponyta атакует.

Rapidash атакует.

Ponyta теряет 5 здоровья.

Ponyta теряет сознание.

Nidoqueen из команды полосатых вступает в бой!

Rapidash промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Rapidash промахивается

Nidoqueen атакует.

Rapidash теряет 3 здоровья.

Rapidash теряет сознание.

Nidoran\_F из команды черных вступает в бой!

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoran\_F увеличивает специальную атаку.

Nidoran\_F атакует.

Nidoqueen теряет 6 здоровья.

Nidoqueen промахивается

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen промахивается

Nidoran\_F растерянно попадает по себе.

Nidoran\_F теряет 5 здоровья.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen промахивается

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen промахивается

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen промахивается

Nidoran\_F промахивается

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen промахивается

Nidoran\_F атакует.

Nidoran\_F увеличивает уклоняемость.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F промахивается

Nidoqueen промахивается

Nidoran\_F растерянно попадает по себе.

****Nidoran\_F теряет 3 здоровья.

Nidoqueen атакует.

Nidoqueen увеличивает уклоняемость.

Nidoran\_F атакует.

Nidoqueen теряет 7 здоровья.

Nidoqueen теряет сознание.

В команде полосатых не осталось покемонов.

Команда черных побеждает в этом бою!

**Выводы:** в процессе выполнения лабораторной работы были получены навыки использования объектно-ориентированного подхода программирования, хронический недосып и желание расстаться с сиею бренной жизнью. А также теперь я разбираюсь в покемонах, и вот самый котеечный из них(Glameow).