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 №3**

Программирование

Вариант №32193

Группа: P3131

Выполнил: Хайкин О. И.

Проверил:

Блохина Елена Николаевна

Санкт-Петербург

2021г

Оглавление

[Задание 3](#_Toc86667431)

[Диаграмма классов реализованной объектной модели 4](#_Toc86667432)

[Исходный код программы 7](#_Toc86667433)

[Package main 7](#_Toc86667434)

[Package people 11](#_Toc86667435)

[Package davilon 15](#_Toc86667436)

[Package rocket 17](#_Toc86667437)

[Результат работы программы 24](#_Toc86667438)

[Заключение 25](#_Toc86667439)

# Задание

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

Конференция началась с того, что Незнайка рассказал собравшимся все, что знал о ракете, на которой был совершен перелет, об ее устройстве и управлении, после чего ученые и журналисты задавали ему вопросы. Журналисты интересовались главным образом тем, что ел Незнайка, когда находился в ракете, и что пил, какие видел сны и как ему понравились жители Давилона. Вопросы ученых носили несколько иной характер и касались преимущественно того, что видел Незнайка во время своего космического путешествия, что наблюдал на поверхности Луны и как выглядит планета Большая Земля.

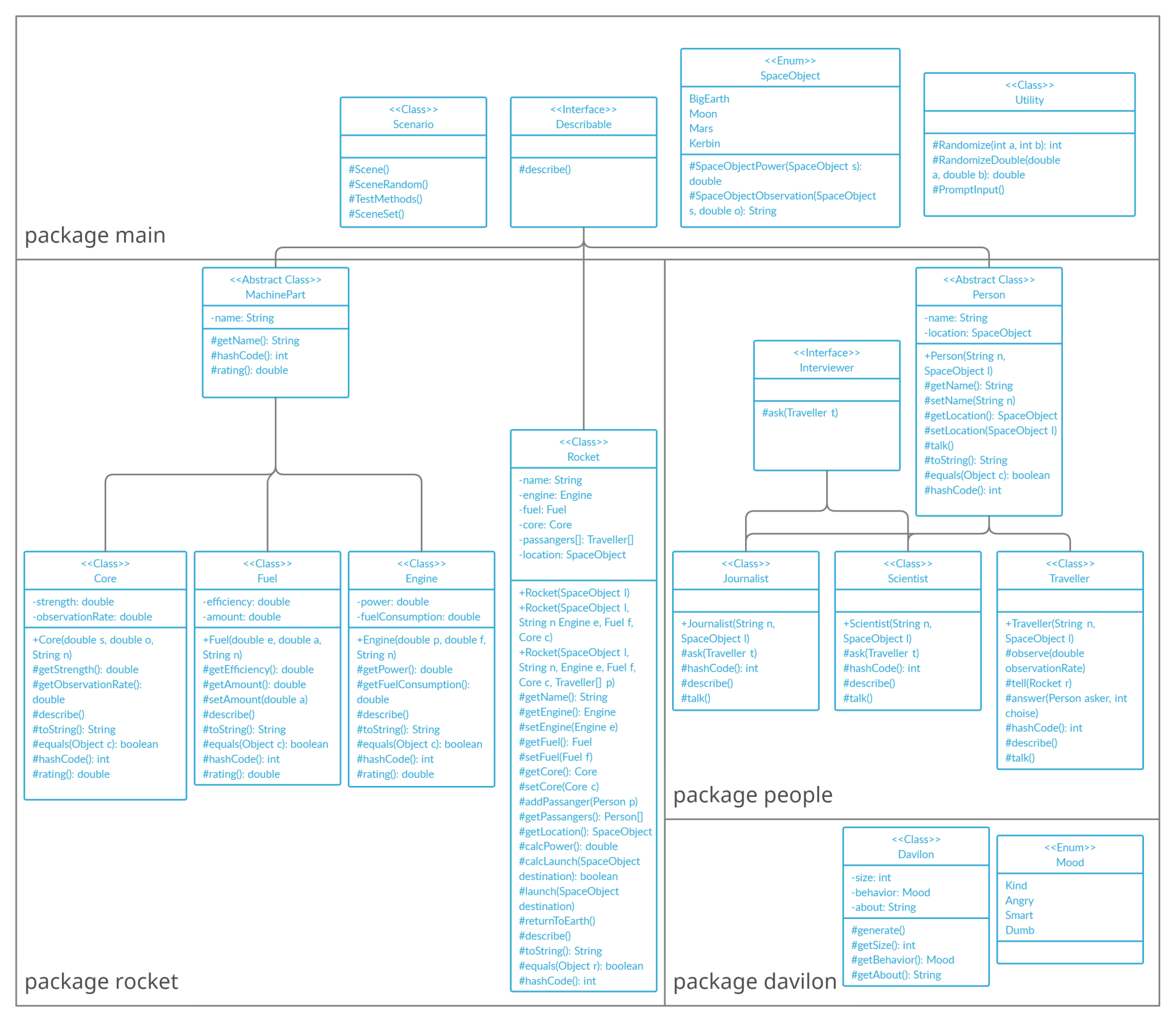
**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

**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.

# Диаграмма классов реализованной объектной модели

Исходный код программы

Lab3.java

import main.\*;

public class lab3 {

public static void main(String[] args) {

Utility.PromptInput();

}

}

## Package main

Describable.java

package main;

public interface Describable {

public void describe();

}

Scenario.java

package main;

import main.\*;

import people.\*;

import rocket.\*;

import davilon.\*;

public class Scenario {

public static void Scene() {

Davilon.generate();

Traveller n = new Traveller("Neznayka", SpaceObject.BigEarth);

Journalist j = new Journalist("Pimanov", SpaceObject.BigEarth);

Scientist s = new Scientist("Hugo", SpaceObject.BigEarth);

Traveller[] t = {n};

Core c = new Core(11.0, 11.0, "Titan GH-73");

Engine e = new Engine(100.0, 1.0, "Kerbodyne KR-2L+ \"Rhino\" Liquid Fuel Engine");

Fuel f = new Fuel(1.0, 100.0, "Liquid fuel \"Ananas\"");

Rocket r = new Rocket(SpaceObject.BigEarth, "Zeus ZF-24", e, f, c, t);

n.describe();

r.launch(SpaceObject.Moon);

if (r.getLocation()!=SpaceObject.BigEarth) {

r.returnToEarth();

n.tell(r);

System.out.println("");

j.ask(n);

s.ask(n);

j.talk();

}

}

public static void SceneRandom() {

Davilon.generate();

Traveller n = new Traveller("Neznayka", SpaceObject.BigEarth);

Journalist j = new Journalist("Pimanov", SpaceObject.BigEarth);

Scientist s = new Scientist("Hugo", SpaceObject.BigEarth);

Traveller[] t = {n};

Core c = new Core(Utility.RandomizeDouble(0, 20), Utility.RandomizeDouble(0, 20), "Titan GH-73");

Engine e = new Engine(Utility.RandomizeDouble(0, 200), Utility.RandomizeDouble(0.5, 1.5), "Kerbodyne KR-2L+ \"Rhino\" Liquid Fuel Engine");

Fuel f = new Fuel(Utility.RandomizeDouble(0.5, 1.5), Utility.RandomizeDouble(0, 200), "Liquid fuel \"Ananas\"");

Rocket r = new Rocket(SpaceObject.BigEarth, "Zeus ZF-24", e, f, c, t);

n.describe();

r.launch(SpaceObject.values()[Utility.Randomize(1, SpaceObject.values().length)]);

if (r.getLocation()!=SpaceObject.BigEarth) {

r.returnToEarth();

n.tell(r);

System.out.println("");

j.ask(n);

s.ask(n);

}

}

public static void TestMethods() {

Davilon.generate();

Traveller n = new Traveller("Neznayka", SpaceObject.BigEarth);

Journalist j = new Journalist("Pimanov", SpaceObject.BigEarth);

Scientist s = new Scientist("Hugo", SpaceObject.BigEarth);

Traveller[] t = {n};

Core c = new Core(11.0, 11.0, "Titan GH-73");

Engine e = new Engine(100.0, 1.0, "Kerbodyne KR-2L+ \"Rhino\" Liquid Fuel Engine");

Fuel f = new Fuel(100.0, 1.0, "Liquid fuel \"Ananas\"");

Rocket r = new Rocket(SpaceObject.BigEarth, "Zeus ZF-24", e, f, c, t);

System.out.println("");

System.out.println("Testing different methods: ");

System.out.println("Rocket.toString(): "+r.toString());

System.out.println("Person.toString(): "+j.toString());

Core c2 = new Core(11.0, 11.0, "Titan GH-73");

System.out.println("Core.equals(): "+c.equals(c2));

System.out.println("Person.equals(): "+n.equals(n));

System.out.println("Person.hashCode(): "+n.hashCode());

System.out.println("Rocket.hashCode(): "+r.hashCode());

System.out.println("c.rating(): "+c.rating());

System.out.println("");

}

public static void SceneSet() {

System.out.println("");

System.out.println("-------------------------------------------------");

System.out.println("");

}

}

SpaceObject.java

package main;

public enum SpaceObject {

BigEarth,

Moon,

Mars,

Kerbin;

public static double SpaceObjectPower(SpaceObject s) {

double power;

switch (s){

case BigEarth: power=0;

break;

case Moon: power=1000;

break;

case Mars: power=6000;

break;

case Kerbin: power=1000000;

break;

default: power=0;

break;

}

return power;

}

public static String SpaceObjectObservation(SpaceObject s, double o) {

String observation;

if (o<10) {

observation="Rocket's illuminators were too bad. I couldn't see anything";

} else {

switch (s) {

case BigEarth: observation="It was really beautiful. Big Earth was so big, I felt smaller than ever before in my life!";

break;

case Moon: observation="Moon was beautiful. At first I thought it for sure was made out of cheese!";

break;

case Mars: observation="It was red and scary.";

break;

case Kerbin: observation="It's really similar to our Big Earth, just not as big. Also, I think I saw a couple other spacecrafts around me?";

break;

default: observation="I was sleeping and didn't see anything... Sorry...";

break;

}

}

return observation;

}

}

Utility.java

package main;

import java.util.Random;

import java.util.Scanner;

public class Utility {

public static int Randomize(int a, int b) {

final Random random = new Random();

int r = random.nextInt(b-a)+a;

return r;

}

public static double RandomizeDouble(double a, double b) {

return Math.random()\*(b-a)+a;

}

public static void PromptInput() {

Scanner wantInput = new Scanner(System.in);

System.out.println("Choose the scenario you wish to see:");

System.out.println("Type \"1\" for a normal scene; type \"2\" for a randomized scene; type \"3\" for a method test; type \"exit\" to exit");

String input = wantInput.nextLine();

Scenario.SceneSet();

switch (input) {

case "1": Scenario.Scene();

break;

case "2": Scenario.SceneRandom();

break;

case "3": Scenario.TestMethods();

break;

case "exit": System.exit(0);

default: System.out.println("Wrong input");

}

Scenario.SceneSet();

Utility.PromptInput();

}

}

## Package people

Interviewer.java

package people;

import people.\*;

public interface Interviewer {

public void ask(Traveller t);

}

Journalist.java

package people;

import main.\*;

public class Journalist extends Person implements Interviewer {

public Journalist (String n, SpaceObject l) {

super(n, l);

}

public void ask(Traveller t) {

if (t.getLocation()==this.getLocation()) {

System.out.print(this.getName()+" asks " + t.getName());

int choise = Utility.Randomize(0, 4);

switch (choise) {

case 0: System.out.println(" about what he was eating while travelling");

break;

case 1: System.out.println(" about what he was drinking while travelling");

break;

case 2: System.out.println(" what he was dreaming about while travelling");

break;

case 3: System.out.println(" wether he liked citizens of Vavilon or not");

break;

default: System.out.println(" how he's feeling");

}

t.answer(this, choise);

} else {

System.out.println(this.getName()+" can't ask "+t.getName()+" anything because they're on different planets!");

}

}

@Override

public int hashCode() {

return super.hashCode()+1;

}

public void describe() {

System.out.println("This is a journalist named "+this.getName());

}

public void talk() {

System.out.println(this.getName()+" says: I'm not getting paid enough for all this...");

}

}

Person.java

package people;

import main.\*;

public abstract class Person implements Describable {

protected String name;

protected SpaceObject location;

public Person(String n, SpaceObject l) {

name=n;

location=l;

}

public String getName() {

return name;

}

public void setName(String n) {

name=n;

}

public SpaceObject getLocation() {

return location;

}

public void setLocation(SpaceObject l) {

location=l;

}

public abstract void talk();

@Override

public String toString() {

return this.getClass().toString().substring(13)+" "+this.getName()+" on "+this.getLocation();

}

@Override

public boolean equals(Object p) {

if (p instanceof Person) {

if (this.getClass()==p.getClass() && this.getName()==((Person) p).getName()) {

return true;

}

}

return false;

}

@Override

public int hashCode() {

int res = this.getName().hashCode()\*10;

return res;

}

}

Scientist.java

package people;

import main.\*;

public class Scientist extends Person implements Interviewer {

public Scientist (String n, SpaceObject l) {

super(n, l);

}

public void ask(Traveller t) {

if (t.getLocation()==this.getLocation()) {

System.out.print(this.getName()+" asks " + t.getName());

int choise = Utility.Randomize(0, 2);

switch (choise) {

case 0: System.out.println(" about Big Earth");

break;

case 1: System.out.println(" about planet he visited");

break;

default: System.out.println(" how he's feeling");

}

t.answer(this, choise);

} else {

System.out.println(this.getName()+" can't ask "+t.getName()+" anything because they're on different planets!");

}

}

@Override

public int hashCode() {

return super.hashCode()+2;

}

public void describe() {

System.out.println("This is a scientist named "+this.getName());

}

public void talk() {

System.out.println(this.getName()+" says: I have so much work to do...");

}

}

Traveller.java

package people;

import main.\*;

import davilon.\*;

import rocket.\*;

public class Traveller extends Person{

protected String[] observations = new String[4];

public Traveller (String s, SpaceObject l) {

super(s, l);

}

public void observe(double observationRate) {

if (this.getLocation()!=SpaceObject.BigEarth) {

observations[1]=SpaceObject.SpaceObjectObservation(this.getLocation(), observationRate);

System.out.println(this.getName()+" meets citizens of Davilon! "+Davilon.getAbout());

} else observations[0]=SpaceObject.SpaceObjectObservation(this.getLocation(), observationRate);

}

public void tell(Rocket r) {

System.out.println(this.getName()+" exits the rocket and says: I'm back! Let me start the story by telling you everything I know about my rocket.");

r.describe();

}

public void answer(Person asker, int choise) {

System.out.print(this.getName()+" answers to "+asker.getName()+": ");

if (observations[0]==null) {

System.out.println("Rocket didn't work...");

} else {

if (asker instanceof Journalist) {

switch (choise) {

case 0: System.out.println("I was eating space food, a special food in tubes prepared for space travel");

break;

case 1: System.out.println("Mostly water.");

break;

case 2: System.out.println("I was dreaming about home.");

break;

case 3: System.out.print("They were really "+Davilon.getBehavior()+". ");

if (Davilon.getBehavior()==Mood.Angry | Davilon.getBehavior()==Mood.Smart) {

System.out.println("I didn't really like them...");

} else System.out.println("I liked them.");

break;

default: System.out.println("I'm good.");

}

} else {

switch (choise) {

case 0: System.out.println(observations[0]);

break;

case 1: System.out.println(observations[1]);

break;

default: System.out.println("I'm good.");

}

}

}

}

@Override

public int hashCode() {

return super.hashCode()\*10+3;

}

public void describe() {

System.out.println("This is a traveller named "+this.getName());

}

public void talk() {

System.out.println(this.getName()+" says: I'm lucky to be where I am today.");

}

}

## Package davilon

Davilon.java

package davilon;

import main.\*;

public class Davilon {

protected static int size;

protected static Mood behavior;

protected static String about = "Citizens of Davilon are ";

public static void generate() {

about="Citizens of Davilon are ";

behavior = Mood.values()[Utility.Randomize(0, Mood.values().length)];

size = Utility.Randomize(100, 230);

about=about.concat(behavior.toString());

int variation = Utility.Randomize(0, 4);

switch (variation){

case 0: about=about.concat(" grey creatures with a lot of fur and 4 legs.");

break;

case 1: about=about.concat(" very round-like purple creatures with very short hands and legs.");

break;

case 2: about=about.concat(" bird-like creatures of different colors.");

break;

case 3: about=about.concat(" almost identical to us creatures.");

break;

default: about=about.concat(" non-existent. There aren't any...");

}

about=about.concat(" Their average height is " + size + "cm");

}

public static int getSize() {

return size;

}

public static Mood getBehavior() {

return behavior;

}

public static String getAbout() {

return about;

}

}

Mood.java

package davilon;

public enum Mood {

Kind,

Angry,

Smart,

Dumb,

}

## Package rocket

Core.java

package rocket;

import main.\*;

public class Core extends MachinePart {

private double strength;

private double observationRate;

public Core(double s, double o, String n) {

strength=s;

observationRate=o;

name=n;

}

public double getStrength() {

return strength;

}

public double getObservationRate() {

return observationRate;

}

public void describe() {

System.out.print("Core called "+this.getName()+". It's hull integrity is "+this.getStrength()+". It's observation rate is "+this.getObservationRate()+".");

}

@Override

public String toString() {

return "CORE "+this.getName()+" with stats: Strength: "+this.getStrength()+" ObservationRate: "+this.getObservationRate();

}

@Override

public boolean equals(Object c) {

if (c instanceof Core && this.getName()==((Core)c).getName() && this.getStrength()==((Core)c).getStrength() && this.getObservationRate()==((Core)c).getObservationRate()) {

return true;

} else return false;

}

@Override

public int hashCode() {

return super.hashCode() + Math.toIntExact(Math.round(this.getStrength()+this.getObservationRate())) + 1;

}

public double rating() {

return strength+observationRate;

}

}

Engine.java

package rocket;

import main.\*;

public class Engine extends MachinePart {

private double power;

private double fuelConsumption;

public Engine(double p, double f, String n) {

power=p;

fuelConsumption=f;

name=n;

}

public double getPower() {

return power;

}

public double getFuelConsumption() {

return fuelConsumption;

}

public void describe() {

System.out.print("Engine called "+this.getName()+". It's power is "+this.getPower()+"W. It's fuel consumption rate is "+this.getFuelConsumption()+"L/s.");

}

@Override

public String toString() {

return "ENGINE "+this.getName()+" with stats: Power: "+this.getPower()+" FuelConsumption: "+this.getFuelConsumption();

}

@Override

public boolean equals(Object e) {

if (e instanceof Engine && this.getName()==((Engine)e).getName() && this.getPower()==((Engine)e).getPower() && this.getFuelConsumption()==((Engine)e).getFuelConsumption()) {

return true;

} else return false;

}

@Override

public int hashCode() {

return super.hashCode() + Math.toIntExact(Math.round(this.getPower()+this.getFuelConsumption())) + 2;

}

public double rating() {

return power+fuelConsumption;

}

}

Fuel.java

package rocket;

import main.\*;

public class Fuel extends MachinePart {

private double efficiency;

private double amount;

public Fuel(double e, double a, String n) {

efficiency=e;

amount=a;

name=n;

}

public double getAmount() {

return amount;

}

public void setAmount(double a) {

amount=a;

}

public double getEfficiency() {

return efficiency;

}

public void describe() {

System.out.print("Fuel called "+this.getName()+". It's efficiency is "+this.getEfficiency()+" and there's "+this.getAmount()+"L of it");

}

@Override

public String toString() {

return "FUEL "+this.getName()+" with stats: Amount: "+this.getAmount()+" Efficiency: "+this.getEfficiency();

}

@Override

public boolean equals(Object f) {

if (f instanceof Fuel && this.getName()==((Fuel)f).getName() && this.getEfficiency()==((Fuel)f).getEfficiency()) {

return true;

} else return false;

}

@Override

public int hashCode() {

return super.hashCode() + Math.toIntExact(Math.round(this.getEfficiency())) + 3;

}

public double rating() {

return efficiency+amount;

}

}

MachinePart.java

package rocket;

import main.\*;

public abstract class MachinePart implements Describable {

protected String name;

public String getName() {

return name;

}

@Override

public int hashCode() {

return this.getName().hashCode();

}

public abstract double rating();

}

Rocket.java

package rocket;

import main.\*;

import people.\*;

public class Rocket implements Describable {

protected String name;

protected Engine engine;

protected Fuel fuel;

protected Core core;

protected Traveller[] passangers;

protected SpaceObject location;

public Rocket(SpaceObject l, String n) {

location=l;

name=n;

}

public Rocket(SpaceObject l, String n, Engine e, Fuel f, Core c) {

location=l;

name=n;

engine=e;

fuel=f;

core=c;

}

public Rocket(SpaceObject l, String n, Engine e, Fuel f, Core c, Traveller[] p) {

location=l;

name=n;

engine=e;

fuel=f;

core=c;

passangers=p;

}

public String getName() {

return name;

}

public Engine getEngine() {

return engine;

}

public Fuel getFuel() {

return fuel;

}

public Core getCore() {

return core;

}

public void setEngine(Engine e) {

engine=e;

}

public void setFuel(Fuel f) {

fuel=f;

}

public void setCore(Core c) {

core=c;

}

public Traveller[] getPassangers() {

return passangers;

}

public SpaceObject getLocation() {

return location;

}

public double calcPower() {

double totalpower = this.getEngine().getPower() \* this.getFuel().getAmount() \* this.getFuel().getEfficiency() / this.getEngine().getFuelConsumption();

return totalpower;

}

public boolean calcLaunch(SpaceObject destination) {

if (this.calcPower()>=SpaceObject.SpaceObjectPower(destination)) {

return true;

} else

return false;

}

public void launch(SpaceObject destination) {

System.out.println("Rocket's destination: "+destination);

if (this.calcLaunch(destination)) {

this.location=destination;

this.getFuel().setAmount(0.0);

System.out.print("Rocket flies off to the "+destination+"! The passanger");

if (passangers.length > 1) {

System.out.print("s are: ");

} else

System.out.print(" is: ");

for (Traveller i : passangers) {

System.out.println(i.getName()+" ");

i.observe(this.getCore().getObservationRate());

i.setLocation(destination);

i.observe(this.getCore().getObservationRate());

}

System.out.println("");

} else

System.out.println("Rocket doesn't have enough power to go to " + destination);

}

public void returnToEarth() {

if (this.location==SpaceObject.BigEarth) {

System.out.println("Rocket is already at Big Earth!");

} else {

System.out.println("Rocket returns to the Big Earth! Journalists and scientists await brave travellers.");

this.location=SpaceObject.BigEarth;

for (Traveller i : passangers) {

i.setLocation(SpaceObject.BigEarth);

}

}

}

public void describe() {

if (this.getLocation()==SpaceObject.BigEarth) {

System.out.print("The rocket is standing on launching pad. ");

}

System.out.print("This rocket is called \"" + this.getName() + "\", it contains ");

engine.describe();

System.out.print(" and ");

core.describe();

System.out.print(" This rocket has ");

fuel.describe();

System.out.println();

}

@Override

public String toString() {

return "ROCKET "+this.getName()+" on "+this.getLocation().toString()+" out of: "+this.getCore().toString()+"; "+this.getEngine().toString()+"; "+this.getFuel().toString();

}

@Override

public boolean equals(Object r) {

if (r instanceof Rocket && this.getName()==((Rocket)r).getName() && this.getCore().equals(((Rocket)r).getCore()) && this.getFuel().equals(((Rocket)r).getFuel()) && this.getEngine().equals(((Rocket)r).getEngine())) {

return true;

} else return false;

}

@Override

public int hashCode() {

return this.getName().hashCode() + this.getFuel().hashCode() + this.getEngine().hashCode() + this.getCore().hashCode();

}

}

# Результат работы программы

Choose the scenario you wish to see:

Type "1" for a normal scene; type "2" for a randomized scene; type "3" for a method test; type "exit" to exit

$ 1

-------------------------------------------------

This is a person named Neznayka who is currently located at BigEarth.

Rocket's destination: Moon

Rocket flies off to the Moon! The passanger is: Neznayka

Neznayka meets citizens of Davilon! Citizens of Davilon are Angry bird-like creatures of different colors. Their average height is 149cm

Rocket returns to the Big Earth! Journalists and scientists await brave travellers.

Neznayka exits the rocket and says: I'm back! Let me start the story by telling you everything I know about my rocket.

The rocket is standing on launching pad. This rocket is called "Zeus ZF-24", it contains Engine called Kerbodyne KR-2L+ "Rhino" Liquid Fuel Engine. It's power is 100.0W. It's fuel consumption rate is 1.0L/s. and Core called Titan GH-73. It's hull integrity is 11.0. It's observation rate is 11.0. This rocket has Fuel called Liquid fuel "Ananas". It's efficiency is 1.0 and there's 0.0L of it

Pimanov asks Neznayka what he was dreaming about while travelling

Neznayka answers to Pimanov: I was dreaming about home.

Hugo asks Neznayka about Big Earth

Neznayka answers to Hugo: It was really beautiful. Big Earth was so big, I felt smaller than ever before in my life!

-------------------------------------------------

Choose the scenario you wish to see:

Type "1" for a normal scene; type "2" for a randomized scene; type "3" for a method test; type "exit" to exit

# Заключение

В ходе выполнения третьей лабораторной работы я узнал принципы SOLID, разобрался в методах класса Object, модификаторах abstract, static и final.