Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №2

Вариант

Преподаватель:

Выполнил:

Группа:

Санкт-Петербург

2018

Задание:

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)

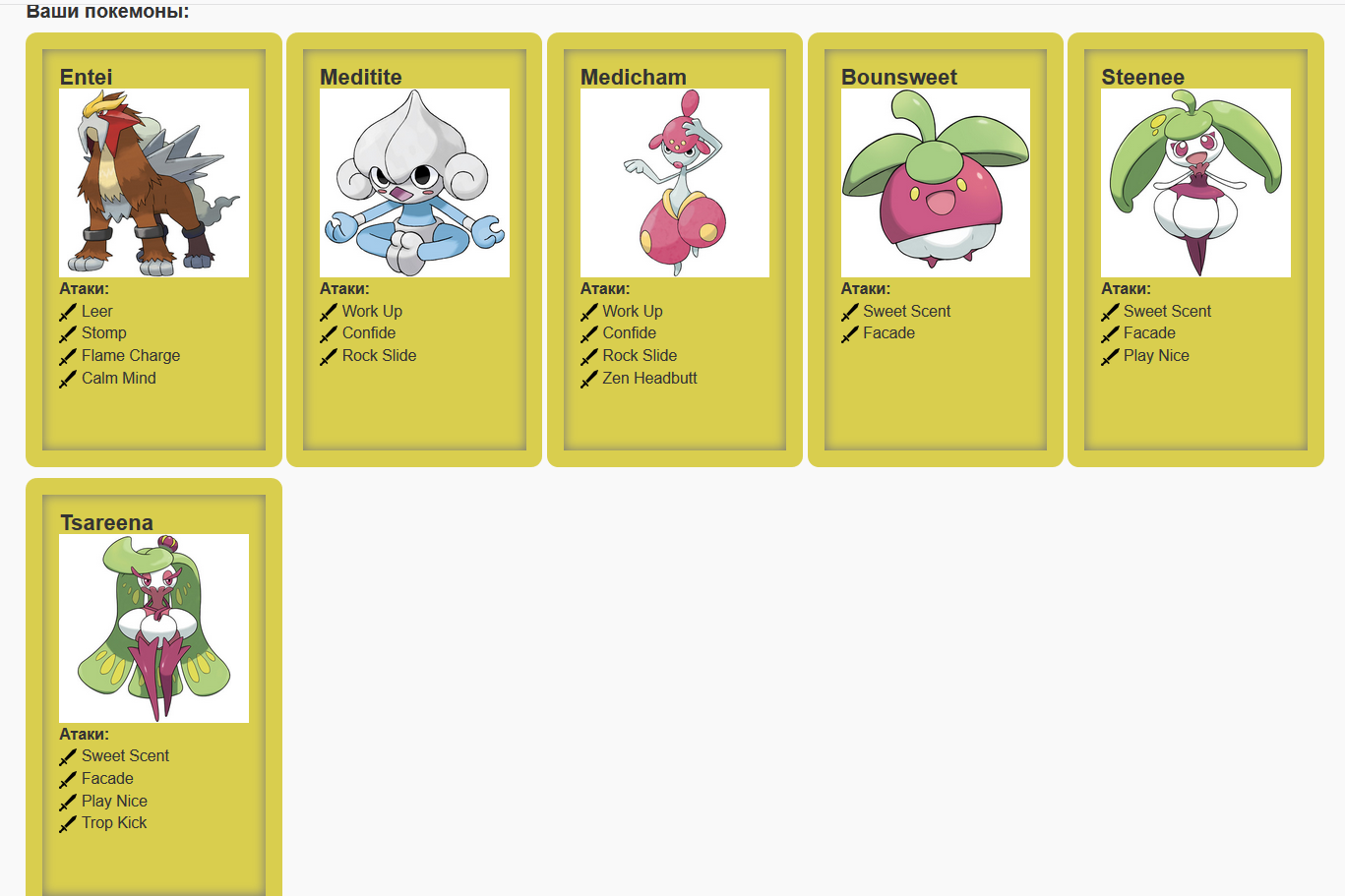
Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых классов PhysicalMove, SpecialMove и StatusMove реализовать свои классы для заданных видов атак.

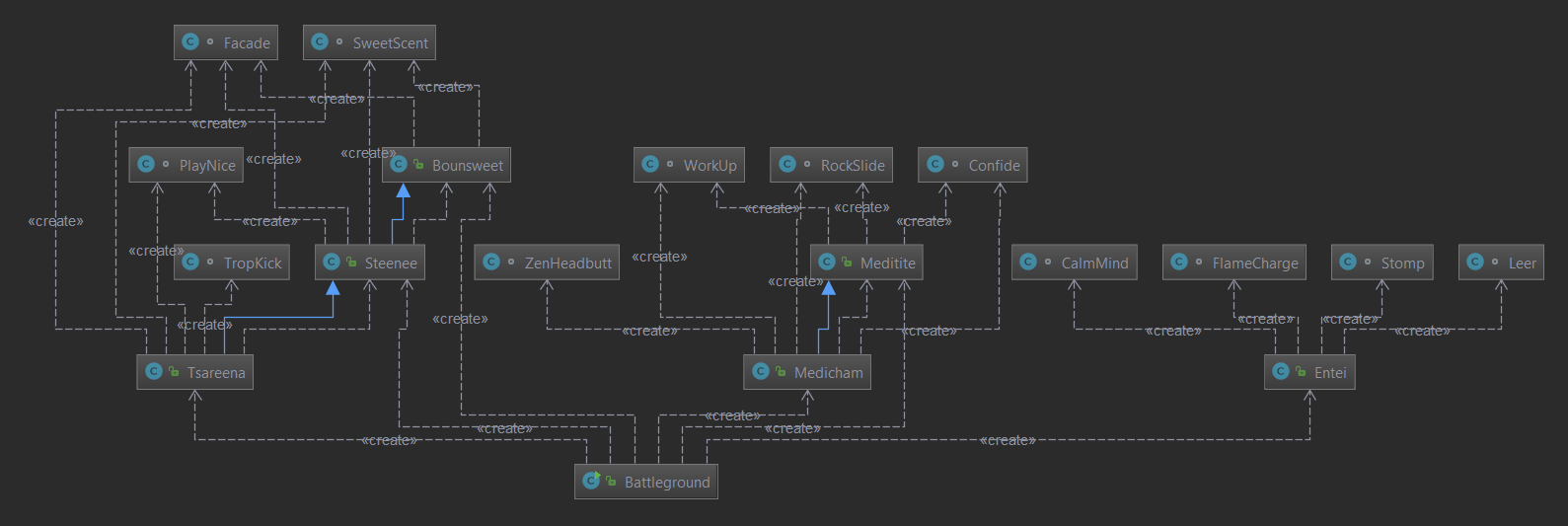
Атака должна иметь стандартные тип, силу (power) и точность (accuracy). Должны быть реализованы стандартные эффекты атаки. Назначить каждому виду покемонов атаки в соответствии с вариантом. Уровень покемона выбирается минимально необходимым для всех реализованных атак.

Используя класс симуляции боя Battle, создать 2 команды покемонов (каждый покемон должен иметь имя) и запустить бой.

Базовые классы и симулятор сражения находятся в [jar-архиве](https://se.ifmo.ru/documents/10180/660917/Pokemon.jar/a7ce60af-6ee6-47d0-a95e-e5ed9a697bd2) (обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - [здесь](https://se.ifmo.ru/~tony/doc/).

Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах <http://poke-universe.ru>, <http://pokemondb.net>,<http://veekun.com/dex/pokemon>

Покемоны:

Диаграмма классов:

Исходный код:

Battleground.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Battleground {  
  
 *public static void* main(String[] args) {  
 Battle field = *new* Battle();  
 field.addAlly(*new* Bounsweet("Pupa", 2));  
 field.addAlly(*new* Entei("Lupa", 1));  
 field.addAlly(*new* Medicham("Vupsen", 3));  
 field.addFoe(*new* Meditite("Pupsen", 2));  
 field.addFoe(*new* Steenee("Mayweather", 3));  
 field.addFoe(*new* Tsareena("Pacquiao", 2));  
 field.go();  
 }  
}

Meditite.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Meditite *extends* Pokemon {  
 *public* Meditite(String name, *int* level) {  
 *super*(name, level);  
 setStats(30, 40, 55, 40, 55, 60);  
 setType(Type.FIGHTING, Type.PSYCHIC);  
 setMove(*new* WorkUp(), *new* Confide(), *new* RockSlide());  
 }  
}

Medicham.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Medicham *extends* Meditite {  
 *public* Medicham(String name, *int* level) {  
 *super*(name, level);  
 setStats(60, 60, 75, 60, 75, 80);  
 setType(Type.FIGHTING, Type.PSYCHIC);  
 setMove(*new* WorkUp(), *new* Confide(), *new* RockSlide(), *new* ZenHeadbutt());  
 }  
}

Bounsweet.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Bounsweet *extends* Pokemon {  
 *public* Bounsweet(String name, *int* level) {  
 *super*(name, level);  
 setStats(42, 30, 38, 30, 38, 32);  
 setType(Type.GRASS);  
 setMove(*new* SweetScent(), *new* Facade());  
 }  
}

Steenee.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Steenee *extends* Bounsweet {  
 *public* Steenee(String name, *int* level) {  
 *super*(name, level);  
 setStats(52, 40, 48, 40, 48, 62);  
 setType(Type.GRASS);  
 setMove(*new* SweetScent(), *new* Facade(), *new* PlayNice());  
 }  
}

Tsareena.java

*import* ru.ifmo.se.pokemon.\*;  
  
*public class* Tsareena *extends* Steenee {  
 *public* Tsareena(String name, *int* level) {  
 *super*(name, level);  
 setStats(72, 120, 98, 50, 98, 72);  
 setType(Type.GRASS);  
 setMove(*new* SweetScent(), *new* Facade(), *new* PlayNice(), *new* TropKick());  
 }  
}

PokeAttacks.java

*import* ru.ifmo.se.pokemon.\*;  
  
*import* java.awt.\*;  
  
*/\*  
class MagicalLeaf extends SpecialMove{  
 protected MagicalLeaf(){  
 super(Type.GRASS, 60, 999999999);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 p.setMod(Stat.ACCURACY, 0);  
 p.setMod(Stat.EVASION, 0);  
 }  
}  
  
class EnergyBall extends SpecialMove{  
 protected EnergyBall(){  
 super(Type.GRASS, 90, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 if (Math.random() <= 0.1) p.setMod(Stat.SPECIAL\_DEFENSE, -1);  
 }  
}  
\*/  
class* PlayNice *extends* StatusMove{  
 *protected* PlayNice(){  
 *super*(Type.NORMAL, 0, 0);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 p.setMod(Stat.ATTACK, -1);  
 }  
}  
*/\*  
class Bite extends PhysicalMove{  
 protected Bite(){  
 super(Type.DARK, 60, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 if (Math.random() <= 30) Effect.flinch(p);  
 }  
}  
  
class Eruption extends SpecialMove{  
 protected Eruption(){  
 super(Type.FIRE, 150, 100);  
 }  
 @Override  
 protected void applyOppDamage(Pokemon def, double damage){  
 if (def.getHP() != def.getStat(Stat.HP))  
 def.setMod(Stat.HP, (int) ((def.getHP()) / def.getStat(Stat.HP) \* 150));  
 }  
}  
  
class Ember extends SpecialMove{  
 protected Ember(){  
 super(Type.FIRE, 40, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 if (Math.random() <= 0.1) Effect.burn(p);  
 }  
}  
\*/  
class* Leer *extends* StatusMove{  
 *protected* Leer(){  
 *super*(Type.NORMAL, 0, 100);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 p.setMod(Stat.ATTACK, -1);  
 }  
}  
*/\*  
class LeafStorm extends SpecialMove{  
 protected LeafStorm(){  
 super(Type.GRASS, 130, 90);  
 }  
 @Override  
 protected void applySelfEffects(Pokemon p){  
 p.setMod(Stat.SPECIAL\_ATTACK, -2);  
 }  
}  
  
class Swagger extends StatusMove{  
 protected Swagger(){  
 super(Type.NORMAL, 0, 85);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 p.setMod(Stat.ATTACK, 2);  
 Effect.confuse(p);  
 }  
}  
  
class PlayRough extends PhysicalMove{  
 protected PlayRough(){  
 super(Type.FAIRY, 90, 90);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 if (Math.random() <= 0.1) Effect.flinch(p);  
 p.setMod(Stat.ATTACK, -1);  
 }  
}  
  
class Confusion extends SpecialMove{  
 protected Confusion(){  
 super(Type.PSYCHIC, 50, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 if (Math.random() <= 0.1) Effect.confuse(p);  
 }  
}  
  
class FirePunch extends PhysicalMove {  
 protected FirePunch() {  
 super(Type.FIRE, 75, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 if (Math.random() <= 0.1) Effect.burn(p);  
 }  
}  
\*/  
class* Stomp *extends* PhysicalMove{  
 *protected* Stomp(){  
 *super*(Type.NORMAL, 65, 100);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 *if* (Math.*random*() <= 0.3) Effect.*flinch*(p);  
 p.setMod(Stat.ACCURACY, 0);  
 }  
 @Override  
 *protected void* applyOppDamage(Pokemon def, *double* damage){  
 def.setMod(Stat.HP, (*int*) Math.*round*(damage) \* 2);  
 }  
}  
*/\*  
class ShadowBall extends SpecialMove{  
 protected ShadowBall(){  
 super(Type.GHOST, 80, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 if (Math.random() <= 0.2) p.setMod(Stat.SPECIAL\_DEFENSE, -1);  
 }  
}  
\*/  
class* CalmMind *extends* StatusMove {  
 *protected* CalmMind() {  
 *super*(Type.PSYCHIC, 0, 0);  
 }  
 @Override  
 *protected void* applySelfEffects (Pokemon p){  
 p.setMod(Stat.SPECIAL\_ATTACK, 1);  
 p.setMod(Stat.SPECIAL\_DEFENSE, 1);  
 }  
 }  
*/\*  
class ForcePalm extends PhysicalMove{  
 protected ForcePalm() {  
 super(Type.FIGHTING, 60, 100);  
 }  
 @Override  
 protected void applyOppEffects(Pokemon p){  
 if (Math.random() <= 0.3) Effect.paralyze(p);  
 }  
}  
\*/  
  
class* FlameCharge *extends* PhysicalMove {  
 *protected* FlameCharge() {  
 *super*(Type.FIRE, 50, 100);  
 }  
 @Override  
 *protected void* applySelfEffects(Pokemon p) {  
 p.setMod(Stat.SPEED, 1);  
 }  
}  
  
*class* WorkUp *extends* StatusMove {  
 *protected* WorkUp(){  
 *super*(Type.NORMAL, 0, 0);  
 }  
 @Override  
 *protected void* applySelfEffects(Pokemon p){  
 p.setMod(Stat.ATTACK, 1);  
 p.setMod(Stat.SPECIAL\_ATTACK, 1);  
 }  
}  
  
*class* Confide *extends* StatusMove {  
 *protected* Confide(){  
 *super*(Type.NORMAL, 0, 0);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 p.setMod(Stat.SPECIAL\_ATTACK, -1);  
 }  
}  
  
*class* RockSlide *extends* PhysicalMove {  
 *protected* RockSlide(){  
 *super*(Type.ROCK, 75, 90);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 *if* (Math.*random*() <= 0.3) Effect.*flinch*(p);  
 }  
}  
  
*class* ZenHeadbutt *extends* PhysicalMove{  
 *protected* ZenHeadbutt(){  
 *super*(Type.PSYCHIC, 80, 90);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 *if* (Math.*random*() <= 0.2) Effect.*flinch*(p);  
 }  
}  
  
*class* SweetScent *extends* StatusMove{  
 *protected* SweetScent(){  
 *super*(Type.NORMAL, 0, 0);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 p.setMod(Stat.EVASION, -1);  
 }  
}  
  
*class* TropKick *extends* PhysicalMove{  
 *protected* TropKick(){  
 *super*(Type.GRASS, 70, 100);  
 }  
 @Override  
 *protected void* applyOppEffects(Pokemon p){  
 p.setMod(Stat.ATTACK, -1);  
 }  
}  
  
*class* Facade *extends* PhysicalMove{  
 *protected* Facade(){  
 *super*(Type.NORMAL, 70, 100);  
 }  
 @Override  
 *protected void* applyOppDamage(Pokemon def, *double* damage){  
 Status PokCon = def.getCondition();  
 *if* (PokCon.equals(Status.BURN) || PokCon.equals(Status.POISON) || PokCon.equals(Status.PARALYZE)) {  
 def.setMod(Stat.HP, (*int*) Math.*round*(damage) \* 2);  
 }  
 }  
}

Результат работы:

Entei Pupa из команды зеленых вступает в бой!

Meditite Pupsen из команды фиолетовых вступает в бой!

Entei Pupa атакует.

Meditite Pupsen теряет 10 здоровья.

Meditite Pupsen промахивается

Entei Pupa атакует.

Meditite Pupsen уменьшает атаку.

Meditite Pupsen промахивается

Entei Pupa промахивается

Meditite Pupsen промахивается

Entei Pupa атакует.

Meditite Pupsen теряет 7 здоровья.

Entei Pupa увеличивает скорость.

Meditite Pupsen теряет сознание.

Steenee Mayweather из команды фиолетовых вступает в бой!

Entei Pupa атакует.

Steenee Mayweather уменьшает атаку.

Steenee Mayweather атакует.

Entei Pupa атакует.

Steenee Mayweather уменьшает атаку.

Steenee Mayweather атакует.

Entei Pupa атакует.

Steenee Mayweather теряет 18 здоровья.

Entei Pupa увеличивает скорость.

Steenee Mayweather теряет сознание.

Tsareena Pacquiao из команды фиолетовых вступает в бой!

Entei Pupa атакует.

Tsareena Pacquiao теряет 8 здоровья.

Tsareena Pacquiao атакует.

Entei Pupa теряет 4 здоровья.

Entei Pupa уменьшает атаку.

Entei Pupa атакует.

Tsareena Pacquiao теряет 15 здоровья.

Entei Pupa увеличивает скорость.

Tsareena Pacquiao теряет сознание.

В команде фиолетовых не осталось покемонов.

Команда зеленых побеждает в этом бою!

Вывод:

В процессе выполнения лабораторной работы были получены навыки использования объектно-ориентированного подхода программирования при использовании языка Java.