Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

факультет Программной инженерии и компьютерных технологий

Лабораторная работа №3

по дисциплине

«Программирование»

Вариант - 52160

Выполнил: Студент группы P3113 Хатиб Ватан

Преподаватель: Райла Мартин.

Санкт-Петербург

2021 г.

Текст

A screenshot of a computer

Description automatically generated with medium confidence

UML

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram, text

Description automatically generated

Diagram

Description automatically generated

Main Class:

public class Main {  
  
 public static void main(String[] args) {  
 Place table = new Place("стол");  
 Telephone telephone = new Telephone();  
 Box normal = new Box("ящик","чистыми бланками","регистрации арестованных");  
 Box flatBox = new FlatBox("коробочка","черной типографской краской","изготовления отпечатков пальцев");  
 Face face =new Face("скуластые","низкие","широколицые");  
 Hair hair =new Hair("темные","жесткие","подстриженные ежиком");  
 HumanWithHair megel = new HumanWithHair("Мигля",face,hair);  
 Helmet megelHelmet = new IronHelmet(megel,50);  
 HumanWithHair fegel = new HumanWithHair("Фигля",face,hair);  
 Place room = new Place("Комната");  
 Helmet fegelHelmet = new Helmet(fegel,100);  
 fegel.walk(room);  
 fegel.addSkill(new informationSkill("снял","с головы "+fegelHelmet.getName()));  
 fegel.addSkill(new placeSkill("поставил "+ fegelHelmet.getName(),table));  
 megelHelmet.isBrighter(fegelHelmet);  
 fegel.getSimiler(megel);  
  
 }  
}

AbsHuman:

package Story;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public abstract class AbsHuman {  
 String name;  
 Face face;  
 List<Iskill> skillsList =new ArrayList<>();  
  
 public AbsHuman(String name, Face face) {  
 this.name = name;  
 this.face = face;  
 }  
  
 public void walk(Place place) {  
 System.*out*.println(name +" входил в "+place.toString());  
 }  
 public void addSkill(Iskill skill){  
 skillsList.add(skill);  
 System.*out*.println(name +" "+skill.descripe());  
 }  
}

Box:

public class Box {  
 String name;  
 String content;  
 String functionality;  
  
 public Box(String name, String content, String functionality) {  
 this.name = name;  
 this.content = content;  
 this.functionality=functionality;  
 System.*out*.println("успешно создан бокс именно "+name +" с "+content +" для "+functionality);  
 }  
}

Face:

public class Face {  
 String cheekbones;  
 String type;  
 String forehead;  
  
 public Face(String cheekbones, String type, String forehead) {  
 this.cheekbones = cheekbones;  
 this.type = type;  
 this.forehead = forehead;  
 }  
  
 @Override  
 public String toString() {  
 return this.type +" , "+this.forehead+" , "+this.cheekbones;  
 }  
}

FlatBox:

public class FlatBox extends Box implements IFlat {  
  
 public FlatBox(String name, String content, String functionality) {  
 super(name, content, functionality);  
 System.*out*.println("успешно создан "+*shape*+" бокс именно " +name +" с " +content +" для "+functionality);  
 }  
}

Hair

public class Hair {  
 String color;  
 String type;  
 String cut;  
  
 public Hair(String color, String type, String cut) {  
 this.color = color;  
 this.type = type;  
 this.cut = cut;  
 }  
  
 @Override  
 public String toString() {  
 return this.type +" , "+this.cut +" , "+ this.color;  
 }  
}

Helmet:

public class Helmet {  
 String name="каска";  
 AbsHuman owner;  
 int brightens;  
  
 public Helmet(AbsHuman owner, int brightens) {  
 this.owner = owner;  
 this.brightens = brightens;  
 System.*out*.println("успешно создан "+name +" для " + owner.name);  
 }  
  
 public void isBrighter(Helmet helmet){  
 if (this.brightens > helmet.brightens){  
 System.*out*.println(this.owner.name +" " +this.name +" болший яркоо чем "+helmet.owner.name +" "+helmet.name);  
 }else if (this.brightens < helmet.brightens){  
 System.*out*.println(this.owner.name +" " +this.name +" меньшии яркоо чем "+helmet.owner.name +" "+helmet.name);  
 }else {  
 System.*out*.println( "такжи яркоо");  
 }  
 }  
  
 @Override  
 public int hashCode() {  
 return super.hashCode() +brightens;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (!obj.getClass().getName().equals("Helmet")){  
 return false;  
 }  
 Helmet other = (Helmet)obj;  
 return other.owner == this.owner && other.brightens == this.brightens && other.name.equals(this.name);  
 }  
  
 public String getName() {  
 return name;  
 }  
}

HumanWithHair:

public class HumanWithHair extends AbsHuman {  
 Hair hair;  
  
 public HumanWithHair(String name,Face face,Hair hair) {  
 super(name,face);  
 this.hair=hair;  
 System.*out*.println("успешно создан человик с волосом иммено "+name);  
 }  
 public void getSimiler(HumanWithHair human){  
 System.*out*.println("сходство между "+ this.name +" и "+human.name);  
 if (human.face.cheekbones.equals(this.face.cheekbones)){  
 System.*out*.println("оба "+this.face.cheekbones);  
 }  
 if (human.face.forehead.equals(this.face.forehead)){  
 System.*out*.println("оба "+this.face.forehead);  
 }  
 if (human.face.type.equals(this.face.type)){  
 System.*out*.println("оба "+this.face.type);  
 }  
 if(human.hair.color.equals(this.hair.color)){  
 System.*out*.println("оба волос "+this.hair.color);  
 }  
 if(human.hair.type.equals(this.hair.type)){  
 System.*out*.println("оба волос "+this.hair.type);  
 }  
 if(human.hair.cut.equals(this.hair.cut)){  
 System.*out*.println("оба волос "+this.hair.cut);  
 }  
 if (human.name.equals(this.name)){  
 System.*out*.println("оба именно "+this.name);  
 }  
  
  
  
  
 }  
  
}

Iflat:

public interface IFlat {  
 String *shape* ="плоская";  
}

informationskill:

public class informationSkill implements Iskill {  
 String name;  
 String information;  
  
 public informationSkill(String name , String information) {  
 this.name = name;  
 this.information=information;  
 }  
  
 @Override  
 public String descripe() {  
 return name + " "+information;  
  
 }  
}

Iron

public interface Iron {  
 String *material* ="медная";  
}

IronHelmet:

public class IronHelmet extends Helmet implements Iron{  
 public IronHelmet(AbsHuman owner, int brightens) {  
 super(owner, brightens);  
 System.*out*.println("успешно создан "+*material* +" "+name +" с яркее="+brightens +" для " +owner.name);  
 }  
}

Iskill:

public interface Iskill {  
 String *name*="";  
 public String descripe();  
  
}

Place:

public class Place {  
 private String placeName;  
 private Places type;  
  
  
 public Place(String name) {  
 this.placeName=name;  
 switch (placeName) {  
 case "стол":  
 type=Places.*Table*;  
 break;  
 case "Комната":  
 type = Places.*Room*;  
 break;  
 }  
 if (type != null){  
 System.*out*.println(name +" успешно создан ! как " +type.toString());  
 }else {  
 System.*out*.println(name +" успешно создан !");  
 }  
  
 }  
  
 public String getPlaceName() {  
 return placeName;  
 }  
  
 @Override  
 public String toString() {  
 return "Место " + this.getPlaceName();  
 }  
  
}

Places:

public enum Places {  
 *Table*,  
 *Room*,  
}

PlaceSkill

public class placeSkill implements Iskill {  
 String name;  
 Place place;  
 public placeSkill(String name , Place place) {  
 this.name = name;  
 this.place=place;  
 }  
  
 @Override  
 public String descripe() {  
 return name +" в " +place.getPlaceName();  
 }  
}

TelePhone:

public class Telephone {  
 String name="телефонный аппарат";  
  
 public Telephone() {  
 System.*out*.println("успешно создан "+name);  
 }  
}

Во время выполнения лабораторной работы я научился работать с абстрактными классами, интерфейсами и нумераторами. Лучше освоил работу с полями, методами и классами и построение UML диаграмм.