Университет ИТМО

Факультет ПИиКТ

Студент Птицын Максим Евгеньевич

Группа P3130

Преподаватель Блохина Елена Николаевна

Лабораторная работа №3.

Вариант № 20279.

г. Санкт-Петербург

2021 г.

**Текст задания:**

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

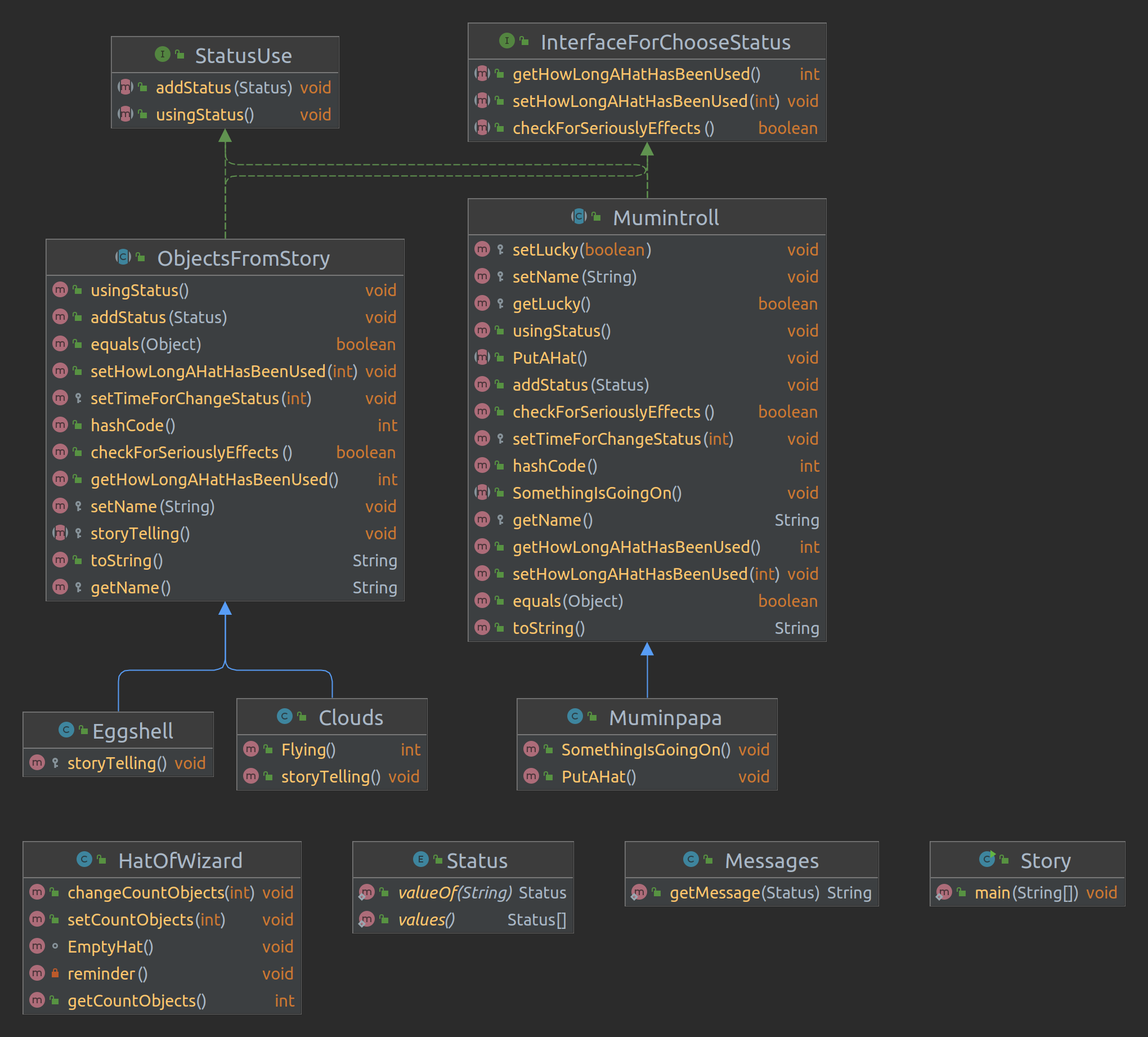
Дело в том, что всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное -- и никогда нельзя знать заранее, во что именно. Муми-папе ужасно повезло, что шляпа ему не подошла: побудь он в ней чуточку подольше -- и только покровителю всех троллей и Сниффов известно, какая участь его ожидала. Муми-папа заработал лишь легкую головную боль (которая прошла после обеда). Зато яичные скорлупки, оставшиеся в шляпе, мало-помалу начали менять свой вид. Они сохранили белый цвет, но все росли и росли в размерах и стали мягкими и пухлыми. Немного погодя они целиком заполнили шляпу, а потом из шляпы выпорхнули пять маленьких круглых тучек. Они выплыли на веранду, мягко спустились с крыльца и повисли в воздухе над самой землей. А в шляпе Волшебника стало пусто.

**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
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).

**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.



Исходный код:

**Story.java**

public class Story {  
 public static void main(String[] args) throws Exception {  
 HatOfWizard hat = new HatOfWizard();  
 Muminpapa papa = new Muminpapa();  
 papa.PutAHat();  
 papa.SomethingIsGoingOn();  
 Eggshell[] eggshells = new Eggshell[5];  
 for (int i = 0; i < eggshells.length; i++) {  
 eggshells[i] = new Eggshell();  
 }  
 hat.changeCountObjects(5);  
 papa.usingStatus();  
 for (int i = 0; i < eggshells.length; i++) {  
 eggshells[i].usingStatus();  
 }  
 hat.EmptyHat();  
 Clouds[] clouds = new Clouds[5];  
 for (int i = 0; i < clouds.length; i++) {  
 clouds[i] = new Clouds();  
 }  
 for (int i = 0; i< clouds.length; i++){  
 hat.changeCountObjects(clouds[i].Flying());  
 }  
 hat.EmptyHat();  
 }  
}

**Status.java**

public enum Status {  
 HEADACHE,  
 SOFT,  
 PLUMP;  
}

**StatusUse.java**

public interface StatusUse {  
 void usingStatus() throws Exception;  
 void addStatus (Status c);  
}

**InterfaceForChooseStatus.java**

public interface InterfaceForChooseStatus {  
 void setHowLongAHatHasBeenUsed(int time);  
 int getHowLongAHatHasBeenUsed();  
 boolean checkForSeriouslyEffects();  
}

**Mumintroll.java**

import java.util.ArrayList;  
  
public abstract class Mumintroll implements InterfaceForChooseStatus, StatusUse {  
 private int HowLongAHatHasBeenOnHead;  
 private String name;  
 private boolean isLucky;  
 private final ArrayList<Status> StatussArrayList = new ArrayList<>();  
 private int TimeForChangeStatus;  
  
 protected Mumintroll(){  
 }  
  
 protected void setLucky(boolean luck){  
 this.isLucky=luck;  
 }  
 protected boolean getLucky(){  
 return this.isLucky;  
 }  
 protected String getName(){  
 return this.name;  
 }  
 protected void setName(String name){  
 this.name=name;  
 }  
 public void addStatus (Status c){  
 this.StatussArrayList.add(c);  
 }  
 public void setHowLongAHatHasBeenUsed(int time){  
 this.HowLongAHatHasBeenOnHead = time;  
 }  
 public int getHowLongAHatHasBeenUsed(){  
 return this.HowLongAHatHasBeenOnHead;  
 }  
 public boolean checkForSeriouslyEffects() {  
 return equals(this);  
 }  
  
 @Override  
 public void usingStatus() throws Exception {  
 if (checkForSeriouslyEffects()) {  
 System.out.println("происходят серьёзные изменения...");  
 } else {  
 addStatus(Status.HEADACHE);  
 System.out.println(getName() + " ощущает " + this.toString() + " .");  
 }  
 Thread.sleep(1000);  
 }

@Override  
 public int hashCode() {  
 return TimeForChangeStatus-HowLongAHatHasBeenOnHead;  
 }  
  
 @Override  
 public boolean equals(Object o){  
 return o.hashCode()<=0;  
 }  
  
 @Override  
 public String toString(){  
 switch (StatussArrayList.size()){  
 case 1:{  
 return Messages.getMessage(StatussArrayList.get(0));  
 }  
 case 2:{  
 return Messages.getMessage(StatussArrayList.get(0))+" и "+Messages.getMessage(StatussArrayList.get(1));  
 }  
 case 3:{  
 return Messages.getMessage(StatussArrayList.get(0))+", "+Messages.getMessage(StatussArrayList.get(1))+" и "+Messages.getMessage(StatussArrayList.get(2));  
 }  
 default:{  
 return "здоров";  
 }  
 }  
 }  
  
 protected void setTimeForChangeStatus(int i){  
 this.TimeForChangeStatus=i;  
 }  
  
 public abstract void PutAHat() throws Exception;  
 public abstract void SomethingIsGoingOn() throws Exception;  
}

**ObjectsFromStory.java**

import java.util.ArrayList;  
  
public abstract class ObjectsFromStory implements InterfaceForChooseStatus, StatusUse {  
  
 private String name;  
 private String color;  
 private ArrayList<Status> StatussArrayList = new ArrayList<>();  
 private int HowLongAHatHasBeenOnHead;  
 private int TimeForChangeStatus;  
  
 protected ObjectsFromStory(){  
 name = "Яичная скорлупка";  
 color = "Белый";  
 }  
  
 protected String getName(){  
 return this.name;  
 }  
 protected void setName(String name){  
 this.name = name;  
 }  
  
 public void addStatus (Status c){  
 this.StatussArrayList.add(c);  
 }  
 public void setHowLongAHatHasBeenUsed(int time){  
 this.HowLongAHatHasBeenOnHead = time;  
 }  
 public int getHowLongAHatHasBeenUsed(){  
 return this.HowLongAHatHasBeenOnHead;  
 }  
 public boolean checkForSeriouslyEffects() {  
 return equals(this);  
 }  
  
 protected void setTimeForChangeStatus(int i){  
 this.TimeForChangeStatus=i;  
 }  
  
 @Override  
 public void usingStatus() throws Exception {  
 if (checkForSeriouslyEffects()) {  
 addStatus(Status.SOFT);  
 addStatus(Status.PLUMP);  
 System.out.println(getName() + " становится " + this.toString()+" ...");  
 Thread.sleep(1000);  
 System.out.println(getName() + " увеличивается в размерах и превращается в тучку.");  
 } else {  
 System.out.println("Ничего не происходит...");  
 }  
 }  
  
 @Override  
 public int hashCode() {  
 return TimeForChangeStatus-HowLongAHatHasBeenOnHead;  
 }

@Override  
 public boolean equals(Object o){  
 return o.hashCode()<=0;  
 }  
  
 @Override  
 public String toString(){  
 switch (StatussArrayList.size()){  
 case 1:{  
 return Messages.getMessage(StatussArrayList.get(0));  
 }  
 case 2:{  
 return Messages.getMessage(StatussArrayList.get(0))+" и "+Messages.getMessage(StatussArrayList.get(1));  
 }  
 case 3:{  
 return Messages.getMessage(StatussArrayList.get(0))+", "+Messages.getMessage(StatussArrayList.get(1))+" и "+Messages.getMessage(StatussArrayList.get(2));  
 }  
 default:{  
 return "здоров";  
 }  
 }  
 }  
 protected abstract void storyTelling() throws Exception;  
}

**Muminpapa.java**

public class Muminpapa extends Mumintroll {  
  
  
 public Muminpapa(){  
 setName("Мумми-папа");  
 setLucky(true);  
 setTimeForChangeStatus(3000);  
 }  
  
 public void PutAHat() throws Exception {  
 System.out.println(getName() + " надевает шляпу Волшебника.");  
 if (getLucky()) {  
 System.out.print("Шляпа ему не подошла. ");  
 System.out.println(getName() + " носит шляпу непродолжительное время. ");  
 setHowLongAHatHasBeenUsed(1000);  
 Thread.sleep(getHowLongAHatHasBeenUsed());  
 }  
 }  
  
 @Override  
 public void SomethingIsGoingOn() throws Exception {  
 System.out.println("Проходит какое-то время, Мумми-папа выбрасывает скорлупки в шляпу...");  
 Thread.sleep(1000);  
 }  
}

**Messages.java**

public class Messages {  
  
 public Messages() {}  
  
 public static String getMessage(Status c){  
 switch (c) {  
 case SOFT: {  
 return "мягкая";  
 }  
 case PLUMP: {  
 return "пухлая";  
 }  
 case HEADACHE: {  
 return "головную боль";  
 }  
 }  
 return "ничего";  
 }  
}

**HatOfWizard.java**

public class HatOfWizard {  
  
 private int CountObjects;  
  
 public HatOfWizard() {  
 setCountObjects(0);  
 reminder();  
 }  
  
 public void setCountObjects(int countObjects) {  
 this.CountObjects = countObjects;  
 }  
  
 public int getCountObjects() {  
 return CountObjects;  
 }  
  
 public void changeCountObjects(int i) {  
 this.CountObjects = CountObjects + i;  
 }  
  
 private void reminder() {  
 System.out.println("Всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное - и никогда нельзя знать заранее, во что именно.");  
 }  
  
 void EmptyHat() {  
 if (getCountObjects() == 0) {  
 System.out.println("Шляпа пуста.");  
 } else {  
 System.out.println("Шляпа заполнена.");  
 }  
 }  
}

**Eggshell.java**

public class Eggshell extends ObjectsFromStory{  
  
 public Eggshell() throws Exception {  
 setTimeForChangeStatus(5000);  
 storyTelling();  
 }  
  
 protected void storyTelling() throws Exception{  
 System.out.println(getName() + " лежит в шляпе...");  
 Thread.sleep(1000);  
 setHowLongAHatHasBeenUsed(10000);  
 }  
}

**Clouds.java**

public class Clouds extends ObjectsFromStory {  
 public Clouds() throws Exception {  
 setName("Тучка");  
 addStatus(Status.PLUMP);  
 addStatus(Status.SOFT);  
 storyTelling();  
 }  
  
 @Override  
 public void storyTelling() throws Exception{  
 System.out.println(getName() + " взлетает из шляпы и мягко парит в воздухе.");  
 Thread.sleep(1000);  
 }  
  
 public int Flying() throws Exception{  
 System.out.println(getName() + " выплывает на веранду и повисает над самой землёй.");  
 Thread.sleep(1000);  
 return -1;  
 }  
}

Результат работы программы:

Всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное - и никогда нельзя знать заранее, во что именно.

Мумми-папа надевает шляпу Волшебника.

Шляпа ему не подошла. Мумми-папа носит шляпу непродолжительное время.

Проходит какое-то время, Мумми-папа выбрасывает скорлупки в шляпу...

Яичная скорлупка лежит в шляпе...

Яичная скорлупка лежит в шляпе...

Яичная скорлупка лежит в шляпе...

Яичная скорлупка лежит в шляпе...

Яичная скорлупка лежит в шляпе...

Мумми-папа ощущает головную боль .

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Шляпа заполнена.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка выплывает на веранду и повисает над самой землёй.

Тучка выплывает на веранду и повисает над самой землёй.

Тучка выплывает на веранду и повисает над самой землёй.

Тучка выплывает на веранду и повисает над самой землёй.

Тучка выплывает на веранду и повисает над самой землёй.

Шляпа пуста.

Выводы по работе:

Научился описывать объектную модель приложения, используя принципы SOLID, переопределять toString(), hashCode(), equals().