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 №86099,2

**Группа**: R3138

**Выполнил**: Ракин Илья Николаевич

**Преподаватель: Харитонова Анастасия Евгеньевна**

Санкт-Петербург

2022

# Текст задания

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Оглавление

[Текст задания 2](#_Toc92888419)

[UML диаграмма классов 4](#_Toc92888420)

[Исходный код программы 5](#_Toc92888421)

[Main 5](#_Toc92888422)

[Death 6](#_Toc92888423)

[Debily 6](#_Toc92888424)

[Move 9](#_Toc92888425)

[Neznaika 9](#_Toc92888426)

[Pan 9](#_Toc92888427)

[PersonAmount 10](#_Toc92888428)

[PersonAmountException 11](#_Toc92888429)

[Drink 11](#_Toc92888430)

[Ponchik 11](#_Toc92888431)

[Porridge 12](#_Toc92888432)

[Room 12](#_Toc92888433)

[Shpuntik 12](#_Toc92888434)

[SkinColor 12](#_Toc92888435)

[Staty 13](#_Toc92888436)

[Talker 13](#_Toc92888437)

[TalkerException 13](#_Toc92888438)

[Temperature 14](#_Toc92888439)

[Thing 14](#_Toc92888440)

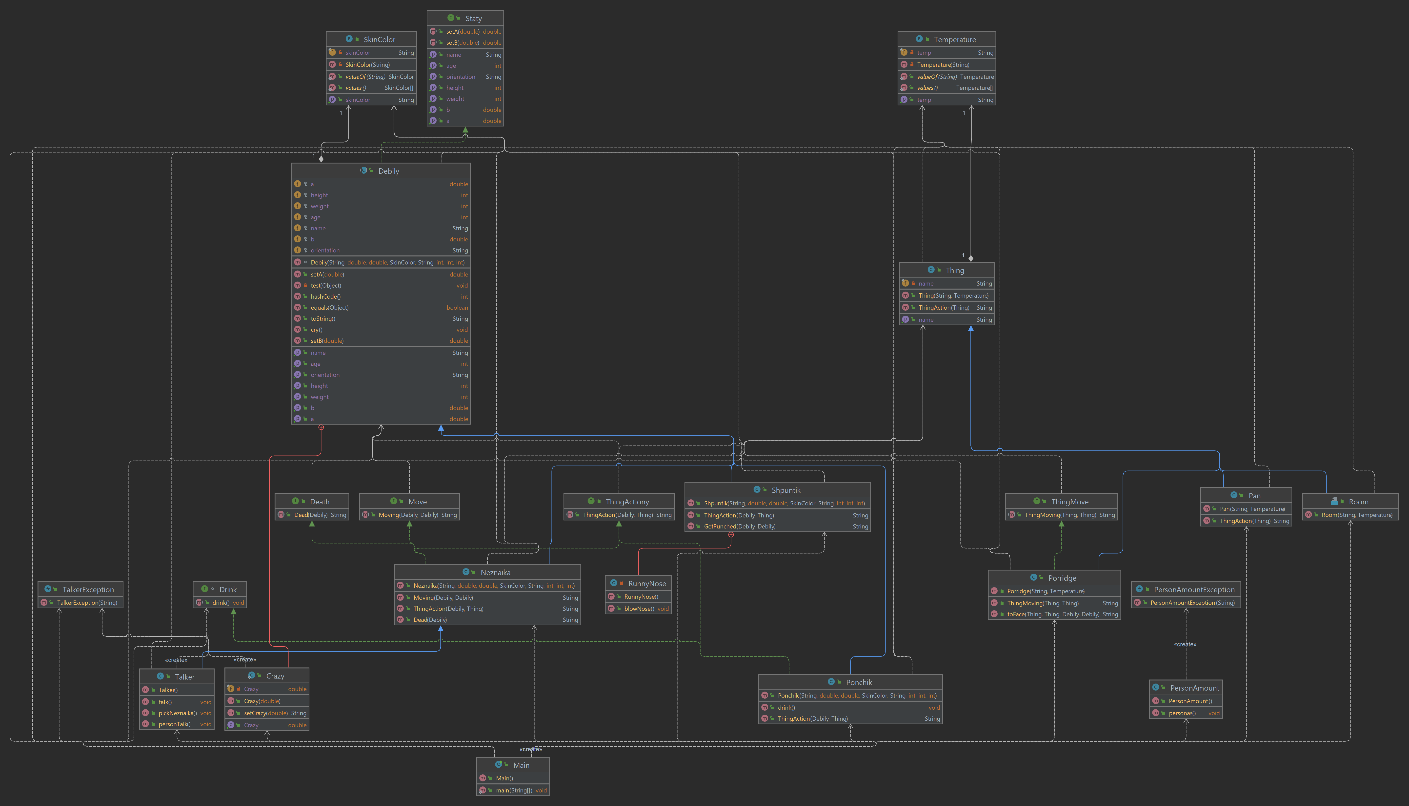
[ThingActiony 14](#_Toc92888441)

[ThingMove 14](#_Toc92888442)

[Выходные данные программы 15](#_Toc92888443)

[Заключение 16](#_Toc92888444)

# UML диаграмма классов



# Исходный код программы

## Main

public class Main {  
 public static void main(String[] args) {  
 Pan pan = new Pan("кастрюля с кашей", Temperature.*HOT*);  
 Room room = new Room("комната", Temperature.*COLD*);  
 Porridge porridge = new Porridge("Манная каша",Temperature.*HOT*);  
 Neznaika neznaika = new Neznaika("Незнайка",15,30,SkinColor.*ALBO*,"Натурал",48,94,190);  
 Shpuntik shpuntik = new Shpuntik("Шпунтик", 138947,180,SkinColor.*BLACK*, "Гомосексуал ", 13, 45, 152);  
 Ponchik ponchik = new Ponchik("Пончик", 9250, 9250, SkinColor.*CHINESE*, "Гетеросексуал", 75, 66, 180);  
 Debily.Crazy crazy = new Debily.Crazy(0);  
 PersonAmount personAmount = new PersonAmount();  
 personAmount.personas();  
 Talker talker = new Talker();  
 talker.personTalk();  
 talker.pickNeznaika();  
 Drink deadponchik = new Drink() {  
 @Override  
 public void drink() {  
 System.*out*.println("Пончик уже не может выпить молока");  
 }  
 };  
 try {  
 talker.talk();  
 shpuntik.cry();  
// shpuntik.blowNose();  
 System.*out*.println(neznaika.Moving(neznaika, shpuntik));  
 System.*out*.println(shpuntik.ThingAction(shpuntik,room));  
 System.*out*.println(pan.ThingAction(pan));  
 System.*out*.println(porridge.ThingMoving(pan, porridge));  
 System.*out*.println(neznaika.ThingAction(neznaika, pan));  
 System.*out*.println(shpuntik.GetPunched(neznaika, shpuntik));  
 System.*out*.println(porridge.toFace(pan,porridge,neznaika,ponchik));  
 System.*out*.println(ponchik.ThingAction(ponchik,porridge));  
 deadponchik.drink();  
 } catch (TalkerException n) {  
 System.*out*.println(n.getMessage());  
 n.printStackTrace();  
 }  
 System.*out*.println("The End.");  
 }  
}

## Death

public interface Death {  
 public String Dead(Debily m);  
}

## Debily

import java.util.Objects;  
  
public abstract class Debily implements Staty {  
 protected double a;  
 protected double b;  
 protected String name;  
 protected SkinColor skinColor;  
 protected String orientation;  
 protected int weight;  
 protected int height;  
 protected int age;  
  
 Debily(String name, double a, double b, SkinColor skinColor, String orientation, int age, int weight, int height) {  
 this.name = name;  
 this.a = a;  
 this.b = b;  
 this.skinColor = skinColor;  
 this.orientation = orientation;  
 this.age = age;  
 this.weight = weight;  
 this.height = height;  
 }  
 @Override  
 public double getA(){  
 return this.a;  
 }  
  
 @Override  
 public double getB(){  
 return this.b;  
 }  
  
 @Override  
 public double setA(double a){  
 this.a = a;  
 return a;  
 }  
  
 @Override  
 public double setB(double b){  
 this.b = b;  
 return b;  
 }  
 @Override  
 public String getName() {  
 return this.name;  
 }  
// @Override  
// public SkinColor getSkinColor() {  
// return this.skinColor;  
// }  
  
 @Override  
 public String getOrientation() {  
 return this.orientation;  
 }  
 @Override  
 public int getAge() {  
 return this.age;  
 }  
  
 @Override  
 public int getWeight() {  
 return this.weight;  
 }  
  
 @Override  
 public int getHeight() {  
 return this.height;  
 }  
 @Override  
 public String toString(){  
 return "Debil{" + "name= '" + name + '\n' + ", skin color= '" + skinColor + '\n' + ", orientation= '"  
 + orientation + '\n' + ", age= " + age + '\n' + ", weight=" + weight + '\n' + ", height=" + height + '}';  
 }  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, a, b);  
 }  
 @Override  
 public boolean equals(Object m) {  
 if (this == m) return true;  
 if (m == null || getClass() != m.getClass()) return false;  
 Debily Debily = (Debily) m;  
 return Objects.*equals*(name, Debily.name)  
 && Objects.*equals*(a, Debily.a)  
 && Objects.*equals*(b, Debily.b);  
 }  
 public void cry(){  
 System.*out*.println(name + " "+ "заплакал");  
 final int x = 1;  
 class SomeClass {  
 public void someMethod(){  
 System.*out*.println(name);  
 System.*out*.println(x);  
 }  
 }  
 SomeClass someObject = new SomeClass();  
 test(someObject);  
 }  
 private void test(Object object){  
  
 }  
 // вложенный статический класс  
 public static class Crazy {  
 String name;  
 public Crazy(double Crazy) {  
 this.Crazy = Crazy;  
 }  
 private final double Crazy;  
 public double getCrazy(){  
 return Crazy;  
 }  
 public String setCrazy(double Crazy) throws TalkerException{  
 if(Crazy == 0) {  
 return (name + " " + "не сошел с ума, все в порядке");  
 } else {  
 throw new TalkerException(name + " " + "сошёл с ума...");  
 }  
 }  
 }  
// public void Blownose(double amount){  
// try {  
// if(amount == 0){  
// System.out.println("У" + " " + name + " " + "текут сопли");  
// }  
// } catch (TalkerException p){  
// Talker.talk()  
// }  
// }  
}  
// @Override  
// public int hashCode(){  
// return Objects.hash(name, )  
// }

*public void* addMember(Conversationable member) {  
 interlocutors.add(member);  
 member.setConversation(*this*);  
 }  
  
 *public void* removeMember(Conversationable member) {  
 interlocutors.remove(member);  
 member.setConversation(*null*);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* interlocutors.toString();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Conversation that = (Conversation) o;  
 *return* interlocutors.equals(that.interlocutors);  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return* Objects.hash(interlocutors);  
 }  
}

## Move

public interface Move {  
 String Moving(Debily k, Debily x);  
}

## Neznaika

public class Neznaika extends Debily implements ThingActiony, Move, Death {  
 public Neznaika(String name, double a, double b, SkinColor skinColor, String orientation, int age, int weight, int height) {  
 super(name, a, b, skinColor, orientation, age, weight, height);  
 }  
  
 @Override  
 public String Moving(Debily k, Debily x) {  
 return k.getOrientation() + " " + k.skinColor.getSkinColor() + " " + k.getName() + " взмахнул кулаком и нанес удар " + x.getName() + "у";  
 }  
  
 @Override  
 public String ThingAction(Debily k, Thing x) {  
 return k.getName() + " " + "от своего же удара отлетел и ударился головой о " + x.getName();  
 }  
  
 @Override  
 public String Dead(Debily m) {  
 return "У " + m.getName() + " отрывается тромб и он умирает из-за слишком резкого замаха руки";  
 }  
}

}

## Pan

public class Pan extends Thing{  
 public Pan(String name, Temperature temp) {  
 super(name, temp);  
 }  
 @Override  
 public String ThingAction(Thing p){  
 return "На плите стояла " + p.getName();  
 }  
}

## PersonAmount

*package* creatures;  
public class PersonAmount {  
 int amount = 3;  
 public PersonAmount(){ // int amount  
// this.amount=amount;  
 }  
 public void personas() throws PersonAmountException {  
 if(amount == 3) {  
 System.*out*.println("В ролях: Незнайка, Шпунтик, Пончик");  
 } else {  
 throw new PersonAmountException("В этой истории не учавствует такое количество героев :(");  
 }  
 }  
}

## PersonAmountException

public class PersonAmountException extends RuntimeException{  
 public PersonAmountException(String message) {  
 super(message);  
 }  
}

## Drink

interface Drink {  
 public void drink();  
}  
public class Ponchik extends Debily implements ThingActiony, Drink {  
 public Ponchik(String name, double a, double b, SkinColor skinColor, String orientation, int age, int weight, int height) {  
 super(name, a, b, skinColor, orientation, age, weight, height);  
 }  
  
 @Override  
 public String ThingAction(Debily k, Thing s) {  
 return s.temp.getTemp() + " " + s.getName() + " начинает прожигать лицо бедного " + k.getName() + "a";  
 }  
  
 @Override  
 public void drink() {  
 System.*out*.println("Пончик выпил стакан молока");  
 }  
}

## Ponchik

interface Drink {  
 public void drink();  
}  
public class Ponchik extends Debily implements ThingActiony, Drink {  
 public Ponchik(String name, double a, double b, SkinColor skinColor, String orientation, int age, int weight, int height) {  
 super(name, a, b, skinColor, orientation, age, weight, height);  
 }  
  
 @Override  
 public String ThingAction(Debily k, Thing s) {  
 return s.temp.getTemp() + " " + s.getName() + " начинает прожигать лицо бедного " + k.getName() + "a";  
 }  
  
 @Override  
 public void drink() {  
 System.*out*.println("Пончик выпил стакан молока");  
 }  
}

## Porridge

public class Porridge extends Thing implements ThingMove {  
 public Porridge(String name, Temperature temp){  
 super(name,temp);  
 }  
 @Override  
 public String ThingMoving(Thing x, Thing s){  
 return "В " + x.getName() + " находится " + s.temp.getTemp() + " " + s.getName();  
 }  
 public String toFace(Thing k, Thing s, Debily u, Debily a){  
 return "От столкновения с " + u.getName() + ", " + k.getName() + " падает и " + s.getName() +  
 " выплескивается на лицо ничего не подозревающего " + a.skinColor.getSkinColor() + " "  
 + a.getOrientation() + " " + a.getAge() + "-ти лет " + a.getName() + "a";  
 }  
}

## Room

public class Room extends Thing{  
 public Room(String name, Temperature temp){  
 super(name, temp);  
 }  
}

## Shpuntik

public class Shpuntik extends Debily {  
 public Shpuntik(String name, double a, double b, SkinColor skinColor, String orientation, int age, int weight, int height) {  
 super(name, a,b, skinColor, orientation, age, weight, height);  
 }  
 public String ThingAction(Debily k, Thing x){  
 return "Получив по щам, " + k.skinColor.getSkinColor() + k.getOrientation() + k.getName() + " завертелся волчком и полетел через всю " + x.getName();  
 }  
 public String GetPunched(Debily x, Debily n){  
 return "Оклемавшись от удара нанесенного " + x.getName() + ", " + n.getName() + " прокричал: Что ты делаешь? Мне всего " + n.getAge() + " лет";  
 }  
 // вложенный нестатический класс  
 private class RunnyNose {  
 public void blowNose(){  
 System.*out*.println("У" + " " + name + " " + "текут сопли");  
 }  
 }  
}  
// public void cry(){  
// System.out.println(name + " "+ "заплакал");  
// }

## SkinColor

public enum SkinColor {  
 *BLACK*("чернокожий "),  
 *WHITE*("белокожий "),  
 *ALBO*("альбинос"),  
 *CHINESE*("азиат");  
  
 private final String skinColor;  
 SkinColor(String skinColor){  
 this.skinColor=skinColor;  
 }  
 public String getSkinColor(){  
 return skinColor;  
 }  
}

## Staty

public interface Staty{  
 double getA();  
 double getB();  
 double setA(double a);  
 double setB(double b);  
 String getName();  
 String getOrientation();  
 int getAge();  
 int getWeight();  
 int getHeight();  
}

## Talker

public class Talker extends Neznaika {  
// String name;  
 boolean isPersonNeznaika;  
 boolean isPersonTalk;  
 public Talker(){  
 super("Незнайка",15,30,SkinColor.*ALBO*,"Натурал",48,94,190);  
// this.name=name;  
 }  
 public void personTalk(){  
 System.*out*.println("Кое-кто очень хочет кое-что Вам рассказать. Cлушаем внимательно!");  
 this.isPersonTalk = true;  
 }  
 public void pickNeznaika(){  
 System.*out*.println("Наш рассказчик сегодня - Незнайка");  
 this.isPersonNeznaika = true;  
 }  
 public void talk() throws TalkerException {  
 if(isPersonNeznaika && isPersonTalk) {  
 System.*out*.println("Начало рассказа..." + '\n' + "А" + " " + name + " " + "сказал:" + " ");  
 } else {  
 throw new TalkerException("Рассказчик" + " " + name + " " + "передумал говорить" + "." + " "  
 + "Истории не будет." + " " + "Повторите попытку позднее.");  
 }  
 }  
}

## TalkerException

public class TalkerException extends Exception {  
 public TalkerException(String msg) {  
 super(msg);  
 }  
}

## Temperature

public enum Temperature {  
 *HOT*("горячая"), *COLD*("холодная");  
  
 private final String temp;  
 Temperature(String temp){  
 this.temp=temp;  
 }  
 public String getTemp(){  
 return temp;  
 }  
}

## Thing

public class Thing {  
 private final String name;  
 protected Temperature temp;  
  
 public Thing(String name, Temperature temp){  
 this.name = name;  
 this.temp = temp;  
 }  
 public String getName(){  
 return name;  
 }  
// public Temperature getTemp(){  
// return temp;  
// }  
 public String ThingAction(Thing o){  
 return "";  
 }  
}

## ThingActiony

public interface ThingActiony {  
 String ThingAction(Debily k, Thing x);  
}

## ThingMove

public interface ThingMove {  
 String ThingMoving(Thing x, Thing s);  
}

# Выходные данные программы

Изображение выглядит как текст

Автоматически созданное описание

# Заключение

Для выполнения этой лабораторной работы я изучил различные типы классов: анонимные, вложенные (static&non-static), локальные, а также узнал о типах исключений в Java, о том, как их обрабатывать и о том, как создавать собственные.