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

Факультет программной инженерии и компьютерной техники

**Отчет**

По лабораторной работа № 4 по дисциплине “Программирование”

Вариант № 21128

Студент:

Мирзаитов Тимур

Группа P3112

Преподаватель:

Харитонова Анастасия Евгеньевна

Санкт-Петербург 2023

Оглавление

[Текст задания 2](#_Toc151488145)

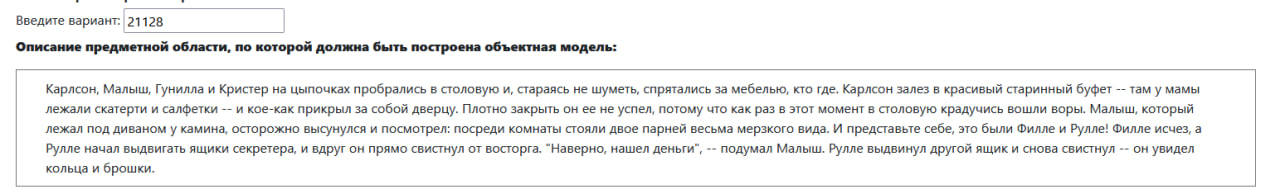
[Диаграмма классов объектной модели 3](#_Toc151488146)

[Исходный код программы 3](#_Toc151488147)

[Результат работы программы 7](#_Toc151488148)

[Выводы по работе 8](#_Toc151488149)

## Текст задания



Программа должна удовлетворять следующим требованиям:

Доработанная модель должна соответствовать принципам SOLID.

Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).

В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().

Программа должна содержать как минимум один перечисляемый тип (enum).

Порядок выполнения работы:

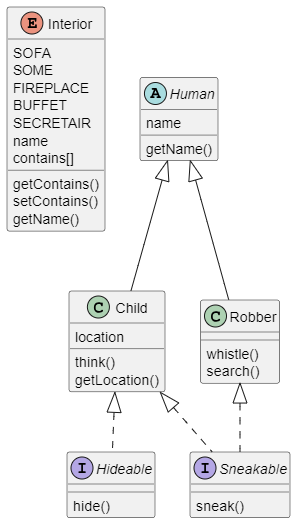
Доработать объектную модель приложения.

Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.

Согласовать с преподавателем изменения, внесённые в модель.

Модифицировать программу в соответствии с внесёнными в модель изменениями.

## Диаграмма классов объектной модели



## Исходный код программы

package ru.Timur;  
  
import java.util.prefs.PreferenceChangeListener;  
  
class Main {  
 public static void main(String[] args) {  
 Robber fille = new Robber("Fille");  
 Robber rulle = new Robber("Rulle");  
 Child malush = new Child("Malush");  
 Ghost karlson = new Ghost("Karlson");  
 Child gunilla = new Child("Gunilla");  
 Child crister = new Child("Crister");  
  
 Child[] children = {malush, karlson, gunilla, crister};  
 Robber[] robbers = {rulle, fille};  
  
 karlson.speak("Давайте тихонько пойдём в столовую — наверно, там твой отец хранит золотые слитки и бриллианты", "karlson");  
  
 for (Child character : children) {  
 String location = "столовую";  
 if(!location.equals("столовую")){  
 throw new LocationException();  
 }else {  
 character.sneak(location);  
 }  
 }  
  
 Interior buffet = Interior.*BUFFET*;  
 Interior some = Interior.*SOME*;  
 Interior sofa = Interior.*SOFA*;  
 Interior secretaire = Interior.*SECRETAIRE*;  
  
 malush.hide(sofa);  
 gunilla.hide(some);  
 crister.hide(some);  
 karlson.hide(buffet);  
  
 buffet.setContains(new String[]{"салфетки", "скатерти"});  
 secretaire.setContains(new String[]{"деньги", "кольцо", "брошки"});  
 *printContains*(buffet);  
  
 for (Robber character : robbers) {  
 String location = "столовую";  
 if(!location.equals("столовую")){  
 throw new LocationException();  
 }else {  
 character.sneak(location);  
  
 }  
 }  
  
 rulle.search(secretaire);  
 rulle.whistle();  
 malush.think("Наверно, нашел деньги");  
 rulle.stopWhistle();  
 do {  
 try {  
 karlson.fly();  
 } catch (EngineException e) {  
 System.*out*.println("Попытка запустить двигатель");  
 karlson.engine.startEngine();  
 }  
 }while(!karlson.isFly);  
  
 buffet.open();  
  
 karlson.makeSound(karlson.new Sound("страшный стон"));  
  
 rulle.lookBack();  
  
 rulle.makeSound("хрюкнул");  
  
 rulle.drop("всё");  
  
 try{  
 karlson.fly();  
 }catch (EngineException e)  
 {  
 //do nothing  
 }  
  
 karlson.makeSound(karlson.new Sound("стон и вздох"));  
  
 karlson.flyTo("кухню");  
  
  
  
  
  
 karlson.breakEngine(new Ghost.Engine(){  
 @Override  
 public void offEngine(){  
 this.setStart(false);  
 System.*out*.println("Двигатель взорвался");  
 }  
 });  
 }  
  
 static void printContains(Interior thing) {  
 System.*out*.print(thing.getName() + " содержит: ");  
 for (String a : thing.getContains()) {  
 System.*out*.print(a + " ");  
 }  
 System.*out*.println();  
 }  
  
  
  
}

package ru.Timur;  
  
import java.util.Objects;  
  
public class Robber extends Human implements Sneakable, LookBack, Dropable {  
 String location = null;  
  
 public Robber(String name) {  
 super(name);  
 }  
  
 @Override  
 public void sneak(String location) {  
 this.location = location;  
 System.*out*.println(getName() + " крадется в " + location);  
 }  
  
 public String getLocation(){  
 return location;  
 }  
  
 public void whistle() {  
 System.*out*.println(getName() + " свистит");  
 }  
  
 public void stopWhistle(){  
 System.*out*.println(getName() + " перестал свистеть");  
 }  
 public void search(Interior where) {  
 System.*out*.print(this.getName() + " нашел: ");  
 for (String a : where.getContains()) {  
 System.*out*.print(a + " ");  
 }  
 System.*out*.println();  
 }  
  
 public void makeSound(String type) {  
 class Sound{  
 String typeOfSound;  
 public Sound(String sound){  
 typeOfSound = sound;  
 }  
  
 public String getTypeOfSound() {  
 return typeOfSound;  
 }  
 }  
 Sound sound = new Sound(type);  
 System.*out*.println(this.getName() + " " + sound.getTypeOfSound());  
 }  
  
 @Override  
 public void lookBack(){  
 System.*out*.println(this.getName() + " обернулся");  
 }  
  
 public void drop(String things){  
 System.*out*.println(this.getName() + " уронил " + things);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getName());  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj == this) {  
 return true;  
 }  
 if (obj.getClass() != this.getClass()) {  
 return false;  
 }  
 Robber that = (Robber) obj;  
 return Objects.*equals*(that.getName(), getName());  
 }  
  
 @Override  
 public String toString() {  
 return "Robber{" +  
 "name: " + this.getName() +  
 "}";  
 }  
}

package ru.Timur;  
  
import java.util.Objects;  
  
public class Child extends Human implements Sneakable, Hideable, Speakable {  
 private Interior location; // поменять тип данных  
  
 public String getMainLocation() {  
 return mainLocation;  
 }  
  
 private String mainLocation;  
 public Child(String name) {  
 super(name);  
 }  
  
  
 @Override  
 public void sneak(String location) {  
 System.*out*.println(getName() + " крадется в " + location);  
 mainLocation = location;  
 }  
  
 @Override  
 public void hide(Interior loc) {  
 location = loc;  
 System.*out*.println(getName() + " прячется, используя " + location.getName());  
 }  
  
 @Override  
 public void speak(String phrase, String author){  
 System.*out*.println("\"" + phrase + "\" сказал " + author);  
 }  
  
 public void think(String think) {  
 System.*out*.println(getName() + " думает: \"" + think + "\"");  
 }  
  
 public Interior getLocation() {  
 return location;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Child child = (Child) o;  
 return location == child.location;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(location);  
 }  
  
 @Override  
 public String toString() {  
 return "Child{" +  
 "name:" + this.getName() +  
 "location" + location.toString() +  
 "}";  
 }  
}

package ru.Timur;  
  
public class Ghost extends Child {  
 Engine engine = new Engine();  
 boolean isFly = false;  
 public Ghost(String name){  
 super(name);  
 }  
  
 static public class Engine{  
 private boolean isStart = false;  
  
 public Engine(){}  
 public void startEngine(){  
  
 isStart = true;  
 System.*out*.println("Двигатель запущен");  
 }  
 public void offEngine(){  
 isStart = false;  
 System.*out*.println("Двигатель выключен");  
 }  
 public boolean getStart(){  
 return isStart;  
 }  
  
 public void setStart(boolean isStart){  
 this.isStart = isStart;  
 }  
  
 }  
  
 public void breakEngine(Engine engine){  
 engine.offEngine();  
 }  
  
 public void fly() throws EngineException {  
 if(engine.getStart()){  
 System.*out*.println("Привидение порхает");  
 isFly = true;  
 }else{  
 throw new EngineException("Двигатель не включен");  
 }  
 }  
  
 public class Sound{  
 String typeOfSound = null;  
  
 public Sound(String type){  
 typeOfSound = type;  
 }  
  
 public String getTypeOfSound(){  
 return typeOfSound;  
 }  
 }  
  
 public void makeSound(Sound sound){  
 System.*out*.println("Приведение издовало:" + sound.getTypeOfSound());  
 }  
  
 public void flyTo(String location){  
 System.*out*.println("устремилось в " + location);  
 }  
  
 public Engine getEngine() {  
 return engine;  
 }  
}

package ru.Timur;  
  
import java.util.Objects;  
  
public abstract class Human {  
 private String name;  
  
 public Human(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Human human = (Human) o;  
 return Objects.*equals*(name, human.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
 @Override  
 public String toString() {  
 return "Human{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
}

package ru.Timur;  
  
import java.util.Arrays;  
  
public enum Interior {  
 *SOFA*("диван"),  
 *SOME*("что-то"),  
 *FIREPLACE*("камин"),  
 *BUFFET*("буфет"){  
 public void open(){  
 System.*out*.println("Буфет распахнулся");  
 }  
 },  
 *SECRETAIRE*("секретар");  
  
 private final String name;  
 String[] contains;  
  
 Interior(String rus\_name) {  
 name = rus\_name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setContains(String[] contains) {  
 this.contains = contains;  
 }  
  
 public String[] getContains() {  
 return contains;  
 }  
  
 @Override  
 public String toString() {  
 return "Interior{" +  
 "name='" + name + '\'' +  
 ", contains=" + Arrays.*toString*(contains) +  
 '}';  
 }  
  
 public void open() {}  
}

package ru.Timur;  
  
//Позволяет объекту краться  
public interface Sneakable {  
 void sneak(String string);  
}

package ru.Timur;  
  
public interface Speakable {  
 void speak(String phrase, String author);  
}

package ru.Timur;  
  
// Даёт возможность прятаться  
public interface Hideable {  
 void hide(Interior obj);  
}

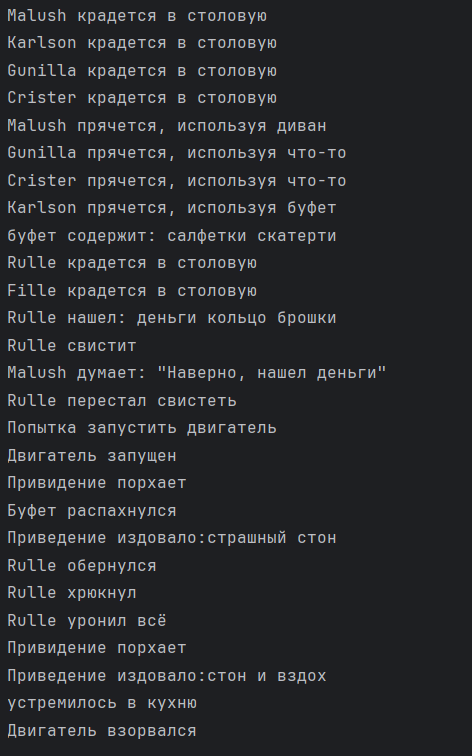
package ru.Timur;  
  
public interface Dropable {  
 void drop(String thing);  
}

package ru.Timur;  
  
public class EngineException extends Exception{  
 public EngineException(String message){  
 super(message);  
 }  
}

package ru.Timur;  
  
public class LocationException extends RuntimeException{  
 public LocationException(){  
 super();  
 }  
}

package ru.Timur;  
  
public interface LookBack {  
 void lookBack();  
}

## Результат работы программы



## Выводы по работе

В ходе данной работы я узнал что бывают разные Вложенные классы: inner, nested, анонимные, локальные. Как работают разные виды исключений