

Факультет ПИ и КТ

Лабораторная работа №3 по программированию

Вариант 150150

Выполнил:

Болорболд Аригуун

Группа P3111

Преподаватель:

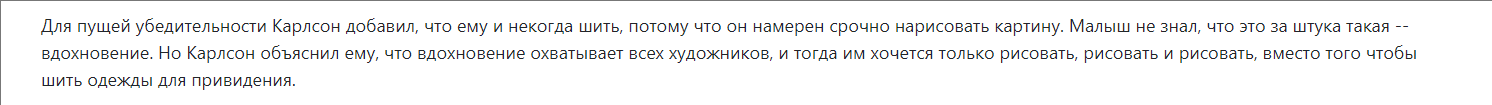
Харитонова Анастасия Евгеньевна

Санкт-Петербург

2023

Текст задания:

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



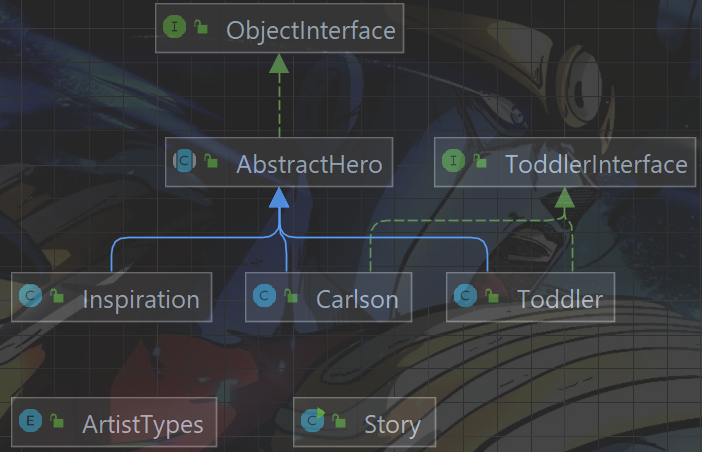
**Программа должна удовлетворять следующим требованиям:**

**1. Доработанная модель должна соответствовать принципам SOLID.**

**2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем). 3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().**

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

**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*}

**Characters:**

1. **Carlson:**

package characters;  
import utilities.\*;  
public class Carlson extends AbstractHero implements ToddlerInterface {  
 private String name;  
 public Carlson() {  
 name = "Карлсон";  
 joinToStory();  
 }  
 public Carlson(String name) {  
 this.name = name;  
 joinToStory();  
 }  
 @Override  
 public void joinToStory() {  
 System.*out*.println("Карлсон '" + name + "' присоединился к рассказу.");  
 }  
 public String sew(boolean hasTime) {  
 String will;  
 if(hasTime){  
 will = "нынче";  
 } else {  
 will = "некогда";  
 }  
 return "Карлсону " + will + " шить одежду";  
 }  
 public String know(boolean doesKnow){  
 if(doesKnow){  
 return "В отличие от него, Карлсон отлично знал, что это за штука, так как желание артиста творить зависит от его вдохновения.";  
 } else {  
 return "Как и малыш, Карлсон к сожалению тоже не знал что это за штука, так как он не являлся настоящим артистом.";  
 }  
  
 }  
 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 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:**

package characters;  
import utilities.\*;  
public class Toddler extends AbstractHero implements ToddlerInterface {  
 private String name;  
 public Toddler(){  
 name = "Малыш";  
 joinToStory();  
 }  
 public Toddler(String name){  
 this.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.");  
 }  
 @Override  
 public String know(boolean doesKnow){  
 if(doesKnow){  
 return "Малыш '" + name + "' знал, что это за штука такая -- вдохновение.";  
 } else {  
 return "Малыш '" + name + "' не знал, что это за штука такая -- вдохновение.";  
 }  
 }  
 @Override  
 public String getName(){  
 return name;  
 }  
 @Override  
 public void setName(String name){  
 this.name = name;  
 }  
 @Override  
 public String toString(){  
 return "Малыш '" + 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:**

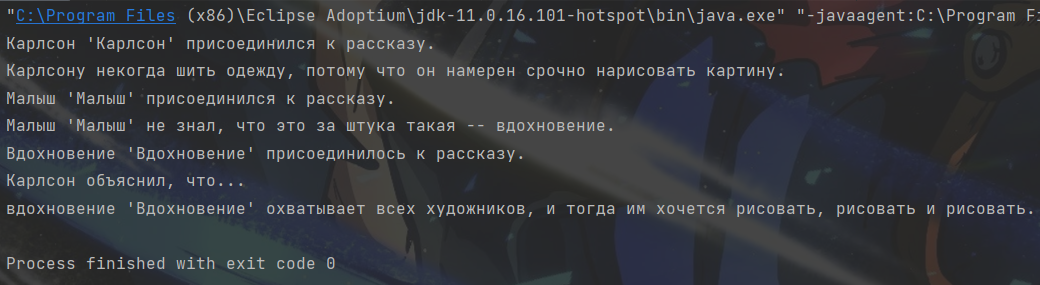
package characters;  
import utilities.\*;  
public class Inspiration extends AbstractHero{  
 private String name;  
 public Inspiration() {  
 name = "Вдохновение";  
 joinToStory();  
 }  
 public Inspiration(String name) {  
 this.name = name;  
 joinToStory();  
 }  
 @Override  
 public void joinToStory() {  
 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.\*;  
public class Story {  
 public static void main(String[] args){  
 Carlson Carlsen = new Carlson();  
 System.*out*.println(Carlsen.sew(false) + Carlsen.paint(true));  
 Toddler Malysh = new Toddler();  
 System.*out*.println(Malysh.know(false));  
 System.*out*.println(Carlsen.know(true));  
 Inspiration inspiration = new Inspiration();  
 Carlsen.explain();  
 System.*out*.println(inspiration.captivate());  
 }  
}

**Результат работы:**



**ГитХаб:** https://github.com/17StarPlatinovich/Proga\_17starplatinum/upload/main/I/I%20semester/Lab3

**Вывод:**

**Я дальше развил свои знания и навыки в области ООП через введения таких новых понятии, как SOLID, интерфейс, перечисляемый тип и уровни абстракции.**

**Мемы ниже**

