**Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.**

Факультет Программной Инженерии и Компьютерной Техники

A picture containing logo

Description automatically generated

**ПРОГРАММИРОВАНИЕ**

Лабораторная работа №4.

Вариант 33130

Выполнил:

ФИО: Окечукву Александр Чуквуэмека

ГРУППА: P3133

ПРЕПОДАВАТЕЛЬ: Шешуков Д. М.

Санкт Петербург

2021 г.

Table of Contents

[Задание 3](#_Toc93120763)

[Диаграмма классов 4](#_Toc93120764)

[Код 5](#_Toc93120765)

[Object: 6](#_Toc93120766)

[Интерфейсы: 7](#_Toc93120767)

[Вывод программы 12](#_Toc93120768)

[Вывод 12](#_Toc93120769)

# Задание

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

Тут за дверью послышались шаги. Незнайка быстро юркнул в постель и накрылся одеялом. В комнату вошли Синеглазка и другая малышка, в белом халате и белой шапочке, с небольшим коричневым чемоданчиком в руках. У нее были пухлые румяные щечки. Серые глазки строго смотрели из-за круглых роговых очков. Незнайка понял, что это и есть Медуница, о которой ему говорила Синеглазка.

**Программа должна удовлетворять следующим требованиям:**

1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.

2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**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

Graphical user interface

Description automatically generated

# Код

Main:

public class Main {  
 public static void main(String [] arg){  
 Girls member = new Girls();  
 Girls.BlueEye payper = member.new BlueEye();  
 Cheeks potter = new Cheeks();  
 GrayEyes santan = new GrayEyes();  
 Girls.Medunica digdat = member.new Medunica();  
 Blanket headie = new Blanket();  
 Steps teflon = new Steps(){  
 private String name = "Steps";  
 public void underDoor(){  
 System.*out*.print("There was a sound outside the door " + name + ".");  
 }  
 };  
 Robe robe = new Robe();  
 WhiteCap cap = new WhiteCap();  
 Suitcase realart = new Suitcase();  
 Dunno dunno = new Dunno();  
 try {dunno.gender();  
 }  
 catch (MaleException e){  
 System.*out*.println(e.getMessage());  
 }  
 System.*out*.println();  
 teflon.underDoor();  
 dunno.goToBed(headie.getName());  
 dunno.PutOnBlanket("covered with a blanket.");  
 payper.enter(robe, cap, realart);  
 potter.were();  
 santan.watch();  
 dunno.understood(digdat.getName());  
 payper.talk();  
 }  
}

# Object:

import java.util.Objects;  
  
public abstract class object implements showName, showColor{  
 private String name;  
 private color value;  
  
 public object(String name){  
 this.name = name;  
 }  
 public object(String name, color value){  
 this.name = name;  
 this.value = value;  
 }  
  
 protected void Join(String name){  
 System.*out*.println(name + " in history");  
 }  
  
 public String getName(){  
 return this.name;  
 }  
  
 public color getColor(){  
 return this.value;  
 }  
  
 public void setColor(color value){  
 this.value = value;  
 }  
  
 public String toString(){  
 return getName();  
 }  
  
 public int HashCode(){  
 return this.name.hashCode();  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 object object = (object) o;  
 return Objects.*equals*(name, object.name) && value == object.value;  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(name, value);  
 }  
}

# Интерфейсы:

showColor

public interface showColor {  
 color getColor();  
}

showName

public interface showName {  
 String getName();  
}

Steps

public interface Steps {  
 void underDoor();  
}

# Объекты:

Blanket.java

public class Blanket extends object{  
 public Blanket(){  
 super("Blanket");  
 Join(getName());  
 }  
}

BlueEye.java

public class BlueEye extends object{  
 public BlueEye(){  
 super("Blue Eyes");  
 Join(getName());  
 }  
 public void enter(object var1,object var2,object var3){  
 System.*out*.println("The room was entered " + getName() + " and another little girl, in " + var1.getColor() + "ом " + var1.getName()  
 + "е и " + var2.getColor() + "ой " + var2.getName() + "е," +  
 " with a little " + var3.getColor() + "ым " + var3.getName() + "om in the hands.");  
 }  
 public void talk(){  
 System.*out*.print("Which I told him about "+ getName() + ".");  
 }  
}

public class Cheeks extends object{  
 public Cheeks(){  
 super("Cheeks", color.*rosy*);  
 Join(getName());  
 }  
 public void were(){  
 System.*out*.print("She had plump " + getColor() + " "+ getName() + ".");  
 }  
}

Dunno.java

public class Dunno extends object{  
 private boolean male = true;  
 public Dunno(){  
 super("Dunno");  
 Join(getName());  
 }  
 public void goToBed(String var){  
 System.*out*.print(" "+ getName() + " he quickly slipped into bed");  
 }  
 public void understood(String var){  
 System.*out*.print(" " + getName() + " I realized that this is " + var + ", ");  
 }  
 public void gender() throws MaleException{  
 if (male == true){  
 System.*out*.println(getName() + " is a man");  
 }  
 else {  
 throw new MaleException(getName() + " is not a man");  
 }  
 }  
 public void PutOnBlanket(String b){  
 class blanket{  
 private String action = "covered with a blanket.";  
 public blanket(){  
 this.action = b;  
 }  
 }  
 System.*out*.println(" и " + b);  
 }  
  
  
}

 Girls.java

public class Girls {  
 public class BlueEye extends object{  
 private boolean female = true;  
  
 public BlueEye(){  
 super("Blue Eyes");  
 Join(getName());  
 }  
 public void enter(object var1,object var2,object var3){  
 System.*out*.println("The room was entered " + getName() + " and another baby, in " + var1.getColor() + "ом " + var1.getName()  
 + "е и " + var2.getColor() + "ой " + var2.getName() + "е," +  
 " with a little " + var3.getColor() + "ым " + var3.getName() + "om in the hands.");  
 }  
 public void talk(){  
 System.*out*.print("Which I told him about "+ getName() + ".");  
 }  
 public void gender2() throws Throwable{  
 if (female == true){  
 System.*out*.println(getName() + " is a woman");  
 }  
 else {  
 throw new RuntimeException();  
 }  
 }  
  
 }  
 public class Medunica extends object{  
 public Medunica(){  
 super("Medunica");  
 Join(getName());  
 }  
 }  
  
  
}

GrayEyes.java

public class GrayEyes extends object{  
 public GrayEyes(){  
 super("GrayEyes", color.*gray*);  
 Join(getName());  
 }  
 public void watch(){  
 System.*out*.print(" " + getColor() + " " + getName() + " Stared sternly from behind round horn-rimmed glasses.");  
 }  
}

Medumica.java

public class Medunica extends object{  
 public Medunica(){  
 super("Meducina");  
 Join(getName());  
 }  
}

Robe.java

public class Robe extends object{  
 public Robe(){  
 super("Robe", color.*white*);  
 Join(getName());  
 }  
}

Suitcase.java

public class Suitcase extends object{  
 private color brown;  
 public Suitcase(){  
 super("Suitcase", color.*brown*);  
 Join(getName());  
 }  
}

WhiteCap.java

public class WhiteCap extends object{  
 public WhiteCap(){  
 super("Cap", color.*white*);  
 Join(getName());  
 }  
}

Enum:

public enum color {  
 *white*,  
 *brown*,  
 *gray*,  
 *rosy*,  
}

Локальный, вложенный статический и нестатический, анонимный классы:

public void PutOnBlanket(String b){  
 class blanket{  
 private String action = "covered with a blanket.";  
 public blanket(){  
 this.action = b;  
 }  
 }  
 System.*out*.println(" и " + b);  
 }  
  
  
}

public class Girls {  
 public class BlueEye extends object{  
 private boolean female = true;  
  
 public BlueEye(){  
 super("Blue Eyes");  
 Join(getName());  
 }  
 public void enter(object var1,object var2,object var3){  
 System.*out*.println("The room was entered " + getName() + " and another baby, in " + var1.getColor() + "ом " + var1.getName()  
 + "е и " + var2.getColor() + "ой " + var2.getName() + "е," +  
 " with a little " + var3.getColor() + "ым " + var3.getName() + "om in the hands.");  
 }  
 public void talk(){  
 System.*out*.print("Which I told him about "+ getName() + ".");  
 }  
 public void gender2() throws Throwable{  
 if (female == true){  
 System.*out*.println(getName() + " is a woman");  
 }  
 else {  
 throw new RuntimeException();  
 }  
 }  
  
 }  
 public class Medunica extends object{  
 public Medunica(){  
 super("Medunica");  
 Join(getName());  
 }  
 }  
  
  
}

Steps teflon = new Steps(){  
 private String name = "Steps";  
 public void underDoor(){  
 System.*out*.print("There was a sound outside the door " + name + ".");  
 }  
};

Exceptions:

public class MaleException extends Exception{  
 public MaleException(String a){  
 super(a);  
 }  
}

public void gender() throws MaleException{  
 if (male == true){  
 System.*out*.println(getName() + " is a man");  
 }  
 else {  
 throw new MaleException(getName() + " is not a man");  
 }  
}

try {dunno.gender();  
}  
catch (MaleException e){  
 System.*out*.println(e.getMessage());  
}  
System.*out*.println();

public void gender2() throws Throwable{  
 if (female == true){  
 System.*out*.println(getName() + " is a woman");  
 }  
 else {  
 throw new RuntimeException();  
 }  
}

# Вывод программы

Text

Description automatically generated

# Вывод

Работая в лаборатории, я изучал различные типы n классов, а также изучал исключения и их использование.