МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант №90006.5

*Выполнил:*

Студент группы P3131

Кадилов Михаил Владимирович

*Преподаватель:*

Николаев Владимир Вячеславович



Санкт-Петербург, 2022

**Содержание**

[Текст 5](#_Toc124269566)

[Диаграмма классов 6](#_Toc124269567)

[Исходный код 7](#_Toc124269568)

[Main.java 7](#_Toc124269569)

[CosmicObject.java 10](#_Toc124269570)

[CosmicObjectType.java 11](#_Toc124269571)

[Planet.java 12](#_Toc124269572)

[Satellite.java 13](#_Toc124269573)

[Star.java 14](#_Toc124269574)

[CosmicFood.java 15](#_Toc124269575)

[Cutlet.java 17](#_Toc124269576)

[Sausage.java 18](#_Toc124269577)

[Soup.java 19](#_Toc124269578)

[TomatoJuice.java 20](#_Toc124269579)

[Edible.java 21](#_Toc124269580)

[FoodType.java 22](#_Toc124269581)

[NonEdible.java 23](#_Toc124269582)

[CurrentLocationException.java 24](#_Toc124269583)

[EmptyArgumentException.java 25](#_Toc124269584)

[Human.java 26](#_Toc124269585)

[Neznayka.java 29](#_Toc124269586)

[Ponchik.java 30](#_Toc124269587)

[Rocket.java 31](#_Toc124269588)

[RocketLaunch.java 33](#_Toc124269589)

[Результат работы 34](#_Toc124269590)

[Продукт борщ (красный) в количестве 10 загружен в ракету. 34](#_Toc124269591)

[Продукт борщ (оранжевый) в количестве 10 загружен в ракету. 34](#_Toc124269592)

[Продукт Котлета по-киевски в количестве 12 загружен в ракету. 34](#_Toc124269593)

[Продукт Баварская колбаска к пиву в количестве 11 загружен в ракету. 34](#_Toc124269594)

[Продукт Томатный сок в количестве 10 загружен в ракету. 34](#_Toc124269595)

[Проверка equals, hashCode, toString для еды. 34](#_Toc124269596)

[true 34](#_Toc124269597)

[-1548755124 34](#_Toc124269598)

[Космическая еда под назавнием:'Томатный сок', калорийность:60, вес:120.0, количество порций:10 34](#_Toc124269599)

[Незнайка появился в ракете. 34](#_Toc124269600)

[Пончик появился в ракете. 34](#_Toc124269601)

[Проверка equals, hashCode, toString для персонажей. 34](#_Toc124269602)

[false 34](#_Toc124269603)

[2014311149 34](#_Toc124269604)

[Персонаж под именем 'Незнайка' находится в локации:'главная палуба', сыт на 2100, находится в сознаии: true 34](#_Toc124269605)

[Незнайка перешёл из главная палуба в пищеблок. 34](#_Toc124269606)

[Пончик перешёл из главная палуба в пищеблок. 34](#_Toc124269607)

[Незнайка не голоден. 34](#_Toc124269608)

[Незнайка сыт на 2100/2500. 34](#_Toc124269609)

[Пончик сыт на 350/2500. 34](#_Toc124269610)

[Незнайка съел Котлета по-киевски в количестве: 1 34](#_Toc124269611)

[Незнайка не голоден. 34](#_Toc124269612)

[Незнайка сыт на 2300/2500. 34](#_Toc124269613)

[Внимание! На ракете закончился продукт: Баварская колбаска к пиву 34](#_Toc124269614)

[Пончик объелся. Он не доел Баварская колбаска к пиву. 35](#_Toc124269615)

[Пончик не голоден. 35](#_Toc124269616)

[Пончик сыт на 2500/2500. 35](#_Toc124269617)

[Незнайка хочет спать? false 35](#_Toc124269618)

[Пончик хочет спать? true 35](#_Toc124269619)

[Пончик заснул c едой во рту. 35](#_Toc124269620)

[Process finished with exit code 0 35](#_Toc124269621)

[Вывод 36](#_Toc124269622)

# Текст

Друзья быстро поднялись в астрономическую кабину. Посмотрев в боковые иллюминаторы, они увидели вокруг бездонное черное небо, усеянное крупными звездами, среди которых сияло ослепительно яркое солнце. Казалось, был день, но в то же время была и ночь. Так на Земле никогда не бывает. Когда на Земле видно солнце, то не видно звезд, и, наоборот, когда есть звезды -- нет солнца. В одном из верхних иллюминаторов ярко светилась Луна. Она казалась несколько крупнее, чем обычно кажется нам с Земли. Теперь, когда Пончик окончательно убедился, что о возвращении на Землю не может быть никакой речи, он понемногу успокоился и сказал: Друзья спустились в пищевой отсек. Незнайке совсем еще не хотелось есть, и он только для того, чтоб составить компанию Пончику, съел одну космическую котлетку. Но Пончик решил не теряться в создавшейся обстановке и отнесся к делу со всей серьезностью. Он заявил, что должен произвести в пищевом отсеке ревизию и проверить качество всех космических блюд, а для этого ему нужно съесть хотя бы по одной порции каждого блюда. Эта задача оказалась, однако, для него не под силу, потому что уже на десятой или на одиннадцатой порции его сморил сон, и Пончик заснул с недоеденной космической сосиской во рту. В этом ничего удивительного не было, так как ночью Пончик спал мало, к тому же каждый, кто находится в состоянии невесомости, может заснуть в любой позе, не укладываясь для этого специально в постель. Зная, что Пончик всю ночь прокувыркался в поисках выхода из ракеты, Незнайка решил дать ему отдохнуть, а сам отправился в астрономическую кабину, чтобы взглянуть, насколько приблизился космический корабль к Луне. В иллюминаторах по-прежнему чернело небо со звездами, с ярко сверкающим диском солнца и серебристой, светящейся Луной сверху. Солнце было такого же размера, каким оно обычно видно с Земли, но Луна сделалась уже вдвое больше. Незнайке казалось, что он замечает на поверхности Луны такие подробности, которых не замечал раньше, но так как прежде он никогда не смотрел на Луну внимательно, то и не мог сказать с уверенностью, видит ли он эти подробности потому, что подлетел к Луне ближе, или он видит их потому, что теперь стал смотреть на Луну внимательнее. Хотя ракета мчалась со страшной скоростью, покрывая пространство в двенадцать километров в одну секунду, Незнайке казалось, что она застыла на месте и ни на полпальца не приближается к Луне. Это объяснялось тем, что расстояние от Земли до Луны очень большое -- около четырехсот тысяч километров. При таком огромном расстоянии скорость двенадцать километров в секунду не так велика, чтоб ее можно было заметить на глаз, да еще находясь в ракете.

# Диаграмма классов

# 

# 

# 

# 

# Исходный код

#### Main.java

import cosmic\_objects.Planet;  
import rocket.Rocket;  
import cosmic\_objects.Satellite;  
import cosmic\_objects.Star;  
import people.\*;  
import food.\*;  
import rocket.RocketLaunch;  
  
public class Main {  
 public static void main(String[] args) {  
  
  
  
 Soup soup1 = new Soup("борщ (красный)", 10);  
 Soup soup2 = new Soup("борщ (оранжевый)", 10);  
 Cutlet cutlet = new Cutlet("Котлета по-киевски", 12);  
 Sausage sausage = new Sausage("Баварская колбаска к пиву", 11);  
 TomatoJuice tomatoJuice = new TomatoJuice("Томатный сок", 10);  
 System.*out*.println();  
  
  
 System.*out*.println("Проверка equals, hashCode, toString для еды.");  
 System.*out*.println(soup1.equals(soup2));  
 System.*out*.println(soup1.hashCode());  
 System.*out*.println(tomatoJuice);  
 System.*out*.println();  
  
  
 Neznayka neznayka = new Neznayka("главная палуба", 2100);  
 Ponchik ponchik = new Ponchik("главная палуба", 350);  
 System.*out*.println();  
  
  
 System.*out*.println("Проверка equals, hashCode, toString для персонажей.");  
 System.*out*.println(neznayka.equals(ponchik));  
 System.*out*.println(ponchik.hashCode());  
 System.*out*.println(neznayka);  
 System.*out*.println();  
  
 RocketLaunch rocketLaunch = new RocketLaunch() { //Anonymous class  
  
 Rocket.OnBoardComputer onBoardComputer = new Rocket.OnBoardComputer();  
 Rocket rocket = new Rocket();  
 Rocket.Engine engine1 = rocket.new Engine("Двигатель 1");  
 Rocket.Engine engine2 = rocket.new Engine("Двигатель 2");  
 Rocket.Engine engine3 = rocket.new Engine("Двигатель 3");  
 @Override  
 public void launchRocket() {  
 onBoardComputer.startOnBoardComputer();  
 engine1.startEngine();  
 engine2.startEngine();  
 engine3.startEngine();  
 System.*out*.println();  
 }  
  
 public void turnOffRocket() {  
 engine1.stopEngine();  
 engine2.startEngine();  
 engine3.startEngine();  
 onBoardComputer.stopOnBoardComputer();  
 System.*out*.println();  
 }  
 };  
  
 rocketLaunch.launchRocket();  
  
 try {  
 neznayka.go("астрономическая кабина");  
 ponchik.go("астрономическая кабина");  
 } catch (EmptyArgumentException e) {  
 System.*out*.println(e.getMessage());  
 }  
  
 Star sun = new Star("Солнце", 1440000000);  
 Star star1 = new Star("Звезда1", 995444666);  
 Star star2 = new Star("Звезда2", 955844606);  
 Star star3 = new Star("Звезда3", 915847654);  
 Satellite moon = new Satellite("Луна", 200000);  
 Planet earth = new Planet("Земля", 200000);  
  
 System.*out*.println();  
 neznayka.seeCosmicObject(sun);  
 neznayka.seeCosmicObject(star1);  
 neznayka.seeCosmicObject(star2);  
 neznayka.seeCosmicObject(star3);  
 ponchik.seeCosmicObject(sun);  
 ponchik.seeCosmicObject(star1);  
 ponchik.seeCosmicObject(star2);  
 ponchik.seeCosmicObject(star3);  
  
 moon.shine();  
 System.*out*.println();  
  
 neznayka.checkDistance(moon);  
 System.*out*.println();  
  
 ponchik.say("Ну что ж, поскольку мы летим на Луну и назад все пути отрезаны, теперь у нас только одна задача: пробраться обратно в пищевой отсек и как следует позавтракать.");  
  
 try {  
 neznayka.go("пищеблок");  
 ponchik.go("пищеблок");  
 } catch (EmptyArgumentException e) {  
 System.*out*.println(e.getMessage());  
 }  
 System.*out*.println();  
  
 neznayka.showHunger();  
 ponchik.showHunger();  
 System.*out*.println();  
  
 try {  
 neznayka.eat(cutlet, 1);  
 ponchik.eat(sausage, 11);  
 } catch (EmptyArgumentException e) {  
 System.*out*.println(e.getMessage());  
 }  
 neznayka.showHunger();  
 ponchik.showHunger();  
 System.*out*.println();  
  
  
 System.*out*.println("Незнайка хочет спать? " + neznayka.getSleepy());  
 System.*out*.println("Пончик хочет спать? " + ponchik.getSleepy());  
 System.*out*.println();  
  
 ponchik.sleep();  
 System.*out*.println();  
  
 try {  
 neznayka.go("астрономическая кабина");  
 System.*out*.println();  
 } catch (EmptyArgumentException e) {  
 System.*out*.println(e.getMessage());  
 }  
  
  
 sun.shine();  
 moon.shine();  
 System.*out*.println();  
  
 neznayka.checkDistance(sun);  
 neznayka.checkDistance(moon);  
 System.*out*.println();  
  
 Rocket.OnBoardComputer.*checkCurrentSpeed*();  
 Rocket.OnBoardComputer.*checkCosmicInformation*(moon, earth);  
  
 }  
}

#### 

#### CosmicObject.java

package cosmic\_objects;  
  
public abstract class CosmicObject {  
 private String name;  
 private int distanceToRocket;  
 private CosmicObjectType type;  
  
 static private boolean *firstCheck* = true;  
  
 public CosmicObject (String name, int distanceToRocket) {  
 this.name = name;  
 this.distanceToRocket = distanceToRocket;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public CosmicObjectType getType() {  
 return type;  
 }  
  
 public void setType(CosmicObjectType type) {  
 this.type = type;  
 }  
  
 public int getDistanceToRocket() {  
 return distanceToRocket;  
 }  
  
 public void setDistanceToRocket(int distanceToRocket) {  
 this.distanceToRocket = distanceToRocket;  
 }  
  
 abstract void shine();  
  
 public void checkObject () {  
 *firstCheck* = false;  
 }  
  
 public boolean checkFirstCheck () {  
 return *firstCheck*;  
 }  
}

#### CosmicObjectType.java

package cosmic\_objects;  
  
public enum CosmicObjectType {  
 *PLANET*,  
 *STAR*,  
 *SATELLITE*}

#### Planet.java

package cosmic\_objects;  
  
public class Planet extends CosmicObject{  
 public Planet(String name, int distanceToRocket) {  
 super(name, distanceToRocket);  
 setType(CosmicObjectType.*PLANET*);  
 }  
  
 public void shine() {  
 System.*out*.println("Планета " + getName() + "' не может сиять, разве что отражать свет.");  
 }  
}

#### Satellite.java

package cosmic\_objects;  
  
public class Satellite extends CosmicObject {  
 public Satellite(String name, int distanceToRocket) {  
 super(name, distanceToRocket);  
 setType(CosmicObjectType.*SATELLITE*);  
 }  
  
 public void shine() {  
 System.*out*.println("Спутник '" + getName() + "' светилcя в одном из иллюминаторов.");  
 }  
  
}

#### Star.java

package cosmic\_objects;  
  
public class Star extends CosmicObject{  
 public Star(String name, int distanceToRocket) {  
 super(name, distanceToRocket);  
 setType(CosmicObjectType.*STAR*);  
 }  
  
 public void shine() {  
 System.*out*.println("Звезда '" + getName() + "' сияла в иллюминаторе.");  
 }  
}

#### CosmicFood.java

package food;  
  
import java.util.Objects;  
  
public abstract class CosmicFood{  
 private String name;  
 private int calories;  
 private double weight;  
 private int numberOfServings;  
  
 protected CosmicFood() {  
 }  
  
 public String getName(){  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getCalories(){  
 return calories;  
 }  
  
 public void setCalories(int calories) {  
 this.calories = calories;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public void setWeight(double weight) {  
 this.weight = weight;  
 }  
  
 public int getNumberOfServings(){  
 return numberOfServings;  
 }  
  
 public void setNumberOfServings(int numberOfServings) {  
 this.numberOfServings = numberOfServings;  
 }  
  
 public void eat(int amount){  
 numberOfServings -= amount;  
  
 if (numberOfServings == 0){  
 System.*out*.println("Внимание! На ракете закончился продукт: " + name);  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Космическая еда " +  
 "под назавнием:'" + name + '\'' +  
 ", калорийность:" + calories +  
 ", вес:" + weight +  
 ", количество порций:" + numberOfServings;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, calories, weight, numberOfServings);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 CosmicFood that = (CosmicFood) o;  
 return calories == that.calories;  
 }  
  
 public abstract FoodType getType();  
  
}

#### Cutlet.java

package food;  
  
public class Cutlet extends CosmicFood implements Edible{  
 public Cutlet(String name, int numberOfServings){  
 setName(name);  
 setNumberOfServings(numberOfServings);  
 setWeight(180);  
 setCalories(200);  
 System.*out*.println("Продукт " + name + " в количестве " + numberOfServings + " загружен в ракету.");  
 }  
  
 public FoodType getType() {  
 return FoodType.*SECOND*;  
 }  
  
 @Override  
 public void use() {  
 if (getNumberOfServings() > 0) {  
 setNumberOfServings(getNumberOfServings() - 1);  
 System.*out*.println("Съеден(а) 1 " + getName());  
 } else {  
 System.*out*.println("Есть уже нечего...");  
 }  
 }  
}

#### Sausage.java

package food;  
  
public class Sausage extends CosmicFood implements Edible{  
 public Sausage(String name, int numberOfServings){  
 setName(name);  
 setNumberOfServings(numberOfServings);  
 setWeight(170);  
 setCalories(210);  
 System.*out*.println("Продукт " + name + " в количестве " + numberOfServings + " загружен в ракету.");  
 }  
  
 public FoodType getType() {  
 return FoodType.*SECOND*;  
 }  
  
 @Override  
 public void use() {  
 if (getNumberOfServings() > 0) {  
 setNumberOfServings(getNumberOfServings() - 1);  
 System.*out*.println("Съеден(а) 1 " + getName());  
 } else {  
 System.*out*.println("Есть уже нечего...");  
 }  
 }  
  
}

#### Soup.java

package food;  
  
public class Soup extends CosmicFood implements Edible{  
 public Soup(String name, int numberOfServings){  
 setName(name);  
 setNumberOfServings(numberOfServings);  
 setWeight(300);  
 setCalories(180);  
 System.*out*.println("Продукт " + name + " в количестве " + numberOfServings + " загружен в ракету.");  
 }  
  
 public FoodType getType() {  
 return FoodType.*FIRST*;  
 }  
  
 @Override  
 public void use() {  
 if (getNumberOfServings() > 0) {  
 setNumberOfServings(getNumberOfServings() - 1);  
 System.*out*.println("Съеден(а) 1 " + getName());  
 }  
 else {  
 System.*out*.println("Есть уже нечего...");  
 }  
 }  
}

#### TomatoJuice.java

package food;  
  
public class TomatoJuice extends CosmicFood implements NonEdible {  
 public TomatoJuice(String name, int numberOfServings) {  
 setName(name);  
 setNumberOfServings(numberOfServings);  
 setWeight(120);  
 setCalories(60);  
 System.*out*.println("Продукт " + name + " в количестве " + numberOfServings + " загружен в ракету.");  
 }  
  
 public FoodType getType() {  
 return FoodType.*DRINK*;  
 }  
  
  
 @Override  
 public void throwAway() {  
 if (getNumberOfServings() > 0) {  
 setNumberOfServings(getNumberOfServings() - 1);  
 System.*out*.println("Выкинул " + getName() + ". Кто вообще ЭТО ест?");  
 } else {  
 System.*out*.println("Какой-то добрый человек уже выкинул эту гадость за тебя.");  
 }  
 }  
}

#### Edible.java

package food;  
  
public interface Edible {  
 void use();  
}

#### FoodType.java

package food;  
  
public enum FoodType {  
 *FIRST*,  
 *SECOND*,  
 *SALAD*,  
 *DRINK*}

#### NonEdible.java

package food;  
  
public interface NonEdible {  
 void throwAway();  
}

#### CurrentLocationException.java

package people;  
  
public class CurrentLocationException extends RuntimeException{  
 public CurrentLocationException(String message) {  
 super(message);  
 }  
}

#### EmptyArgumentException.java

package people;  
  
public class EmptyArgumentException extends Exception{  
 public EmptyArgumentException(String message) {  
 super(message);  
 }  
}

#### Human.java

package people;  
import cosmic\_objects.CosmicObject;  
import cosmic\_objects.CosmicObjectType;  
import food.CosmicFood;  
  
import java.util.Objects;  
  
public abstract class Human{  
 private String name;  
 private String location;  
 private int satiety;  
 private boolean conscious = true;  
 private boolean sleepy;  
  
 public String getName(){  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setSleepy(boolean sleepy) {  
 this.sleepy = sleepy;  
 }  
 public boolean getSleepy() {  
 return sleepy;  
 }  
  
 public void setLocation (String location) {  
 this.location = location;  
 }  
 public String getLocation (){  
 return location;  
 }  
  
 public void setSatiety(int satiety) {  
 this.satiety = satiety;  
 }  
  
 public int getSatiety() {  
 return satiety;  
 }  
  
 public void setConscious(boolean conscious) {  
 this.conscious = conscious;  
 }  
  
 public boolean getConscious() {  
 return conscious;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Human human = (Human) o;  
 return name.equals(human.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, location, satiety, conscious);  
 }  
  
 @Override  
 public String toString() {  
 return "Персонаж " +  
 "под именем '" + name + '\'' +  
 " находится в локации:'" + location + '\'' +  
 ", сыт на " + satiety +  
 ", находится в сознаии: " + conscious;  
 }  
  
 public void eat(CosmicFood dish, int amount) throws EmptyArgumentException {  
 if (dish != null && amount != 0) {  
 if (dish.getNumberOfServings() - amount >= 0) {  
 dish.eat(amount);  
 if (satiety + dish.getCalories() \* amount >= 2500) {  
 satiety += dish.getCalories() \* amount;  
 dish.setNumberOfServings(dish.getNumberOfServings() + (satiety - 2500) / dish.getCalories());  
 if ((satiety - 2500) / dish.getCalories() != 0) {  
 System.*out*.println(name + " объелся. Он не доел " + dish.getName() + " в количестве " + (satiety - 2500) / dish.getCalories() + ". Эта еда вернулась на склад.");  
 } else System.*out*.println(name + " объелся. Он не доел " + dish.getName() + ".");  
 satiety = 2500;  
  
 } else {  
 satiety += dish.getCalories();  
 System.*out*.println(name + " съел " + dish.getName() + " в количестве: " + amount);  
 }  
 } else {  
 System.*out*.println("На ракете нет такого количества еды...");  
 }  
 }  
 else throw new EmptyArgumentException("Передан пустой аргумент!");  
  
 }  
  
 public void go(String place) throws CurrentLocationException, EmptyArgumentException{  
 if ("".equals(place)) {  
 throw new EmptyArgumentException("Передан пустой аргумент!");  
 }  
 else if (getLocation().equals(place)) {  
 throw new CurrentLocationException(getName() + " уже находится в этой локации!");  
 }  
 else {  
 System.*out*.println(name + " перешёл из " + location + " в " + place + ".");  
 location = place;  
 }  
  
 }  
  
 public abstract void sleep();  
  
 public void wakeUp() {  
 setConscious(true);  
 System.*out*.println(getName() + " проснулся.");  
 }  
  
 public void showHunger() {  
 if (satiety > 2000) {  
 System.*out*.println(name + " не голоден.");  
 System.*out*.println(name + " сыт на " + satiety + "/2500.");  
  
 }  
 else {  
 System.*out*.println(name + " сыт на " + satiety + "/2500.");  
 }  
 }  
  
 public void seeCosmicObject(CosmicObject cosmicObject) {  
 System.*out*.println(getName() + " увидел космический объект '" + cosmicObject.getName() + "'.");  
 }  
  
 public void say(String speech) {  
 System.*out*.println(getName() + " сказал: " + speech);  
 }  
  
 public void checkDistance(CosmicObject cosmicObject) {  
 if (cosmicObject.getType() == CosmicObjectType.*STAR*){  
 System.*out*.println(getName() + " оценил расстояне до космического объекта '" + cosmicObject.getName() + "'. Кажется, что расстояние осталось прежним.");  
 } else if (cosmicObject.getName().equals("Луна")) {  
 if (cosmicObject.checkFirstCheck()) {  
 System.*out*.println("Луна казалась несколько крупнее, чем обычно кажется нам с Земли.");  
 cosmicObject.checkObject();  
 }  
 else {  
 System.*out*.println(getName() + " посмотрел на Луну. Кажется, что Луна стала в два раза больше.");  
 }  
 } else if (cosmicObject.getName().equals("Земля")) {  
 System.*out*.println(getName() + " оценил расстояне до Земли . Кажется, что Земля стала значительно меньше.");  
 }  
 else System.*out*.println(getName() + " оценил расстояне до космического объекта '" + cosmicObject.getName() + "'. Кажется, что расстояние не изменилось");  
 }  
}

#### Neznayka.java

package people;  
  
  
public class Neznayka extends Human{  
  
 public Neznayka(String location, int satiety) {  
 setLocation(location);  
 setSatiety(satiety);  
 setName("Незнайка");  
 setSleepy(false);  
 System.*out*.println(getName() + " появился в ракете.");  
 }  
  
 public void sleep(){  
 System.*out*.println(getName() + " заснул.");  
 setSleepy(false);  
 setConscious(false);  
 }  
  
  
}

#### Ponchik.java

package people;  
  
  
public class Ponchik extends Human{  
 public Ponchik(String location, int satiety) {  
 setLocation(location);  
 setSatiety(satiety);  
 setName("Пончик");  
 setSleepy(true);  
 System.*out*.println(getName() + " появился в ракете.");  
 }  
  
 public void sleep(){  
 if (getSatiety() >= 2500) {  
 System.*out*.println(getName() + " заснул c едой во рту.");  
 }  
 else {  
 System.*out*.println(getName() + " заснул.");  
 }  
 setSleepy(false);  
 setConscious(false);  
 }  
}

#### Rocket.java

package rocket;  
  
import cosmic\_objects.CosmicObject;  
  
public class Rocket {  
 private static double *currentSpeed* = 0;  
 private final double enginePower = 180000;  
 private double totalPower = 0;  
  
 public static class OnBoardComputer { //static nested class  
 private static boolean *workMode* = false;  
 public void startOnBoardComputer() {  
 if (!*workMode*) {  
 *workMode* = true;  
 System.*out*.println("Бортовой компьютер запущен.");  
 }  
 else System.*out*.println("Бортовой компьютер уже запущен.");  
 }  
 public void stopOnBoardComputer() {  
 if (*workMode*) {  
 *workMode* = false;  
 System.*out*.println("Бортовой компьютер выключен.");  
 } else System.*out*.println("Бортовой компьютер уже отключен.");  
 }  
  
 public static void checkCurrentSpeed() {  
 System.*out*.println("Текущая скорость ракеты - " + *currentSpeed* + " километров в секунду.");  
 }  
  
 public static void checkCosmicInformation(CosmicObject cosmicObject1, CosmicObject cosmicObject2) {  
 if ((cosmicObject1.getName().equals("Земля") && cosmicObject2.getName().equals("Луна")) || (cosmicObject1.getName().equals("Луна") && cosmicObject2.getName().equals("Земля"))) {  
 System.*out*.println("Расстояние от Земли до Луны очень большое - около четырехсот тысяч километров. При таком огромном расстоянии скорость двенадцать километров в секунду не так велика, чтоб ее можно было заметить на глаз, да еще находясь в ракете.");  
 System.*out*.println("При такой скорости путь от Земли до Луны займет около " + 400000 / 12 / 3600 + " часов.");  
 } else if (cosmicObject1.equals(cosmicObject2)) {  
 System.*out*.println("В качестве аргументов указна один объект. Системная информация отсутсвует.");  
 }  
 else {  
 System.*out*.println("Бортовой компьюте не обладает информацие по указанным объектам: " + cosmicObject1.getName() + ", " + cosmicObject2.getName());  
 }  
 }  
 }  
  
 public class Engine { //Non-static nested class (inner class)  
  
 private final String name;  
 private boolean ignition;  
  
 public Engine (String name) {  
 this.name = name;  
 ignition = false;  
 }  
 public void startEngine() {  
 if (!ignition) {  
 class CombustionChamber {  
 void addOxidizer() {  
 System.*out*.println("Впрыск окислителя...");  
 }  
 void addReducingAgent() {  
 System.*out*.println("Впрыск окислителя...");  
 }  
 void ignite() {  
 System.*out*.println("Зажигание...");  
 ignition = true;  
 totalPower += enginePower;  
 *currentSpeed* += 4;  
 }  
 }  
  
 CombustionChamber combustionChamber = new CombustionChamber();  
 combustionChamber.addOxidizer();  
 combustionChamber.addReducingAgent();  
 combustionChamber.ignite();  
  
 System.*out*.println("Двигатель '" + name + "' запущен. Суммарная мощность двигателей - " + totalPower + " киловатт.");  
 }  
 else System.*out*.println("Двигатель '" + name + "' уже запущен.");  
 }  
 public void stopEngine() {  
 if (ignition) {  
 ignition = false;  
 totalPower -= enginePower;  
 *currentSpeed* -= 4;  
 System.*out*.println("Двигатель '" + name + "' остановлен.");  
 }  
 else System.*out*.println("Двигатель '" + name + "' уже отключен.");  
 }  
 }  
}

#### RocketLaunch.java

package rocket;  
  
public interface RocketLaunch {  
 void launchRocket();  
}

# Результат работы

# Продукт борщ (красный) в количестве 10 загружен в ракету.

# Продукт борщ (оранжевый) в количестве 10 загружен в ракету.

# Продукт Котлета по-киевски в количестве 12 загружен в ракету.

# Продукт Баварская колбаска к пиву в количестве 11 загружен в ракету.

# Продукт Томатный сок в количестве 10 загружен в ракету.

# Проверка equals, hashCode, toString для еды.

# true

# -1548755124

# Космическая еда под назавнием:'Томатный сок', калорийность:60, вес:120.0, количество порций:10

# Незнайка появился в ракете.

# Пончик появился в ракете.

# Проверка equals, hashCode, toString для персонажей.

# false

# 2014311149

# Персонаж под именем 'Незнайка' находится в локации:'главная палуба', сыт на 2100, находится в сознаии: true

# Бортовой компьютер запущен.

# Впрыск окислителя...

# Впрыск окислителя...

# Зажигание...

# Двигатель 'Двигатель 1' запущен. Суммарная мощность двигателей - 180000.0 киловатт.

# Впрыск окислителя...

# Впрыск окислителя...

# Зажигание...

# Двигатель 'Двигатель 2' запущен. Суммарная мощность двигателей - 360000.0 киловатт.

# Впрыск окислителя...

# Впрыск окислителя...

# Зажигание...

# Двигатель 'Двигатель 3' запущен. Суммарная мощность двигателей - 540000.0 киловатт.

# Незнайка перешёл из главная палуба в астрономическая кабина.

# Пончик перешёл из главная палуба в астрономическая кабина.

# Незнайка увидел космический объект 'Солнце'.

# Незнайка увидел космический объект 'Звезда1'.

# Незнайка увидел космический объект 'Звезда2'.

# Незнайка увидел космический объект 'Звезда3'.

# Пончик увидел космический объект 'Солнце'.

# Пончик увидел космический объект 'Звезда1'.

# Пончик увидел космический объект 'Звезда2'.

# Пончик увидел космический объект 'Звезда3'.

# Спутник 'Луна' светилcя в одном из иллюминаторов.

# Луна казалась несколько крупнее, чем обычно кажется нам с Земли.

# Пончик сказал: Ну что ж, поскольку мы летим на Луну и назад все пути отрезаны, теперь у нас только одна задача: пробраться обратно в пищевой отсек и как следует позавтракать.

# Незнайка перешёл из астрономическая кабина в пищеблок.

# Пончик перешёл из астрономическая кабина в пищеблок.

# Незнайка не голоден.

# Незнайка сыт на 2100/2500.

# Пончик сыт на 350/2500.

# Незнайка съел Котлета по-киевски в количестве: 1

# Внимание! На ракете закончился продукт: Баварская колбаска к пиву

# Пончик объелся. Он не доел Баварская колбаска к пиву.

# Незнайка не голоден.

# Незнайка сыт на 2300/2500.

# Пончик не голоден.

# Пончик сыт на 2500/2500.

# Незнайка хочет спать? false

# Пончик хочет спать? true

# Пончик заснул c едой во рту.

# Незнайка перешёл из пищеблок в астрономическая кабина.

# Звезда 'Солнце' сияла в иллюминаторе.

# Спутник 'Луна' светилcя в одном из иллюминаторов.

# Незнайка оценил расстояне до космического объекта 'Солнце'. Кажется, что расстояние осталось прежним.

# Незнайка посмотрел на Луну. Кажется, что Луна стала в два раза больше.

# Текущая скорость ракеты - 12.0 километров в секунду.

# Расстояние от Земли до Луны очень большое - около четырехсот тысяч километров. При таком огромном расстоянии скорость двенадцать километров в секунду не так велика, чтоб ее можно было заметить на глаз, да еще находясь в ракете.

# При такой скорости путь от Земли до Луны займет около 9 часов.

# Process finished with exit code 0

# Вывод

При выполнении данной лабораторной работы я продолжил изучение основных принципов ООП в Java, научился работать с интерфейсами и абстрактными классами. Также я научился обрабатывать исключения, создавать собственные checked и unchecked исключения.

Изображение выглядит как внутренний, млекопитающее, загроможденный

Автоматически созданное описание