**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»**

**Факультет Программной Инженерии и Компьютерной Техники**

**Отчет по лабораторной работе №4:**

**Вариант CUSTOM**

Студент:

Иванов Илья Андреевич

Р3117

Преподаватель:

Коновалов Арсений Антонович

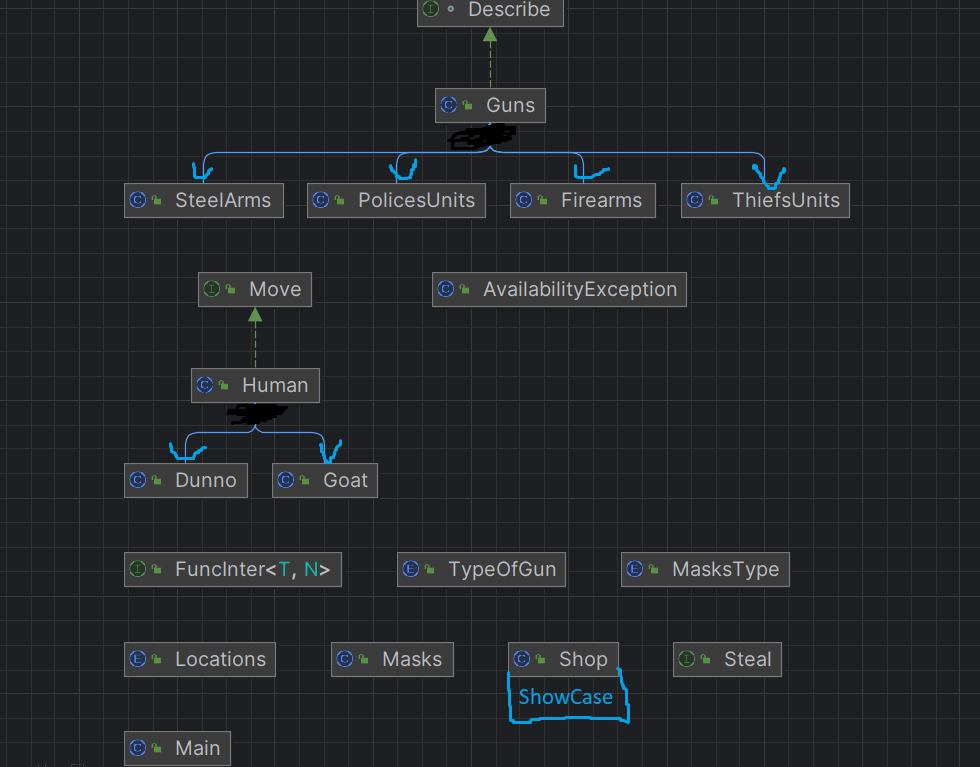
Санкт-Петербург, 2023

# Текст задания

Тут же находилась витрина, в которой были выставлены различные маски: и такие, которые закрывают лишь верхнюю часть лица, с прорезами для глаз, и такие, которые надеваются целиком на голову, в виде островерхого капюшона. Кроме масок, здесь были также грим, парики, накладные бороды и усы – все, что помогает изменить внешность.

В правом углу стояло чучело полицейского в полном обмундировании, с блестящей медной каской на голове и дубинкой в руке. В левом углу было чучело грабителя, подкрадывающегося к несгораемой кассе, с огромным пистолетом в одной руке и потайным электрическим фонарем в другой. Его шея была повязана пестрым клетчатым платком, на голове была клетчатая кепка с широким козырьком, такие же клетчатые брюки плотно облегали его ноги, лицо закрывала черная маска. Оба чучела были сделаны с таким мастерством, что их можно было принять за настоящих живых коротышек.

# Диаграмма классов реализованной объектной модели



# Коды программ

Main:

public class Main {  
 public static void main(String[] args) {  
 Dunno person1 = new Dunno("тупенький"**,** (int) (Math.*random*() \* **10000**))**;** Goat person2 = new Goat("тупенький2"**,** (int) (Math.*random*() \* **10000**))**;** Guns pistol = new Firearms("огромный пистолет")**;** Guns torch = new SteelArms("потайной электрпический фонарь")**;** pistol.setType(TypeOfGun.*PISTOL*)**;** pistol.setPrice(**132**)**;** torch.setPrice(**1**)**;** torch.setType(TypeOfGun.*TORCH*)**;** person1.getMoney()**;** person1.move(Locations.*SHOP*)**;** person2.move(Locations.*SHOP*)**;** person1.buy(pistol)**;** try {  
 person1.use(pistol)**;** } catch (AvailabilityException ex) {  
 System.*out*.println(ex.getMessage())**;** }  
 person2.buy(pistol)**;** person1.buy(torch)**;** person1.getItems()**;** person2.getItems()**;** person1.move(Locations.*STREET*)**;** person2.move(Locations.*STREET*)**;** try {  
 person2.use(torch)**;** } catch (AvailabilityException ex) {  
 System.*out*.println(ex.getMessage())**;** }  
 //person1.isTrue();  
 FuncInter<Goat**,** Dunno> converter = x -> new Dunno(x.getName()**,** x.getValue()**,** x.getLastItem())**;** Dunno personConv = converter.convert(person2)**;** System.*out*.println(personConv)**;** System.*out*.println(person2)**;** Shop shop = new Shop("magazik")**;** Shop.ShowCase vitrina = shop.new ShowCase("витрина")**;** Shop.Dummy choochelo2 = new Shop.Dummy("полицейский")**;** Masks helmet = new Masks("медная каска")**;** helmet.setType(MasksType.*HELMET*)**;** choochelo2.addMask(helmet)**;** Guns baton = new PolicesUnits("дубинка")**;** baton.setType(TypeOfGun.*BATON*)**;** choochelo2.setLeftItem(baton)**;** choochelo2.move(Locations.*RIGHTCORNER*)**;** Shop.Dummy choochelo1 = new Shop.Dummy("грабитель")**;** choochelo1.stealing(Locations.*CASHBOX*)**;** choochelo1.setLeftItem(pistol)**;** choochelo1.setRightItem(torch)**;** Masks neck = new Masks("пестрый клетчатый платок")**;** neck.setType(MasksType.*SHAWL*)**;** choochelo1.addNeck(neck)**;** Masks cap = new Masks("клетчатая кепка с широким козырьком")**;** cap.setType(MasksType.*CAP*)**;** choochelo1.addHelmet(cap)**;** String pants = "клечатые брюки"**;** choochelo1.addPants(pants)**;** Masks mask = new Masks("черная маска")**;** mask.setType(MasksType.*MASK*)**;** choochelo1.addMask(mask)**;** System.*out*.println(choochelo1.toString())**;** System.*out*.println(choochelo2.toString())**;** }  
  
}

enum’s:

public enum Locations {  
 *STREET***,** *SHOP***,** *RIGHTCORNER***,***LEFTCORNER***,** *CASHBOX*}

public enum MasksType {  
 *MAKEUP***,** *TOPCLOSING***,** *FULLCLOSING***,***HELMET***,** *WIGS***,** *BEARDS***,** *MUSTACHES***,** *CAP***,***SHAWL***,***MASK*}

public enum TypeOfGun {  
 *BATON***,***TORCH***,** *HANDCUFFS***,** *SHACKLES***,** *BOMB***,** *MASTERKEYSSET***,** *STEELFILE***,** *DRILL***,** *PLIER***,** *WIRECUTTER***,** *CROWBAR***,** *AUTOGEN***,** *GUN***,** *PISTOL***,** *KNIFE***,** *FINK***,** *DAGGER***,** *FLAIL***,** *KNUCKLE*}

exceptionClass:

public class AvailabilityException extends Exception {  
 public AvailabilityException(String message) {  
 super(message)**;** }  
}

interface’s:

interface Describe {  
 default String describe(){  
 return "использует что-то"**;**}  
}

public interface Move {  
 void move(Locations loc)**;**}

public interface Steal {  
 void stealing(Locations loc)**;**}

functional interface:

@FunctionalInterface  
public interface FuncInter<**T, N**> {  
 //N isTrue(T ta);  
 **N** convert(**T** t)**;** static <**T**> boolean isNotNull(**T** t){  
 return t != null**;** }  
}

Class’es:

public class ThiefsUnits extends Guns {  
 public ThiefsUnits(String name) {  
 super(name)**;** }  
  
 public String describe() {  
 return " крадет что-то с помощью "**;** }  
}

public class Firearms extends Guns {  
 public Firearms(String name) {  
 super(name)**;** }  
  
 public String describe() {  
 return "стреляет из "**;** }  
}

public class SteelArms extends Guns {  
 public SteelArms(String name) {  
 super(name)**;** }  
  
 public String describe() {  
 return " режет, используя "**;** }  
}

public class Masks {  
 private MasksType type**;** private final String name**;** private Integer price**;** public Masks(String name) {  
 this.name = name**;** System.*out*.println("Устанавливаем имя данной маски")**;** }  
  
 public String getName() {  
 return this.name**;** }  
  
 public void setType(MasksType type) {  
 System.*out*.println("Устанавливаем тип " + getName())**;** this.type = type**;** }  
  
 public void setPrice(int price) {  
 this.price = price**;** System.*out*.println("Устанавливаем цену " + getName())**;** }  
  
 public int getPrice() {  
 return this.price**;** }  
  
 public MasksType getType() {  
 return this.type**;** }  
}

public class Dunno extends Human implements Cloneable{  
 public Dunno(String name**,** int cash**,** Guns item) {  
 super(name**,** cash**,** item)**;** }  
  
 public Dunno(String name**,** int cash) {  
 super(name**,** cash)**;** }  
  
 public Dunno(int cash) {  
 super(cash)**;** }  
  
  
 @Override  
 public String toString() {  
 return "Незнайка {" +  
 "name='" + super.getName() + '\'' +  
 ", cash=" + super.getValue() + '}'**;** }  
  
}

public class Goat extends Human {  
 public Goat(String name**,** int cash) {  
 super(name**,** cash)**;** }  
  
 public Goat(int cash) {  
 super(cash)**;** }  
  
 @Override  
 public String toString() {  
 return "Козлик {" +  
 "name='" + super.getName() + '\'' +  
 ", cash=" + super.getValue() + '}'**;** }  
}

public class Masks {  
 private MasksType type**;** private final String name**;** private Integer price**;** public Masks(String name) {  
 this.name = name**;** System.*out*.println("Устанавливаем имя данной маски")**;** }  
  
 public String getName() {  
 return this.name**;** }  
  
 public void setType(MasksType type) {  
 System.*out*.println("Устанавливаем тип " + getName())**;** this.type = type**;** }  
  
 public void setPrice(int price) {  
 this.price = price**;** System.*out*.println("Устанавливаем цену " + getName())**;** }  
  
 public int getPrice() {  
 return this.price**;** }  
  
 public MasksType getType() {  
 return this.type**;** }  
}

static and inner class’es:

public class Shop {  
 private final String shopName**;** public Shop(String shopName) {  
 this.shopName = shopName**;** }  
  
 public class ShowCase {  
 private final String ShowCaseName**;** public ShowCase(String ShowCaseName) {  
 this.ShowCaseName = ShowCaseName**;** }  
  
 public void show(Firearms item) {  
 System.*out*.println("Зачетный ствол от зачетного производителя")**;** }  
  
 public void show(SteelArms item) {  
 System.*out*.println("Отличный вариант для техаской резни")**;** }  
  
 public void show(ThiefsUnits item) {  
 System.*out*.println("Хочешь что-то украсть? - тогда это тебе точно понадобиться ")**;** }  
 }  
  
 public static class Dummy implements Move**,** Steal {  
 private final String name**;** private Masks helmet**;** private Masks mask**;** private Masks neck**;** private Guns leftItem**;** private Guns rightItem**;** private String pants**;** public Dummy(String name) {  
 this.name = name**;** }  
  
 public Dummy() {  
 this.name = "известное в узких кругах"**;** }  
  
 public void addNeck(Masks neck) {  
 this.neck = neck**;** System.*out*.println("Шея чучела " + getName() + " повязана " + this.neck.getName())**;** }  
  
 public Masks getNeck() {  
 return this.neck**;** }  
  
 public void addPants(String pants) {  
 this.pants = pants**;** System.*out*.println(this.pants + " плотно облегали ноги чечела " + getName())**;** }  
  
 public String getPants() {  
 return this.pants**;** }  
  
 public void addHelmet(Masks helmet) {  
 this.helmet = helmet**;** System.*out*.println("На голове чучала " + getName() + " надет " + this.helmet.getName())**;** }  
  
 public void addMask(Masks mask) {  
 this.mask = mask**;** System.*out*.println("Лицо чучела " + getName() + " закрывала " + this.mask.getName())**;** }  
  
 public Masks getMask() {  
 return this.mask**;** }  
  
 public Masks getHelmet() {  
 return this.helmet**;** }  
  
 public String getName() {  
 return this.name**;** }  
  
 public Guns getLeftItem() {  
 return this.leftItem**;** }  
  
 public void setLeftItem(Guns leftItem) {  
 this.leftItem = leftItem**;** System.*out*.println("В левой руке чучала " + getName() + " находится " + this.leftItem.getName())**;** }  
  
 public Guns getRightItem() {  
 return this.rightItem**;** }  
  
 public void setRightItem(Guns rightItem) {  
 this.rightItem = rightItem**;** System.*out*.println("В правой руке чучала " + getName() + " находится " + this.rightItem.getName())**;** }  
  
 public void move(Locations Loc) {  
 System.*out*.println("Чучело " + getName() + " переместили в " + Loc)**;** }  
  
 public void stealing(Locations Loc) {  
 System.*out*.println("Чучело " + getName() + " крадется к " + Loc)**;** }  
  
 @Override  
 public String toString() {  
 return "Чучело " + getName() + " очень похож на настояшего живого коротышку"**;** }  
 }  
}

abstract class’es:

import java.util.ArrayList**;**import java.util.Arrays**;**import java.util.List**;**import java.util.Objects**;**public abstract class Human implements Move {  
 public String name**;** private int cash**;** private Guns item**;** private Masks mask**;** Guns[] items = new Guns[**3**]**;** public Human(String name**,** int cash) {  
 this.name = name**;** this.cash = cash**;** }  
  
 public Human(int cash) {  
 this.name = "неизвестный"**;** this.cash = cash**;** }  
  
 public Human(String name**,** int cash**,** Guns item) {  
 this.name = name**;** this.cash = cash**;** this.item = item**;** }  
  
 public void addValue(int addValue) {  
 this.cash += addValue**;** }  
  
 public int getValue() {  
 return this.cash**;** }  
  
 public void addMask(Masks mask) {  
 this.mask = mask**;** }  
  
 public Masks getMask() {  
 return this.mask**;** }  
  
 public Guns getLastItem() {  
 return items[items.length - **1**]**;** }  
  
 public void getMoney() {  
 System.*out*.println("У " + getName() + " " + getValue() + " денег")**;** }  
  
 public void getItems() {  
 List<String> itt = new ArrayList<>()**;** for (Guns elem : items) {  
 if (elem != null) {  
 itt.add(elem.getName())**;** }  
 }  
 itt.forEach(System.*out*::println)**;** }  
  
  
 public String getName() {  
 return this.name**;** }  
  
 @Override  
 public String toString() {  
 return "Человек {" +  
 "name='" + name + '\'' +  
 ", cash=" + cash + '}'**;** }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name**,** cash**,** Arrays.*hashCode*(items))**;** }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj)  
 return true**;** if (obj == null || getClass() != obj.getClass())  
 return false**;** Human person = (Human) obj**;** return cash == person.cash && name.equals(person.name) && items == person.items**;** }  
  
 public void buy(Guns item) {  
 if (Arrays.*asList*(items).contains(null)) {  
 if (item.getPrice() <= getValue()) {  
 for (int i = **0;** i < **3;** i++) {  
 if (items[i] == null) {  
 items[i] = item**;** System.*out*.println(getName() + " успешно покупает это оружие")**;** break**;** }  
 }  
 addValue(-item.getPrice())**;** } else {  
 System.*out*.println("Не хватает денег на это оружие")**;** }  
 } else {  
 System.*out*.println("Убери какое-нибудь оружие")**;** }  
 }  
  
 public void use(Guns item) throws AvailabilityException {  
 if (Arrays.*asList*(items).contains(item)) {  
 System.*out*.println(getClass().toString().substring(**6**) + " " + getName() + " " + item.describe() + item.getName())**;** } else {  
 throw new AvailabilityException("У него нет такого оружия " + item.getName())**;** }  
 }  
  
 public void move(Locations Loc) {  
 System.*out*.println(getClass().toString().substring(**6**) + " " + getName() + " перемещается в " + Loc)**;** }  
}

abstract public class Guns implements Describe {  
 private TypeOfGun type**;** private final String name**;** private Integer price**;** public Guns(String name) {  
 this.name = name**;** System.*out*.println("Устанавливаем имя данного оружия")**;** }  
  
 public String getName() {  
 return this.name**;** }  
  
 public void setType(TypeOfGun type) {  
 System.*out*.println("Устанавливаем тип " + getName())**;** this.type = type**;** }  
  
 public void setPrice(int price) {  
 this.price = price**;** System.*out*.println("Устанавливаем цену " + getName())**;** }  
  
 public int getPrice() {  
 return this.price**;** }  
  
 public TypeOfGun getType() {  
 return this.type**;** }  
  
}

# Результат работы программы

Устанавливаем имя данного оружия

Устанавливаем имя данного оружия

Устанавливаем тип огромный пистолет

Устанавливаем цену огромный пистолет

Устанавливаем цену потайной электрпический фонарь

Устанавливаем тип потайной электрпический фонарь

У тупенький 65 денег

Dunno тупенький перемещается в SHOP

Goat тупенький2 перемещается в SHOP

Не хватает денег на это оружие

У него нет такого оружия огромный пистолет

тупенький2 успешно покупает это оружие

тупенький успешно покупает это оружие

потайной электрпический фонарь

огромный пистолет

Dunno тупенький перемещается в STREET

Goat тупенький2 перемещается в STREET

У него нет такого оружия потайной электрпический фонарь

Незнайка {name='тупенький2', cash=6357}

Козлик {name='тупенький2', cash=6357}

Устанавливаем имя данной маски

Устанавливаем тип медная каска

Лицо чучела полицейский закрывала медная каска

Устанавливаем имя данного оружия

Устанавливаем тип дубинка

В левой руке чучала полицейский находится дубинка

Чучело полицейский переместили в RIGHTCORNER

Чучело грабитель крадется к CASHBOX

В левой руке чучала грабитель находится огромный пистолет

В правой руке чучала грабитель находится потайной электрпический фонарь

Устанавливаем имя данной маски

Устанавливаем тип пестрый клетчатый платок

Шея чучела грабитель повязана пестрый клетчатый платок

Устанавливаем имя данной маски

Устанавливаем тип клетчатая кепка с широким козырьком

На голове чучала грабитель надет клетчатая кепка с широким козырьком

клечатые брюки плотно облегали ноги чечела грабитель

Устанавливаем имя данной маски

Устанавливаем тип черная маска

Лицо чучела грабитель закрывала черная маска

Чучело грабитель очень похож на настояшего живого коротышку

Чучело полицейский очень похож на настояшего живого коротышку

# Вывод

Я научился обрабатывать исключения различных типов, использовать анонимные , вложенные и локальные классы.