Кафедра вычислительной техники

Программирование

Лабораторная работа №2

Вариант 311286

Преподаватель:

Выполнил: Нгуен Тхи Ми Ту

P3112

Санкт-Петербург

2020

1. Title:

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 (обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - здесь.

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

Комментарии

Цель работы: на простом примере разобраться с основными концепциями ООП и научиться использовать их в программах.

Что надо сделать (краткое описание)

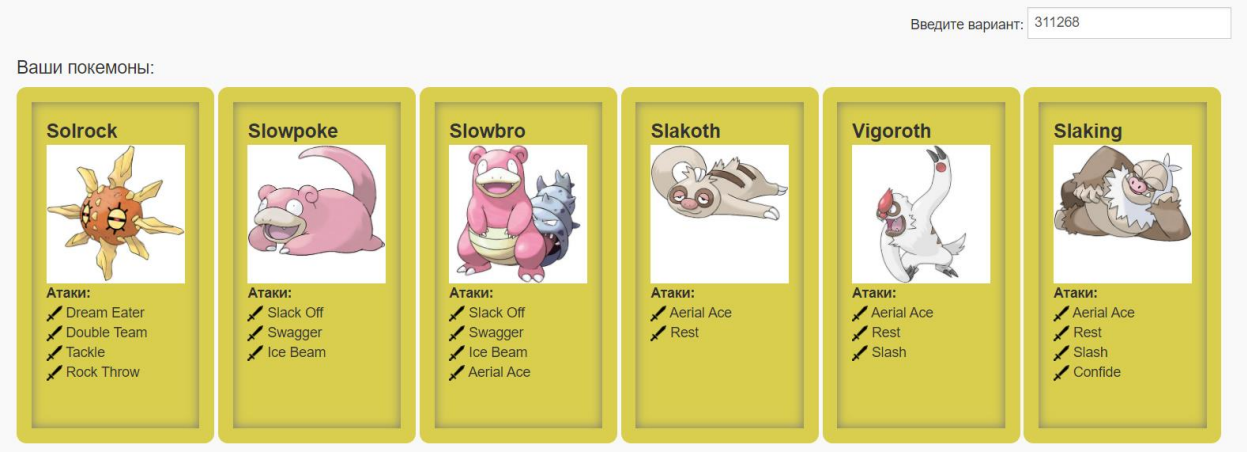
1. Ознакомиться с документацией, обращая особое внимание на классы `Pokemon` и `Move`. При дальнейшем выполнении лабораторной работы читать документацию еще несколько раз.
2. Скачать файл Pokemon.jar. Его необходимо будет использовать как для компиляции, так и для запуска программы. Распаковывать его не надо! Нужно научиться подключать внешние jar-файлы к своей программе.
3. Написать минимально работающую программу и посмотреть как она работает.

```
4. Battle b = new Battle();
5. Pokemon p1 = new Pokemon("Чужой", 1);
6. Pokemon p2 = new Pokemon("Хищник", 1);
7. b.addAlly(p1);
8. b.addFoe(p2);
9. b.go();
```

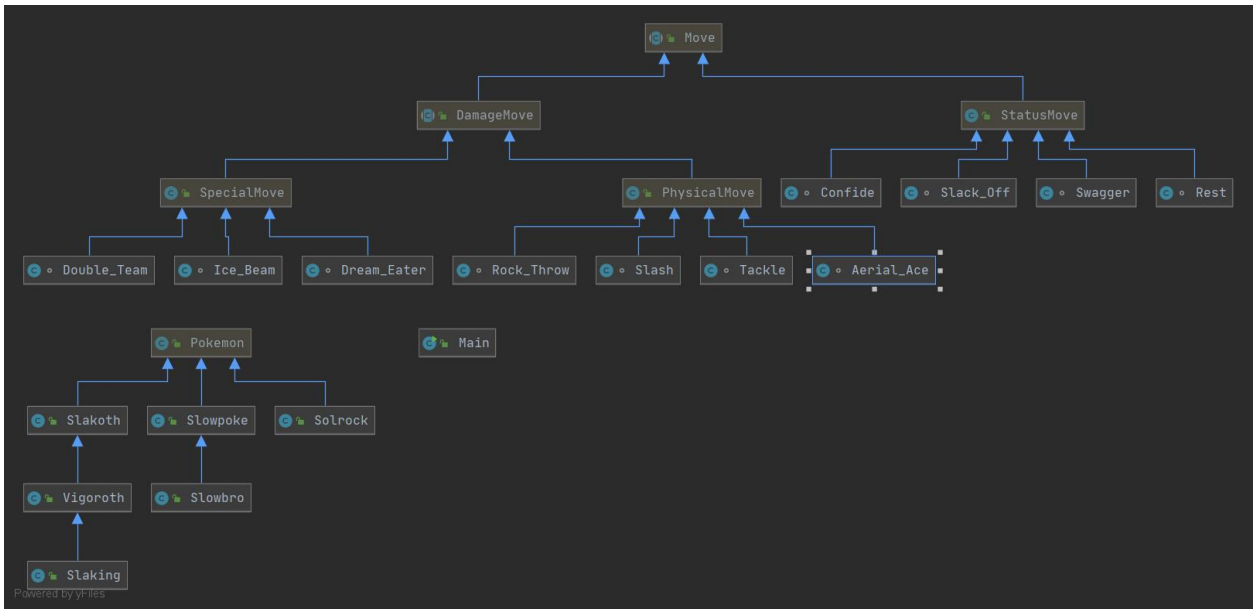
10. Создать один из классов покемонов для своего варианта. Класс должен наследоваться от базового класса `Pokemon`. В конструкторе нужно будет задать типы покемона и его базовые характеристики. После этого попробуйте добавить покемона в сражение.
11. Создать один из классов атак для своего варианта (лучше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

12. 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.
13. Доделать все необходимые атаки и всех покемонов, распределить покемонов по командам, запустить сражение.



2. Diagram:



3. Code

- Main.java

```
import ru.ifmo.se.pokemon.Battle;

public class Main {
        public static void main(String[] args) {
            Battle battle = new Battle();
            battle.addAlly(new Solrock("Trump", 10));
```

```
            battle.addAlly(new Slowpoke("Justin", 13));
            battle.addAlly(new Slowbro("Bidden", 9));
            battle.addFoe(new Vigoroth("Selena", 7));
            battle.addFoe(new Slaking("Jasmine", 15));
            battle.addFoe(new Slakoth("Aladin", 9));
            battle.go();
        }
}
```

- Solrock.java

```
import ru.ifmo.se.pokemon.*;
public class Solrock extends Pokemon {
    public Solrock(String name, int level) {
        super(name, level);
        setStats(90, 95, 85, 55, 65, 70);
        setType(Type.PSYCHIC, Type.ROCK);
        setMove(new Dream_Eater(), new Double_Team(), new Tackle(), new
Rock_Throw());

    }
}
```

- Slowpoke

```
import ru.ifmo.se.pokemon.*;
public class Slowpoke extends Pokemon {
        public Slowpoke(String name,int level){
            super(name,level);
            setStats(90,65,65,40,40,15);
            setType(Type.WATER,Type.PSYCHIC);
            setMove(new Slack_Off(),new Swagger(),new Ice_Beam());
        }
}
```

- Slowbro

```
import ru.ifmo.se.pokemon.*;
public class Slowbro extends Slowpoke{
    public Slowbro(String name, int level){
        super(name,level);
        setType(Type.WATER,Type.PSYCHIC);
        setStats(95,75,110,100,80,30);
        setMove(new Slack_Off(),new Swagger(),new Ice_Beam(),new
Aerial_Ace());
    }
}
```

- Vigoroth

```
import ru.ifmo.se.pokemon.*;
public class Vigoroth extends Slakoth{
    public Vigoroth(String name,int level){
        super(name,level);
        setStats(80,80,80,55,55,90);
        setType(Type.NORMAL);
        setMove(new Aerial_Ace(),new Rest(),new Slash());
    }
}
```

- Slaking

```
import ru.ifmo.se.pokemon.*;
public class Slaking extends Vigoroth{
    public Slaking(String name,int level){
```

```
        super(name,level);
        setStats(150,160,100,95,65,100);
        setMove(new Aerial_Ace(),new Rest(),new Slash(),new Confide());
    }
}
```

- Slakoth

```
import ru.ifmo.se.pokemon.*;
public class Slakoth extends Pokemon{
    public Slakoth(String name,int level){
        super(name,level);
        setStats(60,60,60,35,35,30);
        setType(Type.NORMAL);
        setMove(new Aerial_Ace(),new Rest());
    }
}
```

- Double_Team.java

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

class Double_Team extends SpecialMove {
        public Double_Team() {
            super(Type.NORMAL,0,0);
        }

        @Override
        protected void applySelfEffects(Pokemon p) {

            p.setMod(Stat.EVASION,1);
        }

        @Override
        protected String describe() {
            return "Double Team raises the user's Evasiveness by one stage";
        }

}
```

- Aerial_Ace.java

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

class Aerial_Ace extends PhysicalMove {
    public Aerial_Ace(){
        super(Type.FLYING,100,100);
    }

    @Override
    protected void applySelfEffects(Pokemon p) {
        p.addEffect(new Effect().attack(100).turns(-1));
    }

    @Override
    protected void applyOppDamage(Pokemon def, double damage) {
        def.setMod(Stat.HP,(int) damage);
    }

    @Override
    protected String describe() {
        return("Aerial Ace deals damage");
    }
}
```

- Confide.java

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

class Confide extends StatusMove {
    public Confide(){
        super(Type.NORMAL,0,0);
    }

    @Override
    protected void applyOppEffects(Pokemon def) {
        ;def.setMod(Stat.SPECIAL_ATTACK,-1);
    }

    @Override
    protected String describe() {
        return "Confide lowers the target's Special Attack by one stage.";
    }
}
```

- Dream_Eater.java

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

 class Dream_Eater extends SpecialMove {
        public Dream_Eater() {
            super(Type.PSYCHIC,100,100);
        }

        @Override
        protected void applyOppDamage(Pokemon def, double damage) {
            if (def.getCondition() == Status.SLEEP) {
                def.setMod(Stat.HP, (int) damage);
            }
        }

        @Override
        protected void applySelfEffects(Pokemon p) {
            p.setMod(Stat.HP,(int) (p.getStat(Stat.HP)-p.getHP())/2);
        }

        @Override
        protected String describe() {
            return "Dream Eater deals damage only on sleeping foes and the
user will recover 50% of the HP drained.";
        }

}
```

- Ice_Beam.java

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

class Ice_Beam extends SpecialMove {
    public Ice_Beam() {
        super(Type.ICE, 90, 100);
    }

    @Override
    protected void applyOppDamage(Pokemon def, double damage) {

        def.setMod(Stat.HP, (int) damage);
    }

    @Override
```

```
    protected void applyOppEffects(Pokemon def) {
        if (Math.random() <= 0.1) {
            Effect.freeze(def);
        }
    }

    @Override
    protected String describe() {
        return("Ice Beam deals damage and has a 10% chance of freezing the
target.");
    }
}
```

- Rest.java

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

class Rest extends StatusMove {
    public Rest(){
        super(Type.PSYCHIC,0,0);
    }

    @Override
    protected void applySelfEffects(Pokemon p){
        p.addEffect( new Effect().condition(Status.SLEEP).turns(2));
        p.setMod(Stat.HP, (int) p.getStat(Stat.HP));
    }

    @Override
    protected  String describe(){
        return "User sleeps for 2 turns, but user is fully healed.";
    }
}
```

- Rock_Throw.java

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

 class Rock_Throw extends PhysicalMove {
        public Rock_Throw(){
            super(Type.ROCK,50,90);
        }

        @Override
        protected void applyOppDamage(Pokemon def, double damage) {
            def.setMod(Stat.HP,(int) damage);
        }

        @Override
        protected String describe() {
            return "Rock Throw deals damage with no additional effect.";
        }
}
```

- Slack_Off.java

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

 class Slack_Off extends StatusMove {
        public Slack_Off(){
            super(Type.NORMAL,0,0);
        }

        @Override
        protected void applySelfEffects(Pokemon p) {
            if(p.getHP() < p.getStat(Stat.HP)){
```

```
                p.setMod(Stat.HP,(int) Math.round(p.getStat(Stat.HP))/2);
            }
        }

        @Override
        protected String describe() {
            return("Slack Off recovers up to 50% of the user's maximum HP.");
        }
 }
```

- Slash.java

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

class Slash extends PhysicalMove {
    public Slash(){
        super(Type.NORMAL,70,100);
    }

    @Override
    protected double calcCriticalHit(Pokemon att, Pokemon def) {
        return 1d/8d;
    }

    @Override
    protected String describe() {
        return "Shadow Claw deals damage and has an increased critical hit
ratio";
    }
}
```

- Swagger.java

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

class Swagger extends StatusMove {
        public Swagger(){
            super(Type.NORMAL,0,85);
        }

        @Override
        protected void applyOppEffects(Pokemon def) {
            def.setMod(Stat.ATTACK,2);
            Effect.confuse(def);
        }

        @Override
        protected String describe() {
            return("Swagger confuses the target and raises its Attack by two
stages.");
        }
}
```

- Tackle.java

```
import ru.ifmo.se.pokemon.*;
class Tackle extends PhysicalMove {
        public Tackle(){
            super(Type.NORMAL,40,100);
        }

        @Override
        protected void applyOppDamage(Pokemon def, double damage) {

            def.setMod(Stat.HP,(int) damage);
        }
```

```
        @Override
        protected String describe() {
            return"Tackle deals damage with no additional effects.";
        }
}
```

4. Result:
   Solrock Trump from the team Greren enters the battle!
   Vigoroth Selena from the team Red enters the battle!
   Solrock Trump Rock Throw deals damage with no additional effect..
   Vigoroth Selena loses 14 hit points.

   Vigoroth Selena Shadow Claw deals damage and has an increased critical hit ratio.
   Solrock Trump loses 1 hit points.

   Solrock Trump Dream Eater deals damage only on sleeping foes and the user will recover 50% of the HP drained..

   Vigoroth Selena misses

   Solrock Trump Tackle deals damage with no additional effects..
   Vigoroth Selena loses 6 hit points.

   Vigoroth Selena Shadow Claw deals damage and has an increased critical hit ratio.

   Solrock Trump Dream Eater deals damage only on sleeping foes and the user will recover 50% of the HP drained..

   Vigoroth Selena Aerial Ace deals damage.
   Solrock Trump loses 3 hit points.

   Solrock Trump Tackle deals damage with no additional effects..
   Vigoroth Selena loses 6 hit points.

   Vigoroth Selena Shadow Claw deals damage and has an increased critical hit ratio.
   Solrock Trump loses 1 hit points.

   Solrock Trump Rock Throw deals damage with no additional effect..
   Vigoroth Selena loses 14 hit points.
   Vigoroth Selena faints.
   Slaking Jasmine from the team Red enters the battle!
   Slaking Jasmine misses

   Solrock Trump Tackle deals damage with no additional effects..
   Slaking Jasmine loses 5 hit points.

   Slaking Jasmine Aerial Ace deals damage.
   Solrock Trump loses 14 hit points.

Solrock Trump misses

Slaking Jasmine Shadow Claw deals damage and has an increased critical hit ratio.
Solrock Trump loses 1 hit points.

Solrock Trump Dream Eater deals damage only on sleeping foes and the user will recover 50% of the HP drained..
Solrock Trump loses 10 hit points.

Slaking Jasmine Shadow Claw deals damage and has an increased critical hit ratio.
Solrock Trump loses 2 hit points.

Solrock Trump Dream Eater deals damage only on sleeping foes and the user will recover 50% of the HP drained..
Solrock Trump loses 16 hit points.
Solrock Trump faints.
Slowpoke Justin from the team Greren enters the battle!
Slaking Jasmine misses

Slowpoke Justin Swagger confuses the target and raises its Attack by two stages..
Slaking Jasmine increases attack.

Slaking Jasmine misses

Slowpoke Justin misses

Slaking Jasmine misses

Slowpoke Justin misses

Slaking Jasmine Aerial Ace deals damage.
Slowpoke Justin loses 27 hit points.

Slowpoke Justin Swagger confuses the target and raises its Attack by two stages..
Slaking Jasmine increases attack.

Slaking Jasmine misses

Slowpoke Justin misses

Slaking Jasmine misses

Slowpoke Justin Ice Beam deals damage and has a 10% chance of freezing the target..
Slaking Jasmine loses 7 hit points.

Slaking Jasmine misses

Slowpoke Justin Swagger confuses the target and raises its Attack by two stages..
Slaking Jasmine increases attack.

Slaking Jasmine hits himself in confusion.
Slaking Jasmine loses 23 hit points.

Slowpoke Justin misses

Slaking Jasmine misses

Slowpoke Justin misses

Slaking Jasmine hits himself in confusion.
Slaking Jasmine loses 30 hit points.

Slowpoke Justin Swagger confuses the target and raises its Attack by two stages..
Slaking Jasmine increases attack.

Slaking Jasmine Aerial Ace deals damage.
Slowpoke Justin loses 71 hit points.
Slowpoke Justin faints.
Slowbro Bidden from the team Greren enters the battle!
Slaking Jasmine hits himself in confusion.
Slaking Jasmine loses 34 hit points.
Slaking Jasmine faints.
Slakoth Aladin from the team Red enters the battle!
Slowbro Bidden Swagger confuses the target and raises its Attack by two stages..
Slakoth Aladin increases attack.

Slakoth Aladin misses

Slakoth Aladin misses

Slakoth Aladin misses

Slakoth Aladin Aerial Ace deals damage.
Slowbro Bidden loses 12 hit points.

Slowbro Bidden Ice Beam deals damage and has a 10% chance of freezing the target..
Slakoth Aladin loses 12 hit points.
Slakoth Aladin is frozen

Slowbro Bidden Aerial Ace deals damage.
Slakoth Aladin loses 12 hit points.

Slowbro Bidden misses

Slowbro Bidden Ice Beam deals damage and has a 10% chance of freezing the target..
Slakoth Aladin loses 14 hit points.
Slakoth Aladin faints.
Team Red loses its last Pokemon.

The team Greren wins the battle!

Process finished with exit code 0

5. Вывод:
   After this lab, I know about Object-Oriented Programming, Class, Object, Access Modified, Encapsulation, Inheritance, Polymorphism…