

Факультет ПИ и КТ

Лабораторная работа №4 по Программированию

Вариант 150150.13

Выполнил:

Болорболд Аригуун

Группа P3111

Преподаватель:

Харитонова Анастасия Евгеньевна

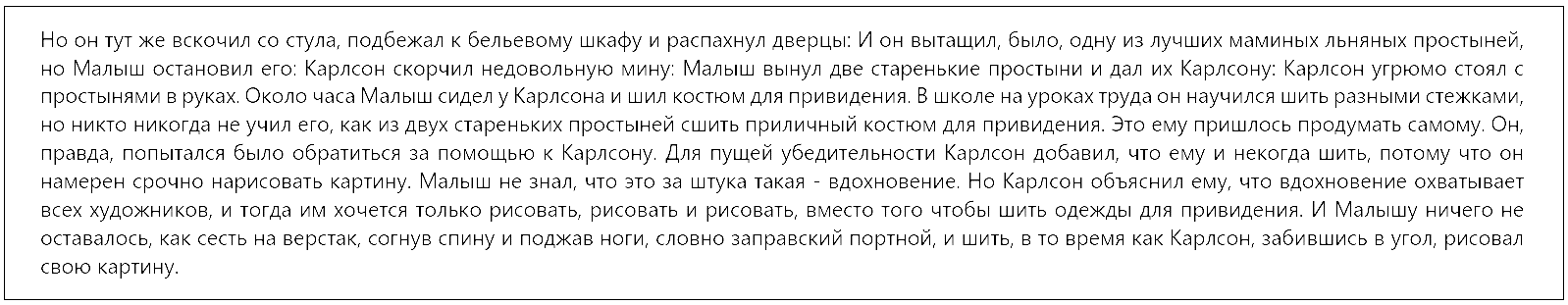
Санкт-Петербург

2023

**Текст задания:**

Доработать программу из лабораторной работы #3, обновив реализацию объектной модели в соответствии с новой версией описания предметной области.

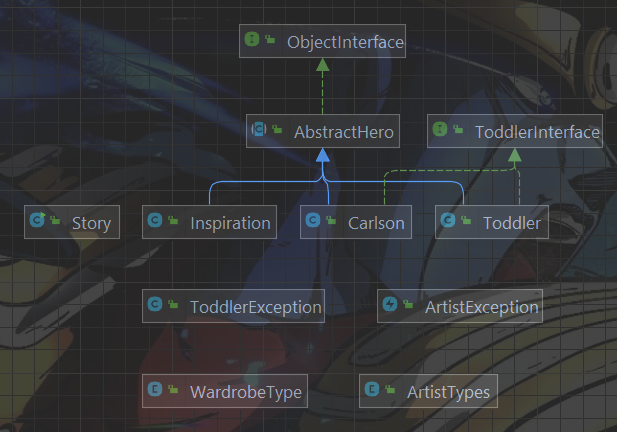
**Описание предметной области, по которой должна быть пост4роена объектная модель:**



**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).

**Диаграмма классов реализованной объектной модели:**

****

**Исходный код программы:**

**Utilities:**

1. **ObjectInterface.java:**

package utilities;  
  
public interface ObjectInterface {  
 void joinToStory();  
 void leaveFromStory();  
 }  
}

1. **AbstractHero.java**

package utilities;  
  
public abstract class AbstractHero implements ObjectInterface {  
 protected String name;  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
}

1. **ToddlerInterface.java**

package utilities;  
  
public interface ToddlerInterface {  
 String know(boolean doesKnow);  
}

1. **ArtistTypes.java**

package utilities;  
  
public enum ArtistTypes {  
 *PAINTER*,  
 *SCULPTOR*,  
 *CARVER*,  
 *MUSICIAN*,  
 *CINEMATOGRAPHER*}

1. **WardrobeType.java**

package utilities;  
  
public enum WardrobeType {  
 *GENERAL*,  
 *LINEN*,  
 *UNDERWEAR*}

**Exceptions:**

1. **ArtistException.java**

package exceptions;  
public class ArtistException extends Exception {  
 private final String message;  
 public ArtistException(String message) {  
 super(message);  
 this.message = message;  
 }  
 @Override  
 public String getMessage(){  
 return message;  
 }  
}

1. **ToddlerException.java**

package exceptions;  
public class ToddlerException extends RuntimeException{  
 private final String message;  
 public ToddlerException(String message) {  
 super(message);  
 this.message = message;  
 }  
 @Override  
 public String getMessage(){  
 return message;  
 }  
}

**Characters:**

1. **Carlson.java**

package characters;  
import exceptions.ArtistException;  
import utilities.\*;  
  
public class Carlson extends AbstractHero implements ToddlerInterface {  
 public static String *name*;  
 public Carlson() {  
 *name* = "Карлсон";  
 joinToStory();  
 }  
 public Carlson(String name) {  
 Carlson.*name* = name;  
 joinToStory();  
 }  
 @Override  
 public void joinToStory() {  
 System.*out*.println("Карлсон '" + *name* + "' присоединился к рассказу.");  
 }  
 @Override  
 public String pullOut(boolean isOld){  
 class Sheet {  
 private final String shName;  
 public Sheet(String shName) {  
 this.shName = shName;  
 }  
 }  
 String Age;  
 if(isOld){  
 Age = "старенькие";  
 } else {  
 Age = "новенькие";  
 }  
 Sheet sheet = new Sheet("простыни");  
 return "вынул" + Age + " " + sheet.shName + ".";  
 }  
 public void give(Toddler toddler){  
 System.*out*.println("и дал их " + toddler.toString());  
 }  
 public void scrunch(){  
 System.*out*.println("Карлсон '" + *name* + "' скорчил недовольную мину.");  
 }  
 public String stand(boolean grumpy){  
 String menacing;  
 if(grumpy){  
 menacing = "угрюмо";  
 } else {  
 menacing = "весело";  
 }  
 return "Карлсон '" + *name* + "' " + menacing + " стоял с простынями в руках.";  
 }  
 public String sew(boolean hasTime) {  
 String will;  
 if(hasTime){  
 will = "нынче";  
 } else {  
 will = "некогда";  
 }  
 return "Карлсону " + will + " шить одежду";  
 }  
 public String know(boolean doesKnow) throws ArtistException {  
 if (!doesKnow) {  
 throw new ArtistException("Чё блин");  
 } else {  
 return "В отличие от него, Карлсон '" + *name* + "' отлично знал, что это за штука, так как желание артиста творить зависит от его вдохновения.";  
 }  
 }  
 public String paint(boolean hasTime) {  
 String intent;  
 if(!hasTime) {  
 intent = "не намерен";  
 } else {  
 intent = "намерен";  
 }  
 return ", потому что он " + intent + " срочно нарисовать картину.";  
 }  
 public void crammed(boolean inCorner){  
 String claustrophilia;  
 if(inCorner){  
 claustrophilia = "забившись в угол, ";  
 } else {  
 claustrophilia = "в середине комнаты, ";  
 }  
 System.*out*.println("Карлсон '" + *name* + "', " + claustrophilia + "рисовал свою картину.");  
 }  
 public void explain() {  
 System.*out*.println("Карлсон объяснил, что...");  
 }  
 @Override  
 public void leaveFromStory() {  
 System.*out*.println("Карлсон " + *name* + " ливнул с рассказа.");  
 }  
  
 @Override  
 public String getName() {  
 return *name*;  
 }  
 @Override  
 public void setName(String name){  
 this.*name* = name;  
 }  
 @Override  
 public String toString() {  
 return "Карлсон '" + *name* + "'";  
 }  
 @Override  
 public boolean equals(Object obj) {  
 if(this == obj) return true;  
 if(obj instanceof Carlson){  
 return *name*.equals(((Carlson) obj).getName());  
 }  
 return false;  
 }  
 @Override  
 public int hashCode() {  
 return *name*.hashCode();  
 }  
}

1. **Toddler.java**

package characters;  
  
import exceptions.ToddlerException;  
import utilities.\*;  
public class Toddler extends AbstractHero implements ToddlerInterface {  
 private static String *name*;  
 public Toddler(){  
 *name* = "Малыш";  
 joinToStory();  
 }  
 public Toddler(String name){  
 Toddler.*name* = name;  
 joinToStory();  
 }  
 @Override  
 public void joinToStory(){  
 System.*out*.println("Малыш '" + *name* + "' присоединился к рассказу.");  
 }  
 @Override  
 public void leaveFromStory() {  
 System.*out*.println("Малыш" + *name* + "ливнул с рассказа.");  
 }  
 public String jump(String cName) {  
 class Stool{  
 private final String cName;  
 public Stool(String cName) {  
 this.cName = cName;  
 }  
 }  
 Stool stool = new Stool(cName);  
 return "Но он (" + *name* + ") подскочил со стула '" + stool.cName + "', ";  
 }  
 public String sprint(String wName){  
 String Type = "";  
 WardrobeType wardrobeType = WardrobeType.*LINEN*;  
 switch (wardrobeType){  
 case *GENERAL*:  
 Type = " ";  
 break;  
 case *LINEN*:  
 Type = "бельевому";  
 break;  
 case *UNDERWEAR*:  
 Type = "нижнебельевому";  
 }  
 class Wardrobe{  
 private final String wName;  
 public Wardrobe(String wName){  
 this.wName = wName;  
 }  
 }  
 Wardrobe wardrobe = new Wardrobe("шкаф");  
 return "подбежал к " + Type + " шкафу '" + wardrobe.wName + "', ";  
 }  
 public String openDoor(String dName){  
 return "и распахнул дверцы '" + dName + "'.";  
 }  
 @Override  
 public String pullOut(boolean isOld){  
 class Sheet {  
 private final String shName;  
 public Sheet(String shName) {  
 this.shName = shName;  
 }  
 }  
 String Age;  
 if(isOld){  
 Age = "стареньких";  
 } else {  
 Age = "новеньких";  
 }  
 Sheet sheet = new Sheet("простыней");  
 return "И вытащил одну из " + Age + " " + sheet.shName + ".";  
 }  
 public String give(){  
 return "Малыш '" + *name* + "' вынул два стареньких простыней и дал их Карлсону '" + Carlson.*name* + "'.";  
 }  
 @Override  
 public String know(boolean doesKnow) throws ToddlerException {  
 if (doesKnow){  
 throw new ToddlerException("\*?");  
 } else {  
 return "Малыш '" + *name* + "' не знал, что это за штука такая -- вдохновение.";  
 }  
 }  
 @Override  
 public String sew(boolean hasTime){  
 Runnable costume = () -> System.*out*.print("Около часа шил костюму для привидения ");  
 costume.run();  
 return "Малыш '" + *name* + "'.";  
 }  
 public class Knowledge implements ObjectInterface{  
 public String name;  
 public Knowledge(String name){  
 this.name = name;  
 }  
 @Override  
 public void joinToStory(){  
 System.*out*.println("Знание " + name + " присоединился к рассказу.");  
 }  
 @Override  
 public void leaveFromStory(){  
 System.*out*.println("Знание " + name + " ливнул с рассказа.");  
 }  
 public String isTaught(boolean isTaught){  
 if (isTaught){  
 return "В школе на уроках трудам он научился шить разными стежками.";  
 } else {  
 return "В школе ему никто не научил шить разными стежками.";  
 }  
 }  
 public String magicSew(boolean isTaught){  
 if (isTaught){  
 return "Плюс к этому, он (" + Toddler.*name* + ") маг -- умел сшить костюму из двух простыней.";  
 } else {  
 return "Но никто ему (" + Toddler.*name* + "у) не учил, как из двух стареньких простыней сшить костюму.";  
 }  
 }  
 public String figOut(){  
 return "Ему (" + Toddler.*name* + "у) пришлось придумать это самому.";  
 }  
 }  
 public static String askForHelp(boolean helpless, boolean will){  
 if(helpless && will){  
 return "Он '" + *name* + "' попытался обратиться за помощью к Карлсону '" + Carlson.*name* + "'.";  
 } else {  
 return "Ему (" + *name* + ") не надо было обратиться к Карлсону" + Carlson.*name* + "за помощью.";  
 }  
 }  
 public static class ParallelActions{  
 public String name;  
 public ParallelActions(String name){  
 this.name = name;  
 }  
 public boolean isDenied = true;  
 public String sit(boolean posture){  
 class CraftingTable{  
 public String tName;  
 public CraftingTable(String tName){  
 this.tName = tName;  
 }  
 }  
 CraftingTable craftingTable = new CraftingTable("верстак");  
 if(isDenied || !posture){  
 return "И Малышу '" + Toddler.*name* + "' ничего не оставалось, как сесть на верстак '" + craftingTable.tName + "'," + " согнув спину и поджав руки, ";  
 } else {  
 return "А Малышу '" + Toddler.*name* + "' делать стало много, прежде чем сесть на верстак '" + craftingTable.tName + "'," + " не согнув спину и поджав руки, ";  
 }  
 }  
 public void continueSew(boolean isDenied){  
 if(isDenied){  
 System.*out*.println("продолжал шить.");  
 } else {  
 System.*out*.println("перестал шить.");  
 }  
 }  
 }  
 @Override  
 public String getName(){  
 return *name*;  
 }  
 @Override  
 public void setName(String name){  
 this.*name* = name;  
 }  
 @Override  
 public String toString(){  
 return "Малышу '" + *name* + "'";  
 }  
 @Override  
 public boolean equals(Object obj){  
 if(this == obj) return true;  
 if(obj instanceof Toddler){  
 return *name*.equals(((Toddler) obj).getName());  
 }  
 return false;  
 }  
 @Override  
 public int hashCode(){  
 return *name*.hashCode();  
 }  
}

1. **Inspiration.java**

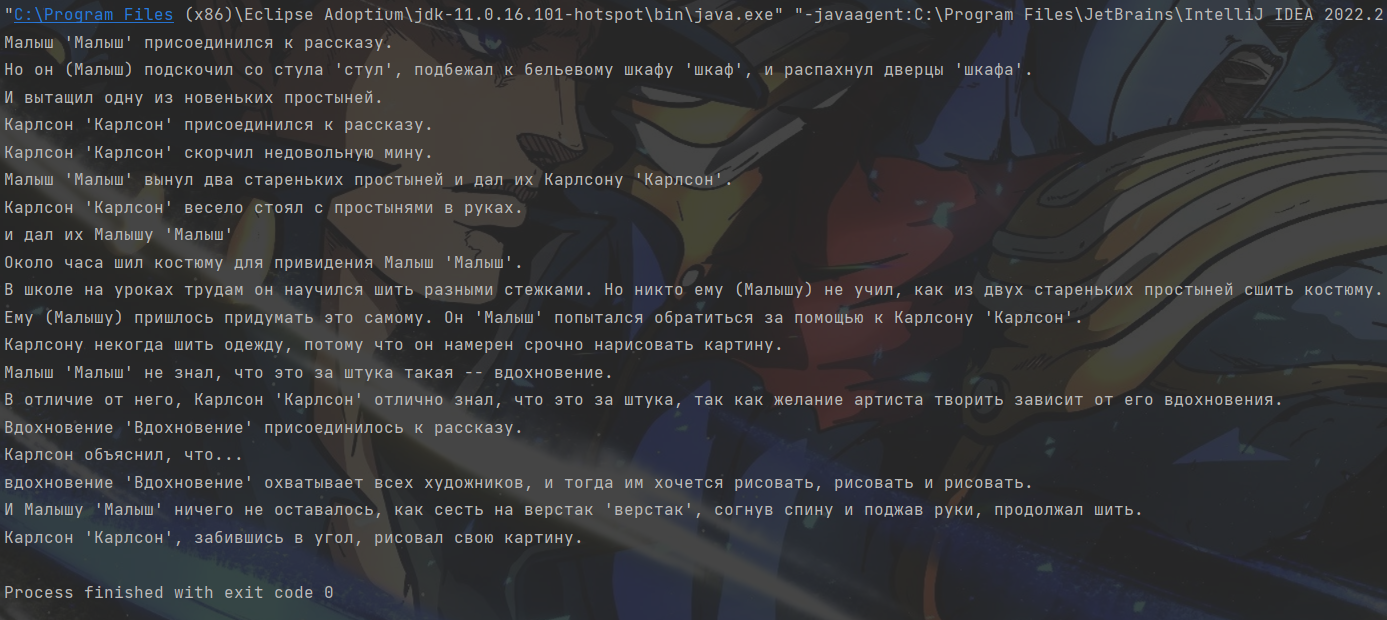
package characters;  
  
import utilities.\*;  
public class Inspiration extends AbstractHero{  
 private String name;  
 public Inspiration() {  
 name = "Вдохновение";  
 joinToStory();  
 }  
 public Inspiration(String name) {  
 this.name = name;  
 joinToStory();  
 }  
 @Override  
 public void joinToStory() {  
 System.*out*.println("Вдохновение '" + name + "' присоединилось к рассказу.");  
 }  
  
 @Override  
 public void leaveFromStory() {  
 System.*out*.println("Вдохновение "+ name + "ливнул с рассказа.");  
 }  
  
 public String captivate() {  
 String artist = "";  
 String wantedAction = "";  
 ArtistTypes ArtistType = ArtistTypes.*PAINTER*;  
 switch (ArtistType) {  
 case *PAINTER*:  
 artist = "художников";  
 wantedAction = "рисовать, рисовать и рисовать";  
 break;  
 case *SCULPTOR*:  
 artist = "скульпторов";  
 wantedAction = "скульптить, скульптить и скульптить";  
 break;  
 case *CARVER*:  
 artist = "резчиков";  
 wantedAction = "резать, резать и резать";  
 break;  
 case *MUSICIAN*:  
 artist = "музыкантов";  
 wantedAction = "исполнять, исполнять и исполнять";  
 break;  
 case *CINEMATOGRAPHER*:  
 artist = "кинематографов";  
 wantedAction = "снимать, снимать и снимать";  
 break;  
 }  
 return "вдохновение '" + name + "' охватывает всех " + artist + ", и тогда им хочется " + wantedAction + ".";  
 }  
 @Override  
 public String getName(){  
 return name;  
 }  
 @Override  
 public void setName(String name){  
 this.name = name;  
 }  
 @Override  
 public String toString(){  
 return "Вдохновение '" + name + "'";  
 }  
 @Override  
 public boolean equals(Object obj){  
 if(this == obj) return true;  
 if(obj instanceof Inspiration){  
 return name.equals(((Inspiration) obj).getName());  
 }  
 return false;  
 }  
 @Override  
 public int hashCode(){  
 return name.hashCode();  
 }  
}

**Run:**

**Story.java:**

package Run;  
  
import characters.\*;  
import exceptions.\*;  
  
public class Story {  
 public static void main(String[] args){  
 Toddler Malysh = new Toddler();  
 System.*out*.println(Malysh.jump("стул") + Malysh.sprint("шкаф") + Malysh.openDoor("шкафа"));  
 System.*out*.println(Malysh.pullOut(false));  
 Carlson Carlsen = new Carlson();  
 Carlsen.scrunch();  
 System.*out*.println(Malysh.give());  
 System.*out*.println(Carlsen.stand(false));  
 Carlsen.give(Malysh);  
 System.*out*.println(Malysh.sew(true));  
 Toddler.Knowledge knowledge = Malysh.new Knowledge("знание");  
 System.*out*.println(knowledge.isTaught(true) + " " + knowledge.magicSew(false));  
 System.*out*.println(knowledge.figOut() + " " + Toddler.*askForHelp*(true, true));  
 System.*out*.println(Carlsen.sew(false) + Carlsen.paint(true));  
 try {  
 System.*out*.println(Malysh.know(false));  
 } catch (ToddlerException te){  
 System.*out*.println("Вы чё упоротый? Как Малыш слышал, тем более знает про это слово?");  
 }  
 try {  
 System.*out*.println(Carlsen.know(true));  
 } catch (ArtistException ae){  
 System.*out*.println("Артист не знает про вдохновения? Не смешите.");  
 }  
 Inspiration inspiration = new Inspiration();  
 Carlsen.explain();  
 System.*out*.println(inspiration.captivate());  
 Toddler.ParallelActions parallelActions = new Toddler.ParallelActions("действия");  
 System.*out*.print(parallelActions.sit(true));  
 parallelActions.continueSew(true);  
 Carlsen.crammed(true);  
 }  
}

**Результат работы:**

****

**Вывод:**

**В этой лабораторной работе я дорабатывал и программу, и свои навыки по использованию SOLID-а в практике. Помимо этого, ещё научился использовать исключения в своих программах (создание классов исключении и их обработка). Это действительно поможет мне в 5ом лабораторном работе. Там будет жестокая заруба.**