Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №4

Вариант 332

Преподаватель: Письмак Алексей Евгеньевич

Выполнил: Кузнецов Даниил Александрович

Р3134

Санкт-Петербург

2023

# Задание:

**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.

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

**Вопросы к защите лабораторной работы:**

1. Обработка исключительных ситуаций, три типа исключений.
2. Вложенные, локальные и анонимные классы.
3. Механизм рефлексии (reflection) в Java. Класс Class.

# Описание предметной области:

Но прежде чем Малыш успел ответить, Карлсон, ликуя, поднял руку с пистолетом над головой и выстрелил. Резкий звук прокатился по крышам и замер. В соседних домах загудели голоса: то испуганные, то сердитые, а кто-то крикнул, что нужно вызвать полицию. Тут Малыш совсем вышел из себя. Но Карлсон сидел с невозмутимым видом и жевал булочку, уже последнюю.

# Диаграмма классов:

# 

# Исходный код:

### Main.java – класс, который отвечает за запуск сцены.

// v332  
  
import Scene.Scene;  
  
public class Main {  
 public static void main(String[] args) {  
 Scene carlsonScene = new Scene("Сцена с Карлсоном", "Карлсон", "Малыш");  
 carlsonScene.run();  
 }  
}

# Scene.java – класс, который отвечает за создание сцены и персонажей в ней

package Scene;  
  
import Scene.Enums.voicesCharacteristic;  
import Scene.Heroes.\*;  
import Scene.InanimatedObjects.\*;  
import Scene.Interfaces.InanimateActions;  
  
import java.util.Objects;  
  
public class Scene {  
 final Carlson carlson;  
 final Boy boy;  
 final Sound sound;  
 final Voices voices;  
 final Somebody somebody;  
 final String name;  
  
 public Scene(String name, String carlson\_name, String boy\_name) {  
 carlson = new Carlson(carlson\_name, this, Math.*random*() <= 0.2 ? 0 : 1);  
 boy = new Boy(boy\_name, this);  
 sound = new Sound(this);  
 voices = new Voices(this);  
 somebody = new Somebody(this);  
 this.name = name;  
 }  
  
 public Carlson getCarlson() {  
 return carlson;  
 }  
  
 public Boy getBoy() {  
 return boy;  
 }  
  
 public Voices getVoices(){  
 return voices;  
 }  
  
 public Sound getSound() {  
 return sound;  
 }  
  
 public Somebody getSomebody() {  
 return somebody;  
 }  
  
 public String getName(){  
 return name;  
 }  
  
 public void run(){  
 System.*out*.println("\n");  
 boy.startAction();  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Scene))  
 return false;  
  
 Scene tmp = (Scene) o;  
  
 return tmp.getCarlson().equals(this.getCarlson()) &&  
 tmp.getBoy().equals(this.getBoy()) &&  
 tmp.getSound().equals(this.getSound()) &&  
 tmp.getSomebody().equals(this.getSomebody()) &&  
 tmp.getVoices().equals(this.getVoices());  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(carlson, boy, sound, voices, somebody);  
 }  
  
 @Override  
 public String toString() {  
 return "Scene{Name: " + getName() + ", " +  
 "Carlson: " + getCarlson().toString() + ", " +  
 "Boy: " + getBoy().toString() + ", " +  
 "Sound: " + getSound().toString() + ", " +  
 "Voices: " + getVoices().toString() + ", " +  
 "Somebody: " + getSomebody().toString() + "}";  
 }  
  
 // Nested class  
 public class Voices extends StoryObject implements InanimateActions {  
 public Voices(Scene scene){  
 super(scene);  
 }  
  
 @Override  
 public void generalAction() {  
 System.*out*.print("В соседних домах загудели голоса: ");  
 System.*out*.print("то " + voicesCharacteristic.*scared*.getTitle() + ", то "  
 + voicesCharacteristic.*angry*.getTitle() + ", ");  
 }  
  
 @Override  
 public void misfireAction() {  
 System.*out*.print("В соседних домах царила тишина, ");  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Voices))  
 return false;  
  
 Voices tmp = (Voices) o;  
  
 return tmp.getScene() == this.getScene();  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(this.getScene());  
 }  
  
 public String toString() {  
 return "Voices{Scene: " + super.getScene().getName() + "}";  
 }  
 }  
}

# Hero.java – абстрактный класс, который выступает первоначальным для всех героев

# StoryObject.java – абстрактный класс, который выступает первоначальным для всех неживых объектов

# Boy.java – класс, который отвечает за Малыша

package Scene.Heroes;  
  
import Scene.Interfaces.BoyActions;  
import Scene.Scene;  
  
import java.util.Objects;  
  
public class Boy extends Hero implements BoyActions {  
 public Boy(String boy\_name, Scene scene){  
 super(boy\_name, scene);  
 }  
  
 @Override  
 public void startAction() {  
 System.*out*.print("Но прежде чем ");  
 System.*out*.print(super.getName());  
 System.*out*.print(" успел ответить, ");  
 super.getScene().getCarlson().startAction();  
 }  
  
 @Override  
 public void getAngry() {  
 System.*out*.println("Тут " + super.getName() + " совсем вышел из себя. ");  
 }  
  
 @Override  
 public void voicesReaction() {  
 getAngry();  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Boy))  
 return false;  
  
 Boy tmp = (Boy) o;  
  
 return tmp.getName().equals(this.getName()) && tmp.getScene() == this.getScene();  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(this.getName(), this.getScene());  
 }  
  
 @Override  
 public String toString() {  
 return "Boy{Scene: " + super.getScene().getName() + ", " +  
 "Name: " + super.getName() + "}";  
 }  
}

# Carlson.java – класс, который отвечает за Карлсона

package Scene.Heroes;  
  
import Scene.Interfaces.Bun;  
import Scene.Interfaces.CarlsonActions;  
import Scene.Scene;  
import Scene.Exceptions.misfireException;  
import Scene.Exceptions.noGunException;  
import java.util.Objects;  
  
public class Carlson extends Hero implements CarlsonActions {  
 private int gun;  
  
 public Carlson(String carlson\_name, Scene scene, int is\_gun) {  
 super(carlson\_name, scene);  
 gun = is\_gun;  
 }  
  
 @Override  
 public void startAction(){  
 System.*out*.print(getName());  
 System.*out*.print(", ликуя, ");  
  
 try {  
 fired();  
 }  
 catch (misfireException | noGunException e){  
 System.*out*.println(e.getMessage());  
 super.getScene().getSound().misfireAction();  
 super.getScene().getVoices().misfireAction();  
 super.getScene().getSomebody().misfireAction();  
 }  
 }  
  
 @Override  
 public void fired() throws misfireException, noGunException {  
 if (gun == 0){  
 throw new noGunException("хотел выстрелить, но вспомнил, что не взял пистолет.");  
 }  
  
 System.*out*.print("поднял руку с пистолетом над головой и выстрелил. ");  
  
 if (Math.*random*() <= 0.25) {  
 throw new misfireException("Пистолет дал осечку. ");  
 }  
 else {  
 super.getScene().getSound().generalAction();  
 super.getScene().getVoices().generalAction();  
 super.getScene().getSomebody().generalAction();  
 }  
 }  
  
 @Override  
 public void voicesReaction() {  
 System.*out*.print("Но " + super.getName() + " ");  
 sat();  
 }  
  
 @Override  
 public void sat() {  
 System.*out*.print("сидел с невозмутимым видом и ");  
 eat();  
 }  
  
 // static nested class  
 public static class BunCond{  
 private static String *cond1*;  
  
 public static void setCond(String cond){  
 *cond1* = cond;  
 }  
 public static String getCond(){  
 return *cond1*;  
 }  
 }  
  
 @Override  
 public void eat() {  
  
 // anonymous local nested class  
 Bun bun = new Bun(){  
 @Override  
 public String getCondition() {  
 return BunCond.*getCond*();  
 }  
  
 @Override  
 public void setCondition(String cond) {  
 BunCond.*setCond*(cond);  
 }  
 };  
  
 bun.setCondition("уже последнюю");  
 System.*out*.println("жевал булочку, " + bun.getCondition() + ". ");  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Carlson))  
 return false;  
  
 Carlson tmp = (Carlson) o;  
  
 return tmp.getName().equals(this.getName()) && tmp.getScene() == this.getScene();  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(this.getName(), this.getScene());  
 }  
  
 @Override  
 public String toString() {  
 return "Carlson{Scene: " + super.getScene().getName() + ", " +  
 "Name: " + super.getName() + "}";  
 }  
}

# Somebody.java – класс, который отвечает за толпу людей

package Scene.InanimatedObjects;  
  
import Scene.Interfaces.InanimateActions;  
import Scene.Scene;  
  
import java.util.Objects;  
  
public class Somebody extends StoryObject implements InanimateActions {  
 public Somebody(Scene scene) {  
 super(scene);  
 }  
  
 @Override  
 public void generalAction() {  
 System.*out*.print("а кто то крикнул, что нужно вызывать полицию. ");  
 super.getScene().getBoy().voicesReaction();  
 super.getScene().getCarlson().voicesReaction();  
 }  
  
 @Override  
 public void misfireAction() {  
 System.*out*.print("люди спокойно занимались своими делами. \n");  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Somebody))  
 return false;  
  
 Somebody tmp = (Somebody) o;  
  
 return tmp.getScene() == this.getScene();  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(this.getScene());  
 }  
  
 @Override  
 public String toString() {  
 return "Somebody{Scene: " + super.getScene().getName() + "}";  
 }  
}

# 

# Sound.java – класс, который отвечает за звук

package Scene.InanimatedObjects;  
  
import Scene.Interfaces.InanimateActions;  
import Scene.Scene;  
  
import java.util.Objects;  
  
public class Sound extends StoryObject implements InanimateActions {  
 public Sound(Scene scene){  
 super(scene);  
 }  
  
 @Override  
 public void generalAction() {  
 System.*out*.println("Резкий звук прокатился по крышам и замер.");  
 }  
  
 @Override  
 public void misfireAction() {  
 System.*out*.print("Резкий звук выстрела не раздался. ");  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if (o == this)  
 return true;  
  
 if (!(o instanceof Sound))  
 return false;  
  
 Sound tmp = (Sound) o;  
  
 return tmp.getScene() == this.getScene();  
 }  
  
 @Override  
 public int hashCode(){  
 return Objects.*hash*(this.getScene());  
 }  
  
 public String toString() {  
 return "Sound{Scene: " + super.getScene().getName() + "}";  
 }  
}

# misfireException.java – класс исключения, которое выбрасывается в случае осечки пистолета

package Scene.Exceptions;  
  
public class misfireException extends RuntimeException {  
 public misfireException(String message){  
 super(message);  
 }  
}

# noGunException.java – класс исключения, которое выбрасывается в случае отсутствия пистолета у Карлсона

package Scene.Exceptions;  
  
import java.io.IOException;  
  
public class noGunException extends IOException {  
 public noGunException(String message){  
 super(message);  
 }  
}

# VoicesCharacteristic.java – класс перечисляемого типа, который описывает состояние голосов толпы

package Scene.Enums;  
  
public enum voicesCharacteristic {  
 *scared*("испуганные"),  
 *angry*("сердитые");  
  
 private String title;  
  
 voicesCharacteristic(String title) {  
 this.title = title;  
 }  
  
 public String getTitle(){  
 return title;  
 }  
}

# BoyActions.java – интерфейс, описывающий поведение Малыша

package Scene.Interfaces;  
  
public interface BoyActions {  
 public void getAngry();  
}

# Bun.java – интерфейс батона для создания анонимного класса

package Scene.Interfaces;  
  
public interface Bun {  
 String getCondition();  
 void setCondition(String cond);  
}

# CarlsonActions.java – интерфейс, описывающий поведение Карлсона

package Scene.Interfaces;  
  
import Scene.Exceptions.noGunException;  
  
public interface CarlsonActions {  
 public void fired() throws noGunException;  
 public void sat();  
 public void eat();  
}

# InanimatedActions.java – интерфейс, описывающий поведение неживых объектов

package Scene.Interfaces;  
  
public interface InanimateActions {  
 public void generalAction();  
 public void misfireAction();  
}

Результаты работы

**#1**

Но прежде чем Малыш успел ответить, Карлсон, ликуя, поднял руку с пистолетом над головой и выстрелил. Резкий звук прокатился по крышам и замер.

В соседних домах загудели голоса: то испуганные, то сердитые, а кто то крикнул, что нужно вызывать полицию. Тут Малыш совсем вышел из себя.

Но Карлсон сидел с невозмутимым видом и жевал булочку, уже последнюю.

**#2**

Но прежде чем Малыш успел ответить, Карлсон, ликуя, поднял руку с пистолетом над головой и выстрелил. Пистолет дал осечку.

Резкий звук выстрела не раздался. В соседних домах царила тишина, люди спокойно занимались своими делами.

**#3**

Но прежде чем Малыш успел ответить, Карлсон, ликуя, хотел выстрелить, но вспомнил, что не взял пистолет.

Резкий звук выстрела не раздался. В соседних домах царила тишина, люди спокойно занимались своими делами.

# Вывод:

В процессе выполнения лабораторной работы были получены навыки использования объектно-ориентированного подхода программирования при использовании языка Java.