Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа № 4

Вариант № 32126

Группа: P3132

Выполнил: Миняев И.А.

Проверил: Горбунов М.

Санкт-Петербург
2021г

Оглавление

Текст задания

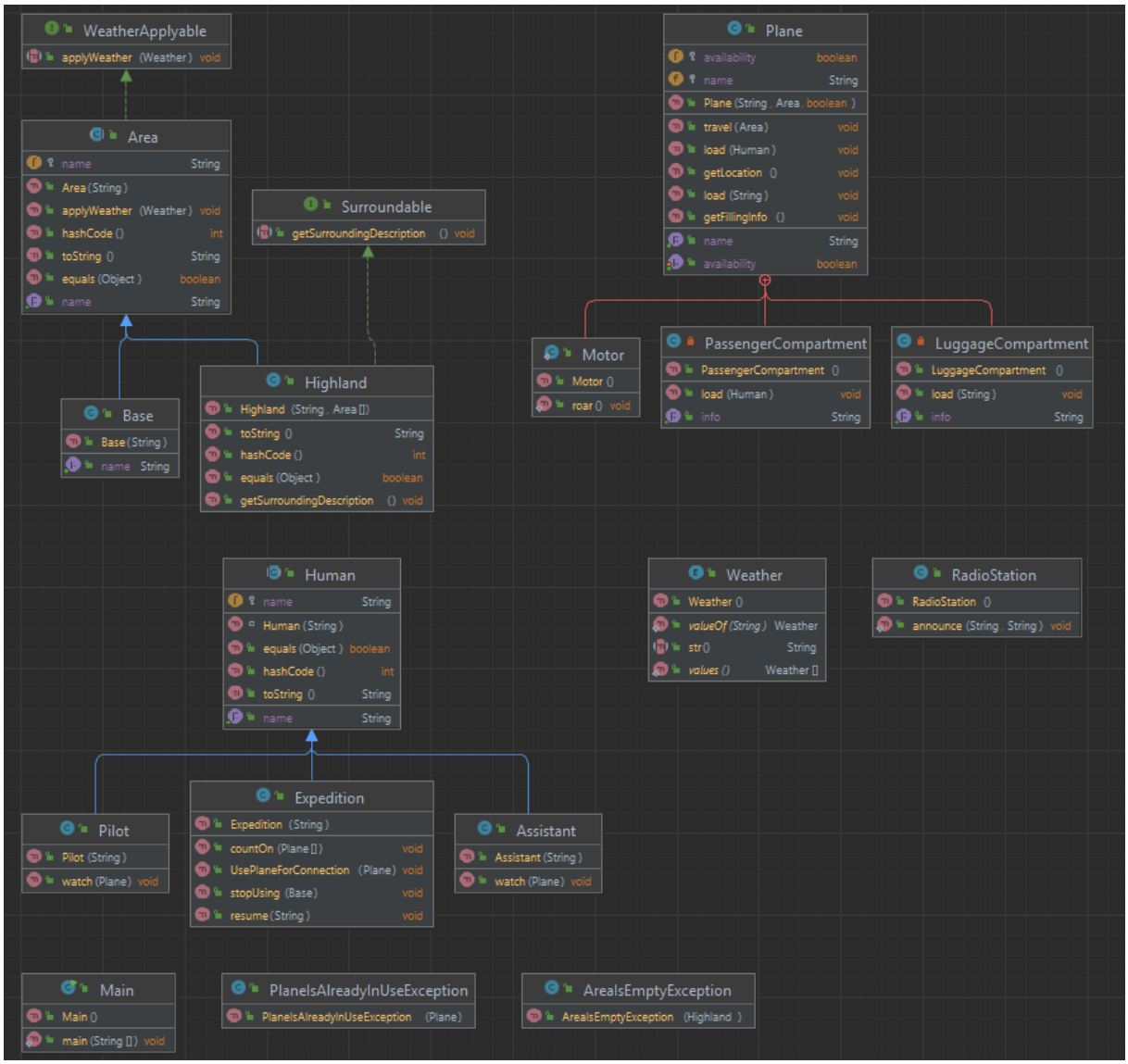
Описание предметной области, по которой должна быть построена объектная модель:

> Экспедиция могла рассчитывать только на четыре самолета, пятый оставался на базе под присмотром летчика и еще двух подручных и в случае пропажи остальных самолетов должен был доставить нас на "Аркхем". Позже, когда какой-нибудь самолет или даже два были свободны от перевозки аппаратуры, мы использовали их для связи: помимо этой основной базы, у нас имелось еще одно временное пристанище на расстоянии шестисот -- семисот миль -- в южной части огромного плоскогорья, рядом с ледником Бирдмора. Несмотря на метели и жесточайшие ветры, постоянно дующие с плоскогорья, мы в целях экономии и эффективности работ отказались от промежуточных баз. В радиосводках от 21 ноября сообщалось о нашем захватывающем беспосадочном полете в течение четырех часов над бескрайней ледяной равниной, окаймленной на западе горной грядой. Рев мотора разрывал вековое безмолвие; ветер не мешал полету, а попав в туман, мы продолжили путь по радиокомпасу.

Порядок выполнения работы:

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями

UML диаграмма классов



Main.java

```
import Exeptions.*;
import People.*;
import Places.*;
import Utility.*;
import Enums.*;
public class Main {
    public static void main(String[] args) throws
PlaneIsAlreadyInUseException {
        Expedition expedition = new Expedition("Экспедиция");
        Pilot pilot = new Pilot("Летчик");
        Assistant assistant1 = new Assistant("Первый");
        Assistant assistant2 = new Assistant("Второй");

        Base mainBase = new Base("Основная");
        Base temporalBase = new Base("Промежуточная");
        Base arhkem = new Base("Аркхем");
        Area birdmor = new Area("Бирдмор"){
            public String getName(){
                return "Ледник " + name;
            }
        };

        Plane plane1 = new Plane("Самолет №1", mainBase, true);
        Plane plane2 = new Plane("Самолет №2", mainBase, true);
        Plane plane3 = new Plane("Самолет №3", mainBase, true);
        Plane plane4 = new Plane("Самолет №4", mainBase, true);
        Plane plane5 = new Plane("Самолет №5", mainBase, false);

        Plane[] availablePlanes = {plane1, plane2, plane3, plane4};
        expedition.countOn(availablePlanes);

        plane5.getLocation();
        assistant1.watch(plane5);
        assistant2.watch(plane5);
        pilot.watch(plane5);

        plane1.load("аппаратура");
        plane1.load(expedition);
        plane2.load("аппаратура");
        plane2.load(expedition);
        plane1.travel(mainBase);
        plane2.travel(mainBase);

        try{
            expedition.UsePlaneForConnection(plane1);
        }catch(PlaneIsAlreadyInUseException e){
            plane1.setAvailability(true);
            expedition.UsePlaneForConnection(plane1);
        }
        try{
            expedition.UsePlaneForConnection(plane2);
        }catch(PlaneIsAlreadyInUseException e){
            plane2.setAvailability(true);
            expedition.UsePlaneForConnection(plane2);
        }

        Area[] vicinity = {temporalBase, birdmor};
        Highland highland = new Highland("Огромное плоскогорье", vicinity);

        highland.getSurroundingDescription();

        highland.applyWeather(Weather.BLIZZARD);
```

```
        highland.applyWeather(Weather.WIND);

        expedition.stopUsing(temporalBase);


        RadioStation.announce("21 ноября",
                "Экспедиция совершила захватывающий беспосадочный " +
                        "полет в течение четырех часов над бескрайней " +
                        "ледяной равниной, окаймленной на западе " +
                        "горной грядой");
        Plane.Motor.roar();
        expedition.resume("путь по радиокомпасу");

    }
}
```

Enums/Weather.java

```
package Enums;
public enum Weather {
    CLEAR{
        @Override
        public String str(){
            return "";
        }
    },
    BLIZZARD{
        @Override
        public String str(){
            return "Метель";
        }
    },
    WIND{
        @Override
        public String str(){
            return "Дует ветер";
        }
    },
    FOG{
        @Override
        public String str() {
            return "Туман";
        }
    };
    public abstract String str();
}
```

Interfaces/Surroundable.java

```
package Interfaces;
public interface Surroundable {
    public void getSurroundingDescription();
}
```

Interface/WeatherApplyable.java

```
package Interfaces;
import Enums.Weather;
public interface WeatherApplyable {
    public void applyWeather(Weather weather);
}
```

People/Human.java

```
package People;

import java.util.Objects;

import Utility.*;
public abstract class Human {
    protected String name;
    public String getName(){
        return name;
    }
    Human(String name){this.name = name;}

    @Override
    public String toString() {
        return "Human{" +
                "name='" + name + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Human human = (Human) o;
        return Objects.equals(name, human.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name);
    }
}
```

People/Assistant.java

```
package People;
import Utility.*;
public class Assistant extends Human{
    public Assistant(String name) {super(name);}

    public void watch(Plane plane){
        System.out.println("Подручный " + name + " наблюдает за " +
plane.getName());

    }
}
```

People/Expedition.java

```
package People;
import Exeptions.PlaneIsAlreadyInUseException;
import Utility.*;
import Places.*;

public class Expedition extends Human{
    public Expedition(String name) {super(name);}

    public void countOn(Plane[] plane){
        System.out.print(name + " рассчитывает на ");
        for(int i = 0; i < plane.length; i++)
            System.out.print(plane[i].getName() +", ");
        System.out.println();
    }
```

```
    public void UsePlaneForConnection(Plane plane) throws
PlaneIsAlreadyInUseException {
        if(plane.getAvailability() == true)
            System.out.println(name + " использует " + plane.getName() + "
для связи");
        else
            throw new PlaneIsAlreadyInUseException(plane);
    }
    public void stopUsing(Base base){
        System.out.println(name + " отказывается от использования " +
base.getName());
    }
    public void resume(String act) {
        System.out.println(name + " продолжает " + act);
    }
}
```

People/Pilot.java

```
package People;
import Utility.*;
public class Pilot extends Human{
    public Pilot(String name) {super(name);}

    public void watch(Plane plane) {
        System.out.println(name + " наблюдает за " + plane.getName());
    }
}
```

Places/Area.java

```
package Places;
import Enums.*;
import Interfaces.*;

import java.util.Arrays;
import java.util.Objects;

public abstract class Area implements WeatherApplyable {
    protected String name;
    protected Weather[] weather;

    public String getName(){
        return name;
    }

    public Area(String name){this.name = name;}
    public void applyWeather(Weather weather){
        System.out.println("На области " + name + " " + weather.str());
    }

    @Override
    public String toString() {
        return "Area{" +
                "name='" + name + '\'' +
                ", weather=" + Arrays.toString(weather) +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
```

```
        Area area = (Area) o;
        return Objects.equals(name, area.name) && Arrays.equals(weather,
area.weather);
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(name);
        result = 31 * result + Arrays.hashCode(weather);
        return result;
    }
}
```

Places/Base.java

```
package Places;

public class Base extends Area{
    public Base(String name) {super(name);}
    @Override
    public String getName(){
        return "База " + name;
    }
}
```

Places/Highland.java

```
package Places;

import Exeptions.AreaIsEmptyException;
import Interfaces.Surroundable;

import java.util.Arrays;

public class Highland extends Area implements Surroundable {
    Area[] vicinity;
    public Highland(String name, Area[] vicinity) {
        super(name);
        if(vicinity.length == 0)
            throw new AreaIsEmptyException(this);
        this.vicinity = vicinity.clone();
    }
    public void getSurroundingDescription(){
        System.out.print("На " + name + " располагаются: ");
        for(int i = 0; i < vicinity.length; i++)
            System.out.print(vicinity[i].getName() + ", ");
        System.out.println();
    }

    @Override
    public String toString() {
        return "Highland{" +
                "vicinity=" + Arrays.toString(vicinity) +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Highland highland = (Highland) o;
        return Arrays.equals(vicinity, highland.vicinity);
```

```
    }

    @Override
    public int hashCode() {
        int result = super.hashCode();
        result = 31 * result + Arrays.hashCode(vicinity);
        return result;
    }
}
```

Utility/Radio.java

```
package Utility;

public class RadioStation {
    public static void announce(String date, String info){
        class Announcer{
            public void saySpeech(){ System.out.println("Радиосводка от " +
date + ": " + info); }
        }
        Announcer announcer = new Announcer();
        announcer.saySpeech();
    }
}
```

Utility/Plane.java

```
package Utility;

import Places.Area;
import People.*;

public class Plane {
    protected String name;
    protected boolean availability = false;
    protected Area currentLocation;

    public boolean getAvailability(){ return availability; }
    public String getName(){return name;}
    public void getLocation(){System.out.println(name + " находится на " +
currentLocation.getName());}

    public void setAvailability(boolean availability){
        this.availability = availability;
        if(availability == true)
            System.out.println("Самолет освобожден");
        else
            System.out.println("Самолет занят");
    }

    public Plane(String name, Area currentLocation, boolean availability){
        this.name = name;
        this.currentLocation = currentLocation;
        this.availability = availability;
    }

    //-----Содержимое самолета------
    private class PassengerCompartment{
        protected Human passenger;
        public void load(Human passenger){ this.passenger = passenger; }
        public String getInfo(){
            return passenger.getName();
        }
```

```
    }
    private class LuggageCompartment{
        protected String loot;
        public void load(String loot){ this.loot = loot; }
        public String getInfo(){
            return loot;
        }
    }
    public static class Motor{
        public static void roar(){
            System.out.println("Мотор ревет");
        }
    }

    PassengerCompartment passC = new PassengerCompartment();
    LuggageCompartment luggC = new LuggageCompartment();

    public void load(Human passenger){ passC.load(passenger);}
    public void load(String loot){ luggC.load(loot);}

    public void getFillingInfo(){ System.out.println("на борту: " +
passC.getInfo() + " и " + luggC.getInfo()); }
    //-------------------------

    public void travel(Area destination){
        System.out.print(name + " переместился из " +
currentLocation.getName() + " в " + destination.getName()  + " и перевез
");
        getFillingInfo();
        currentLocation = destination;
    }

}
```

Результат работы программы

Экспедиция рассчитывает на Самолет №1, Самолет №2, Самолет №3, Самолет №4, Самолет №5 находится на База Основная
Подручный Первый наблюдает за Самолет №5
Подручный Второй наблюдает за Самолет №5
Летчик наблюдает за Самолет №5
Самолет №1 переместился из База Основная в База Основная и перевез на борту: Экспедиция и аппаратура
Самолет №2 переместился из База Основная в База Основная и перевез на борту: Экспедиция и аппаратура
Экспедиция использует Самолет №1 для связи
Экспедиция использует Самолет №2 для связи
На Огромное плоскогорье располагаются: База Промежуточная, Ледник Бирдмор,
На области Огромное плоскогорье Метель
На области Огромное плоскогорье Дует ветер
Экспедиция отказывается от использования База Промежуточная
Радиосводка от 21 ноября: Экспедиция совершила захватывающий беспосадочный полет в течение четырех часов над бескрайней ледяной равниной, окаймленной на западе горной грядой
Мотор ревет
Экспедиция продолжает путь по радиокомпасу

Заключение

Навыки полученные в процессе выполнения работы:

1) Использование исключений
2) Использование локальных, анонимных и вложенных классов