Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Дисциплина “Программирование”

Отчёт по лабораторной работе №2

Вариант №19880

Выполнил:

Ануфриев Андрей Сергеевич,

P3119

Проверил:

Ермаков Михаил Константинович

г. Санкт-Петербург

2024 год

Оглавление

1. Текст задания

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.

2. Исходный код программы.

```
package Lab2.moves.physical;

import ru.ifmo.se.pokemon.*;

public final class Aerial_Ace extends PhysicalMove{
    public Aerial_Ace(){
        super (Type.FLYING, 60,1);
    }
    @Override
    public String describe(){
        return "использует Aerial Ace";
    }
}
        package Lab2.moves.physical;

import ru.ifmo.se.pokemon.*;

public final class Fury_Swipes extends PhysicalMove{
    public Fury_Swipes(){
        super (Type.NORMAL, 18, 0.8);
    }
    @Override
    protected String describe(){
        return("использует Fury Swipes");
    }
}
package Lab2.moves.physical;

import ru.ifmo.se.pokemon.*;

public final class Poison_Sting extends PhysicalMove{
    public Poison_Sting(){
        super(Type.POISON, 15, 1);
    }
    @Override
    protected String describe(){
        return("использует Poison Sting");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if (Math.random()<=0.3){
            Effect.poison(p);
        }
    }
}
package Lab2.moves.physical;

import ru.ifmo.se.pokemon.PhysicalMove;
import ru.ifmo.se.pokemon.Type;

public final class Shadow_Claw extends PhysicalMove{
    public Shadow_Claw(){
        super(Type.GHOST, 70, 1);
    }
    @Override
    protected String describe(){
        return ("использует Shadow Claw");
    }
}
package Lab2.moves.physical;

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

```java
public final class Smart_Strike extends PhysicalMove{
    public Smart_Strike(){
        super (Type.STEEL, 70, 0);
    }
    @Override
    protected String describe(){
        return ("использует Smart Strike");
    }
}
```

```java
package Lab2.moves.special;

import ru.ifmo.se.pokemon.*;

public final class Focus_Blast extends SpecialMove{
    public Focus_Blast(){
        super(Type.FIGHTING, 120, 0.7);
    }
    @Override
    protected String describe(){
        return("использует Focus Blast");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if(Math.random()<= 0.1){
            p.addEffect(new Effect().stat(Stat.SPECIAL_DEFENSE, -1));
        }
    }
}
```

```java
package Lab2.moves.special;

import ru.ifmo.se.pokemon.*;

public final class Thunder extends SpecialMove{
    public Thunder(){
        super(Type.ELECTRIC, 110, 0.7);
    }
    @Override
    protected String describe(){
        return("использует Thunder");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        if (Math.random()<=0.3){
            Effect.paralyze(p);
        }
    }
}
```

```java
package Lab2.moves.status;

import ru.ifmo.se.pokemon.*;

public final class Confide extends StatusMove {
    public Confide(){
        super(Type.NORMAL, 0, 0);
    }
    @Override
    protected String describe(){
        return("использует Confide");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        p.addEffect(new Effect().stat(Stat.SPECIAL_ATTACK, -1));
    }
}
```

```java
package Lab2.moves.status;

import ru.ifmo.se.pokemon.*;

public final class Double_Team extends StatusMove {
    public Double_Team(){
        super (Type.NORMAL, 0, 0);
    }
    @Override
    protected String describe(){
        return ("исполльзует  Double Team");
    }
    @Override
    protected void applySelfEffects(Pokemon p){
        p.addEffect(new Effect().stat(Stat.DEFENSE,+1));
    }
}
package Lab2.moves.status;

import ru.ifmo.se.pokemon.*;

public final class Leer extends StatusMove{
    public Leer(){
        super(Type.NORMAL, 0, 1);
    }
    @Override
    protected String describe(){
        return("использует Leer");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        p.addEffect(new Effect().stat(Stat.DEFENSE, -1));
    }
}
package Lab2.moves.status;

import ru.ifmo.se.pokemon.*;

public final class Swagger extends StatusMove{
    public Swagger(){
        super(Type.NORMAL, 0, 0.85);
    }
    @Override
    protected String describe(){
        return ("использует Swagger");
    }
    @Override
    protected void applyOppEffects(Pokemon p){
        Effect.confuse(p);
        p.addEffect(new Effect().stat(Stat.ATTACK, +2));

    }
}
package Lab2.moves.status;

import ru.ifmo.se.pokemon.*;

public final class Swords_Dance extends StatusMove{
    public  Swords_Dance (){
        super (Type.NORMAL,0,0);
    }
    @Override
    protected void applySelfEffects(Pokemon p){
        p.addEffect(new Effect().stat(Stat.ATTACK, (int)p.getStat(Stat.ATTACK)+2));
    }
```

```
    @Override
    protected String describe(){
        return "использует  Swords Dance";
    }
}
package Lab2.mypokemon;

import Lab2.moves.physical.*;
import Lab2.moves.status.*;
import ru.ifmo.se.pokemon.*;

public final class Ariados extends Spinarak {
    public Ariados (String name, int lvl){
        super(name, lvl);
        setType(Type.BUG, Type.POISON);
        setStats(70, 90, 70, 60, 70, 40);
        setMove(new Swagger(), new Fury_Swipes(), new Confide(), new Smart_Strike());
    }
}
package Lab2.mypokemon;

import Lab2.moves.physical.Aerial_Ace;
import Lab2.moves.special.Focus_Blast;
import Lab2.moves.status.Double_Team;
import Lab2.moves.status.Swords_Dance;
import ru.ifmo.se.pokemon.*;

public final class Groudon extends Pokemon {
    public Groudon(String name, int lvl){
        super (name, lvl);
        setType(Type.GROUND);
        setStats(100, 150, 140, 100, 90, 90);
        setMove(new Aerial_Ace(), new Swords_Dance(), new Focus_Blast(), new
Double_Team());
    }
}
package Lab2.mypokemon;

import Lab2.moves.physical.*;
import Lab2.moves.special.*;
import Lab2.moves.status.*;
import ru.ifmo.se.pokemon.*;

public final class Nidoking extends Nidorino{
    public Nidoking (String name, int lvl){
        super (name, lvl);
        setType(Type.POISON, Type.GROUND);
        setStats(81, 102, 77, 85, 75, 85);
        setMove(new Thunder(), new Poison_Sting(), new Leer(), new Shadow_Claw());
    }
}
package Lab2.mypokemon;

import Lab2.moves.physical.Poison_Sting;
import Lab2.moves.special.Thunder;
import ru.ifmo.se.pokemon.*;

public class Nidoran_M extends Pokemon{
    public Nidoran_M (String name, int lvl){
        super(name, lvl);
        setType(Type.POISON);
        setStats(46, 57, 40, 40, 40, 50);
        setMove(new Thunder(), new Poison_Sting());
    }
}
```

```java
package Lab2.mypokemon;

import Lab2.moves.physical.Poison_Sting;
import Lab2.moves.special.Thunder;
import Lab2.moves.status.Leer;
import ru.ifmo.se.pokemon.*;

public class Nidorino extends Nidoran_M{
    public Nidorino (String name, int lvl){
        super (name, lvl);
        setType(Type.POISON);
        setStats(61, 72, 57, 55, 55, 65);
        setMove(new Thunder(), new Poison_Sting(), new Leer());
    }
}
```

```java
package Lab2.mypokemon;

import Lab2.moves.physical.Fury_Swipes;
import Lab2.moves.status.Confide;
import Lab2.moves.status.Swagger;
import ru.ifmo.se.pokemon.*;

public class Spinarak extends Pokemon {
    public Spinarak(String name, int lvl) {
        super (name, lvl);
        setType(Type.BUG, Type.POISON);
        setStats(40,60,40,40,40,30);
        setMove(new Swagger(), new Fury_Swipes(), new Confide());
    }
}
```

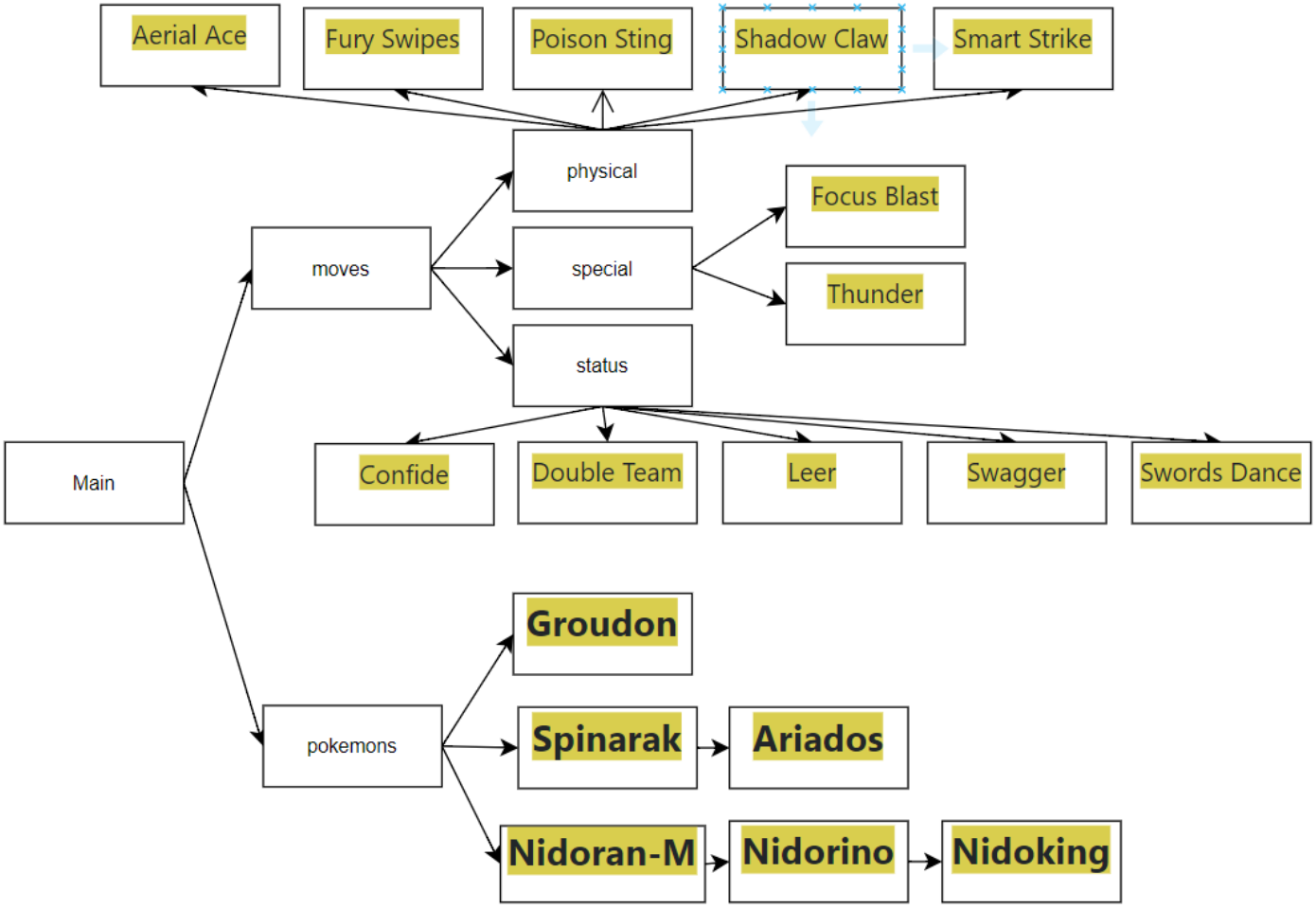
```java
package Lab2;

import Lab2.mypokemon.*;
import ru.ifmo.se.pokemon.*;

public class Main {
    public static void main(String[] args) {
        Battle a = new Battle();

        a.addFoe(new Ariados("Ivan", 30));
        a.addFoe(new Groudon("Vasiliy", 30));
        a.addFoe(new Nidoking("Petr", 30));
        a.addAlly(new Nidorino("Ibragim", 30));
        a.addAlly(new Nidoran_M("Muxamed", 30));
        a.addAlly(new Spinarak("Abdulla", 30));
        a.go();
    }
}
```

3. Диаграмма классов реализованной объектной модели.



4. Результат работы программы.

Nidorino Ibragim из команды синих вступает в бой!

Ariados Ivan из команды черных вступает в бой!

Nidorino Ibragim использует Thunder.

Ariados Ivan теряет 20 здоровья.

Ariados Ivan использует Swagger.

Nidorino Ibragim использует Leer.

……

Nidoking Petr растерянно попадает по себе.

Nidoking Petr теряет 6 здоровья.

Spinarak Abdulla промахивается

Nidoking Petr растерянно попадает по себе.

Nidoking Petr теряет 6 здоровья.

Nidoking Petr теряет сознание.

В команде желтых не осталось покемонов.

Команда черных побеждает в этом бою!

5. Ответы на вопросы

Объект — программная модель реальных объектов или абстрактных понятий, представляющая собой совокупность переменных, задающих состояние объекта, и связанных с ними методов, определяющих поведение объекта.

Класс — это прототип, описывающий переменные и методы, определяющие характеристики объектов данного класса.

Инкапсуляция — сокрытие данных внутри объекта и обеспечение доступа к ним с помощью общедоступных методов

Наследование или расширение — приобретение одним классом (подклассом) свойств другого класса (суперкласса). При наследовании поля и методы суперкласса (с модификаторами public или protected доступны для использования в подклассе. При этом в подклассе могут быть дополнительно определены собственные поля и методы.

Полиморфизм — возможность единообразно обрабатывать объекты разных типов. Различают полиморфизм включения (или полиморфизм подтипов), который позволяет обращаться с помощью единого интерфейса к классу и к любым его потомкам, и параметрический полиморфизм, который определяет обобщенный интерфейс с одинаковой реализацией.

Поле – характеристики

Методы – способности

Конструктор – это то как мы создаём покемона, добавляя ему характеристики и способности.(вернее сам конструктор уже создан в архиве, а мы его только заполняем

Модификаторы доступа –

Private — наиболее строгий модификатор доступа. Он ограничивает видимость данных и методов пределами одного класса.

Default (package visible) по умолчанию

protected

Поля и методы, обозначенные модификатором доступа protected, будут видны:

в пределах всех классов, находящихся в том же пакете, что и наш;

в пределах всех классов-наследников нашего класса.

Public – виден всем, нужен для общение с пользователем

Модификаторы	Доступ из…		
	Своего класса	Своего пакета	Любого класса
private	Есть	Нет	Нет
нет модификатора (**package**)	Есть	Есть	Нет
public	Есть	Есть	Есть

Область видимости переменных

1. Переменная, объявленная в методе, существует/видна с начала объявления до конца метода.

2. Переменная, объявленная в блоке кода, существует до конца этого блока кода.

3. Переменные — аргументы метода — существуют везде внутри метода.

4. Переменные класса/объекта существуют все время жизни содержащего их объекта. Их видимость дополнительно регулируется специальными модификаторами доступа: public, private.

5. Статические переменные классов существуют все время работы программы. Их видимость также определяется модификаторами доступа.

Модификаторы:

Доступа

Static Статические переменные называются переменными класса и являются уникальными для всех экземпляров данного класса. Статические методы могут быть вызваны без создания объекта, в котором они описаны

Final фактически служит для переменной указанием на то, что она является константой. Для методов — что они не могут быть переопределены при наследовании, ну а для классов это указание на то, что наследоваться от него нельзя (immutable).

Abstract *применим только к методам и классам*. Абстрактный метод — это метод без реализации (тела).

Если класс помечается как абстрактный, он либо содержит абстрактные методы, либо это делается для того чтобы запретить создание экземпляров этого класса. Если

проводить аналогию, посреди инструкции можно увидеть заголовок «Раскраска объекта», после которого нет описания. Т.е. по этой инструкции можно создать объект и раскрасить его тоже можно, но в конкретно этой инструкции не написано как (пишите свою инструкцию по созданию красного объекта на основе этого объекта и опишите как его раскрашивать).

Чтобы создать неизменяемые поля в классе перед ними будем писать static final – если надо чтобы был доступ из классов других пакетов

Protected final но тогда доступа из других классов не будет

6. Выводы по работе.

В ходе работы я узнал что такое ООП, из чего оно состоит, основные принципы. Научался использовать сторонние библиотеки и подключать из в свою программу. Узнал, кто такие покемоны и что они умеют.