**Отчёт выполнения задания по теме**

**«Понятия ООП: абстракция, инкапсуляция»**

Создан репозиторий <https://github.com/may17s/homeworks.git>.

Создаем ветку для добавления проекта

# git branch -b homework06



# git add .



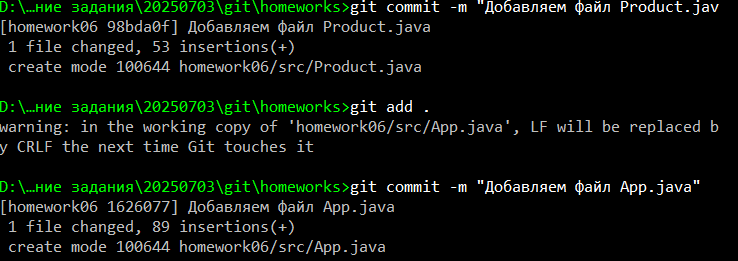
# git commit -m " Создаём проект с файлом Person.java"

# git add .

# git commit -m " Добавляем Product.java"

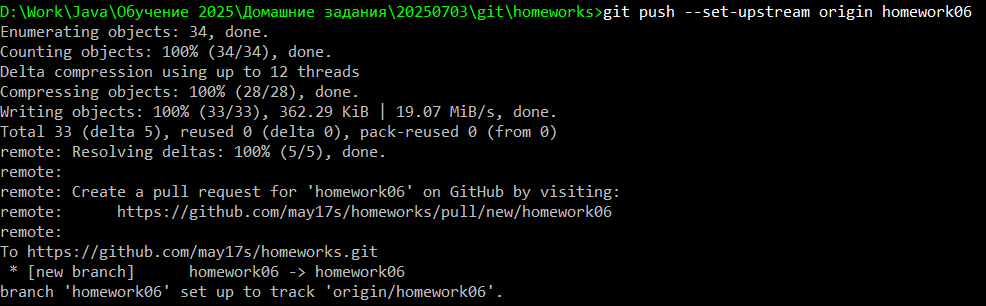
# git add .

# git commit -m " Добавляем файл App.java"



Заливаем всё на github в ветку homework06

# git push --set-upstream origin homework06



Дополнительное задание

Редактируем код и выполняем:

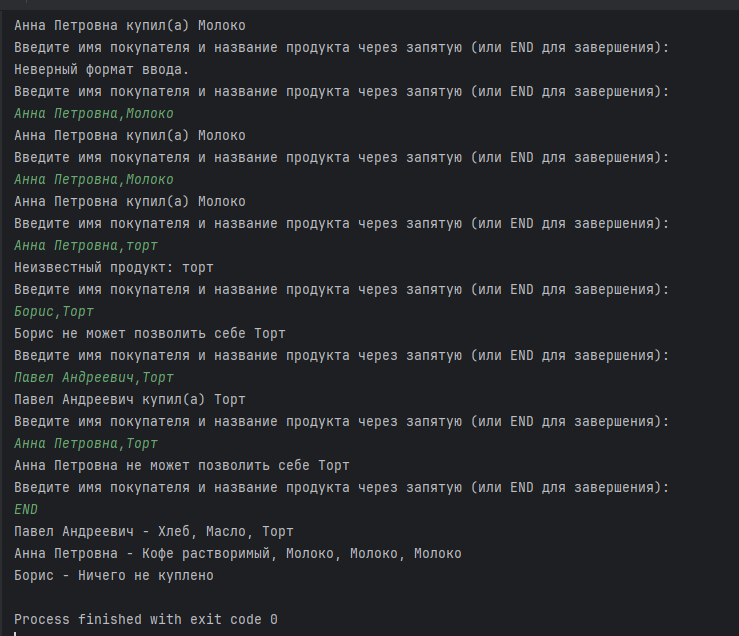
# git add .

# git commit -m "Добавляем файл TelevisionSet.java, Program.java, Channel.java и редактируем App.java"

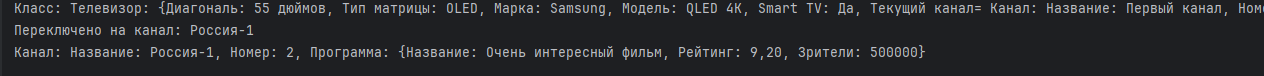
Заливаем всё на github в ветку homework06

# git push --set-upstream origin homework06

Пример данных о покупках



Вывод дополнительного задания о каналах и программах класса Телевизор



import java.util.Objects;  
  
public class Product {  
 private String name;  
 private double cost;  
  
 public Product(String name, double cost) {  
 this.setName(name);  
 this.setCost(cost);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getCost() {  
 return cost;  
 }  
  
 public void setName(String name) {  
 if (name == null || name.isEmpty()) {  
 throw new IllegalArgumentException("Название продукта не может быть пустым");  
 }  
 this.name = name;  
 }  
  
 public void setCost(double cost) {  
 if (cost < 0) {  
 throw new IllegalArgumentException("Стоимость продукта не может быть отрицательной");  
 }  
 this.cost = cost;  
 }  
  
 @Override  
 public String toString() {  
 return name + ", " + cost + "руб.";  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null || getClass() != object.getClass()) return false;  
  
 Product product = (Product) object;  
 return Double.*compare*(product.cost, cost) == 0  
 && Objects.*equals*(name, product.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, cost);  
 }  
}

import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Person {  
 private String name;  
 private double money;  
 private List<Product> bag;  
  
 public Person(String name, double money) {  
 this.setName(name);  
 this.setMoney(money);  
 this.bag = new ArrayList<>();  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 if (name == null || name.isEmpty()) {  
 throw new IllegalArgumentException("Имя покупателя не может быть пустым");  
 }  
 this.name = name;  
 }  
  
 public double getMoney() {  
 return money;  
 }  
  
 public void setMoney(double money) {  
 if (money < 0) {  
 throw new IllegalArgumentException("Деньги не могут быть отрицательными");  
 }  
 this.money = money;  
 }  
  
 public List<Product> getBag() {  
 return new ArrayList<>(bag);  
 }  
  
 public void buy(Product product) {  
 if (product == null) {  
 throw new IllegalArgumentException("Продукт не может быть пустым");  
 }  
 if (this.getMoney() >= product.getCost()) {  
 this.setMoney(this.getMoney() - product.getCost());  
 this.bag.add(product);  
 System.*out*.println(this.name + " купил(а) " + product.getName());  
 } else {  
 System.*out*.println(this.name + " не может позволить себе " + product.getName());  
 }  
 }  
  
 @Override  
 public String toString() {  
 if (bag.isEmpty())  
 return name + " - Ничего не куплено";  
  
 StringBuilder sb = new StringBuilder();  
 sb.append(name);  
 sb.append(" - ");  
 for (Product product : bag) {  
 sb.append(product.getName());  
 sb.append(", ");  
 }  
 sb.setLength(sb.length() - ", ".length());  
 return sb.toString();  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null || getClass() != object.getClass()) return false;  
  
 Person person = (Person) object;  
 return Double.*compare*(person.money, money) == 0  
 && Objects.*equals*(name, person.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, money);  
 }  
}

import java.util.Arrays;  
import java.util.Scanner;  
  
public class App {  
 public static void main(String[] args) {  
 // Тестовые данные №1  
 Person[] persons = new Person[] {  
 new Person("Павел Андреевич", 10000),  
 new Person("Анна Петровна", 2000),  
 new Person("Борис", 10)  
 };  
  
 // Тестовые данные №2  
 /\*  
 Person[] persons = new Person[] {  
 new Person("Женя", 0)  
 };  
 \*/  
 // Тестовые данные №3  
 /\*  
 Person[] persons = new Person[] {  
 new Person("Света", -3)  
 };  
 \*/  
  
 Product[] products = new Product[] {  
 new Product("Хлеб", 40),  
 new Product("Молоко", 60),  
 new Product("Торт", 1000),  
 new Product("Кофе растворимый", 879),  
 new Product("Масло", 150),  
 new Product("Мороженое", 200),  
 new Product("Макароны", 800)  
 };  
  
 try (Scanner scanner = new Scanner(System.*in*)) {  
 while (true) {  
 System.*out*.println("Введите имя покупателя и название продукта через запятую (или END для завершения):");  
 String input = scanner.nextLine();  
 if (input.equalsIgnoreCase("END")) {  
 break;  
 }  
  
 String[] parts = input.split(",", 2);  
 if (parts.length != 2) {  
 System.*out*.println("Неверный формат ввода.");  
 continue;  
 }  
  
 String personName = parts[0];  
 String productName = parts[1];  
  
 int index = -1;  
 for (int i = 0; i < persons.length; i++) {  
 if (persons[i].getName().equals(personName)) {  
 index = i;  
 break;  
 }  
 }  
  
 if(index == -1) {  
 System.*out*.println("Неизвестный покупатель: " + personName);  
 continue;  
 }  
  
 Person person = persons[index];  
  
 index = -1;  
 for (int i = 0; i < products.length; i++) {  
 if (products[i].getName().equals(productName)) {  
 index = i;  
 break;  
 }  
 }  
  
 if(index == -1) {  
 System.*out*.println("Неизвестный продукт: " + productName);  
 continue;  
 }  
  
 person.buy(products[index]);  
 }  
  
 for(Person person: persons) {  
 System.*out*.println(person);  
 }  
 }  
  
 // Создаём сущности  
 Program newsProgram = new Program("Новости", 8.5, 1000000);  
 Program movieProgram = new Program("Очень интересный фильм", 9.2, 500000);  
  
 Channel channel1 = new Channel("Первый канал", 1, newsProgram);  
 Channel channel2 = new Channel("Россия-1", 2, movieProgram);  
  
 Channel[] channels = {channel1, channel2};  
 TelevisionSet tv = new TelevisionSet(55, TelevisionSet.DisplayTechnology.*OLED*, "Samsung", "QLED 4K", true, channels);  
  
 tv.setTvSwitch(); // Включим телевизор  
 System.*out*.println(tv); // Вывод информации о телевизоре  
  
 tv.changeChannel(2); // Переключимся на второй канал  
 System.*out*.println(tv.getCurrentChannel()); // Вывод текущего канала  
 tv.setTvSwitch(); // Выключим телевизор  
 }  
}

import java.util.Objects;  
  
class Program {  
 private String name;  
 private double rating;  
 private int viewers;  
  
 public Program(String name, double rating, int viewers) {  
 this.name = name;  
 this.rating = rating;  
 this.viewers = viewers;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getRating() {  
 return rating;  
 }  
  
 public void setRating(double rating) {  
 if (rating >= 0 && rating <= 10) {  
 this.rating = rating;  
 } else {  
 System.*out*.println("Рейтинг должен быть в диапазоне от 0 до 10.");  
 }  
 }  
  
 public int getViewers() {  
 return viewers;  
 }  
  
 public void setViewers(int viewers) {  
 if (viewers >= 0) {  
 this.viewers = viewers;  
 } else {  
 System.*out*.println("Число зрителей не может быть отрицательным.");  
 }  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Program program = (Program) o;  
 return Double.*compare*(program.rating, rating) == 0 &&  
 viewers == program.viewers &&  
 Objects.*equals*(name, program.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, rating, viewers);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Программа: {Название: %s, Рейтинг: %.2f, Зрители: %d}", name, rating, viewers);  
 }  
}

import java.util.Objects;  
  
class Channel {  
 private String name;  
 private int number;  
 private Program prg;  
  
 public Channel(String name, int number, Program prg) {  
 this.name = name;  
 this.number = number;  
 this.prg = prg;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void setNumber(int number) {  
 this.number = number;  
 }  
  
 public Program getProgram() {  
 return prg;  
 }  
  
 public void setProgram(Program program) {  
 this.prg = program;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null || getClass() != object.getClass()) return false;  
  
 Channel channel = (Channel) object;  
 return number == channel.number &&  
 Objects.*equals*(name, channel.name) &&  
 Objects.*equals*(prg, channel.prg);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, number, prg);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Канал: Название: %s, Номер: %d, %s", name, number, prg);  
 }  
}

import java.util.Arrays;  
import java.util.Objects;  
  
public class TelevisionSet {  
  
 public enum DisplayTechnology { *NONE*, *CRT*, *PLASMA*, *LED*, *OLED*, *QLED* };  
  
 private int screenSize; // Размер экрана (дюймы)  
 private DisplayTechnology displayTechnology; // Тип матрицы  
 private String brandName; // Марка  
 private String modelName; // Модель  
 private boolean hasSmartTV; // Имеет ли Smart TV  
 private int currentChannelIndex; // текущий канал  
 private Channel[] channels; // список каналов  
 private int volume; // Громкость (0..100)  
 private boolean tvSwitch; // Признак включенного телевизора  
  
 public TelevisionSet() {  
 this.screenSize = 0;  
 this.displayTechnology = DisplayTechnology.*NONE*;  
 this.brandName = "";  
 this.modelName = "";  
 this.hasSmartTV = false;  
 this.currentChannelIndex = 0;  
 this.volume = 25;  
 this.tvSwitch = false;  
 }  
  
 public TelevisionSet(int screenSize, DisplayTechnology displayTechnology, String brandName, String modelName, boolean hasSmartTV, Channel[] channels) {  
 this.screenSize = screenSize;  
 this.displayTechnology = displayTechnology;  
 this.brandName = brandName;  
 this.modelName = modelName;  
 this.hasSmartTV = hasSmartTV;  
 this.currentChannelIndex = 0;  
 this.volume = 25;  
 this.tvSwitch = false;  
 this.channels = channels;  
 }  
  
 public int getScreenSize() {  
 return this.screenSize;  
 }  
  
 public void setScreenSize(int screenSize) {  
 if (screenSize > 0) {  
 this.screenSize = screenSize;  
 } else {  
 System.*out*.println("Размер экрана указан не верно! Должен быть больше нуля.");  
 }  
 }  
  
 public DisplayTechnology getDisplayTechnology() {  
 return displayTechnology;  
 }  
  
 public void setDisplayTechnology(DisplayTechnology displayTechnology) { this.displayTechnology = displayTechnology; }  
  
 public String getBrandName() {  
 return brandName;  
 }  
  
 public void setBrandName(String brandName) {  
 this.brandName = brandName;  
 }  
  
 public String getModelName() {  
 return modelName;  
 }  
  
 public void setModelName(String modelName) {  
 this.modelName = modelName;  
 }  
  
 public void setHasSmartTV(boolean hasSmartTV) {  
 this.hasSmartTV = hasSmartTV;  
 }  
  
 public boolean isHasSmartTV() {  
 return hasSmartTV;  
 }  
  
  
 public int getVolume() { return volume; }  
  
 public void setVolume(int volume) {  
 if (volume >= 0 && volume <= 100) {  
 this.volume = volume;  
 } else {  
 System.*out*.println("Громкость должна быть в диапазоне от 0 до 100 (по умолчанию 25)");  
 }  
 }  
  
 public boolean getTvSwitch() { return tvSwitch; }  
  
 public void setTvSwitch() { this.tvSwitch = !this.tvSwitch; }  
  
 public void changeChannel(int channelNumber) {  
 if (!tvSwitch) {  
 System.*out*.println("Телевизор выключен. Включение...");  
 setTvSwitch();  
 }  
 for (int i = 0; i < channels.length; i++) {  
 if (channels[i].getNumber() == channelNumber) {  
 currentChannelIndex = i;  
 System.*out*.printf("Переключено на канал: %s%n", channels[i].getName());  
 return;  
 }  
 }  
 System.*out*.println("Канал не найден.");  
 }  
  
 public void setChannels(Channel[] channels) {  
 this.channels = channels;  
 }  
  
 public Channel getCurrentChannel() {  
 if (channels.length > 0 && currentChannelIndex < channels.length) {  
 return channels[currentChannelIndex];  
 }  
 return null;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null || getClass() != object.getClass()) return false;  
  
 TelevisionSet televisionSet = (TelevisionSet) object;  
 return screenSize == televisionSet.screenSize  
 && hasSmartTV == televisionSet.hasSmartTV  
 && currentChannelIndex == televisionSet.currentChannelIndex  
 && volume == televisionSet.volume  
 && tvSwitch == televisionSet.tvSwitch  
 && Objects.*equals*(brandName, televisionSet.brandName)  
 && Objects.*equals*(modelName, televisionSet.modelName)  
 && Objects.*deepEquals*(channels, televisionSet.channels);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(brandName, modelName, screenSize, hasSmartTV, currentChannelIndex, volume, tvSwitch, Arrays.*deepHashCode*(channels));  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Класс: Телевизор: " +  
 "{Диагональ: %d дюймов, Тип матрицы: %s, Марка: %s, Модель: %s, Smart TV: %s, Текущий канал= %s, Громкость=%d, Включен=%s}",  
 screenSize, displayTechnology.toString(), brandName, modelName, (hasSmartTV ? "Да" : "Нет"), getCurrentChannel(), volume, (tvSwitch ? "Да" : "Нет"));  
 }  
}