**Домашнее задание по теме**

**«Система контроля версий Git. Работа с локальным репозиторием.**

**Работа с удаленным репозиторием»**

Формулировка задания:

Установить и настроить систему контроля версий GIT.

1. Настроить Git на локальном компьютере (если не был настроен

ранее). По желанию можно также установить Tortoise Git на Windows;

2. Создать удаленный репозиторий в GitHub (если не был создан ранее).

Репозиторию дать права доступа public на время работы с курсом;

3. Создать локальную ветку для выгрузки заданий. В созданную ветку

выгрузить задачи:

a. Папка homework01. Содержимое: Первая программа на Java

(тема «Подготовка рабочего места. Среда разработки для Java»);

b. Папка homework02. Содержимое: Базовый синтаксис Java

(тема «Точка входа в программу. JVM структура»);

c. Папка homework03. Содержимое: Работа с классами (темы

«Классы: поля, свойства, методы» и «Класс Object. Класс String»);

d. Папка homework04. Содержимое: Класс String (темы

«Классы: поля, свойства, методы» и «Класс Object. Класс String»).

Ветка создается по шаблону:

homeworks/homework%number%

Пример для домашнего задания:

homeworks/homework05

4. Отправить локальную ветку на удаленный репозиторий.

5. Создать pull request в ветку main из ветки с домашним заданием в

удаленном репозитории. Назначить проверяющим наставника группы.

6. Дополнительно. Добавить в проект решение ещё одной задачи. Для

этого:

a. Создать ветку homeworks/homework05Addition.

b. В ветке решить дополнительное задание, представленное

ниже. Задание решить в папке homework05.

c. Отправить локальную ветку из п.6 на удаленный

репозиторий.

d. Создать pull request в ветку main из

homeworks/homework05Addition. Назначить проверяющим наставника группы.

Дополнительная задача:

Доработать класс Телевизор:

1. В класс Телевизор добавить поля (если не были добавлены ранее):

1) Номер включенного канала — integer;

2) Громкость звука — integer (от 0 до 100);

3) Признак включенного телевизора — boolean.

2. Переопределить метод toString класса Телевизор таким образом, чтобы

распечатывались: название класса, все поля класса и их значения.

3. Добавить в класс Телевизор методы equals и hashcode.

На вход программы в классе App, методе main подается информация о

телевизорах в количестве 10 телевизоров. Считать данные в массив объектов

(можно выполнить считывание данных в цикле). В каждом из 10 экземпляров

класса должны быть заполнены следующие поля:

a. Как минимум, 1 поле по выбору, добавленное студентом в задании 3;

b. Номер включенного канала - целое число,

c. Громкость звука - целое число;

d. Признак включен ли телевизор.

Считать с клавиатуры число допустимого значения громкости звука

maxVolume (рекомендуется ввести с клавиатуры целое число от 50 до 70).

Далее необходимо в цикле вывести только включенные телевизоры, у

которых звук является допустимым (меньшим или равным maxVolume).

Дополнительно. Добавить в массив Телевизоров сортировку по номеру

канала (по возрастанию).

Планируемый результат:

1. Ссылка на программу и отчёт со скриншотами в репозитории github;

2. Архив с программой и отчётом со скриншотами выполнения задач.

В отчёте представить следующие скриншоты:

- установка git (если гит устанавливался на систему),

- создание ветки git,

- структура программы по папкам,

- выгрузка программы в удаленный репозиторий,

- код дополнительного задания и скриншот выполнения кода (в случае,

если задание выполнялось).

Описания плана работы:

Выполнение задания в соответствии с формулировкой требований к

задаче.

Перечень инструментов, необходимых для реализации деятельности:

Персональный компьютер, JDK 17 (либо OpenJDK 17), Intellij Idea для

разработки на Java, GIT, возможны дополнительные инструменты (например,

Tortoise GIT, GitHub Desktop).

**Отчёт выполнения задания**

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

клонируем репозиторий командой:

# git clone <https://github.com/may17s/homeworks.git>

Создаем ветку для добавления проектов

# git branch -b homework05

Добавляем всё

# git add .

Коммитим

# git commit -m "add homeworks"

Создаем токен для отправки на удаленный сервер и указываем его

# git remote add origin <https://may17s:xxxxxxxxxxxxxxxxxxxxxxx@github.com/may17s/homeworks.git>

Заливаем всё на github

# git push --set-upstream origin main

Создаем ветку для pull request

# git checkout -b homework05Addition

Редактируем код и добавляем всё

# git add .

Коммитим

# git commit -m "pull request homework5Addition"

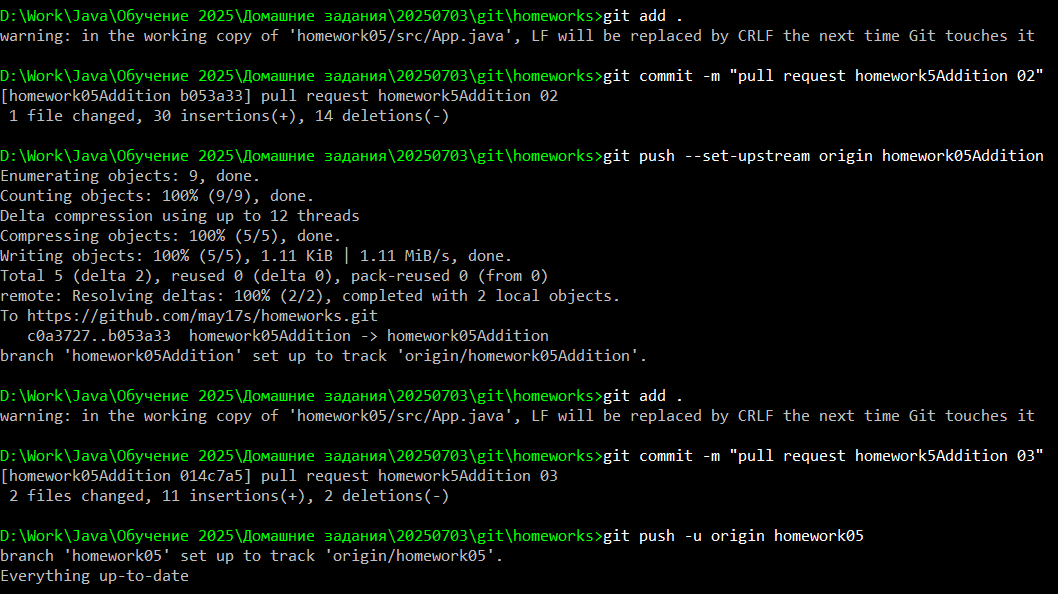
Создаем pull request

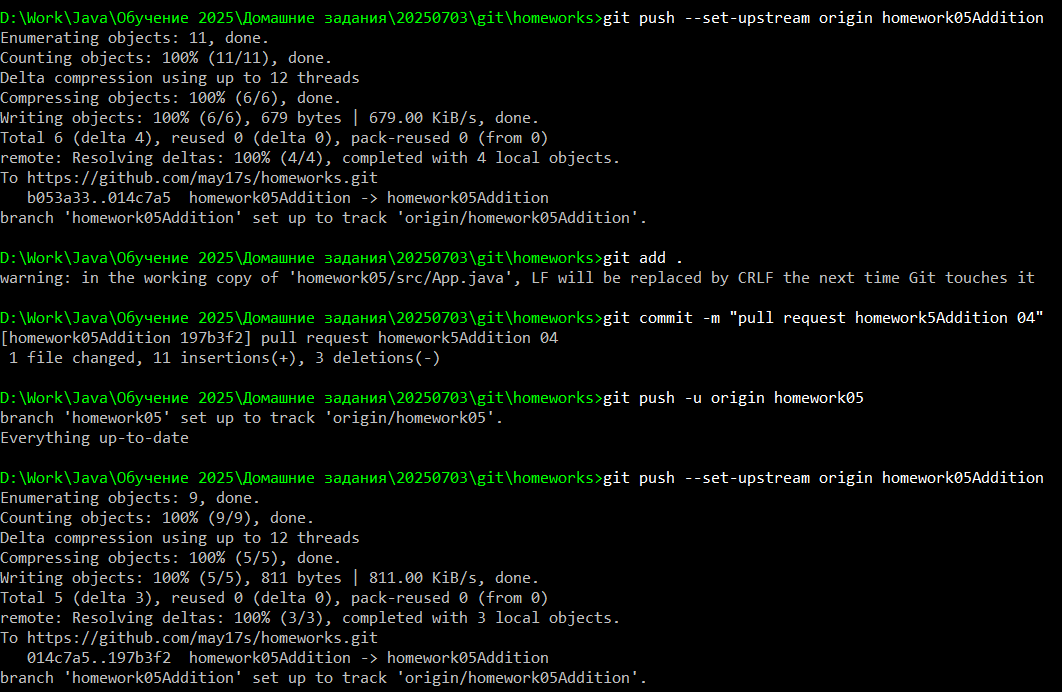
# git push -u origin homework05

Заливаем всё на github

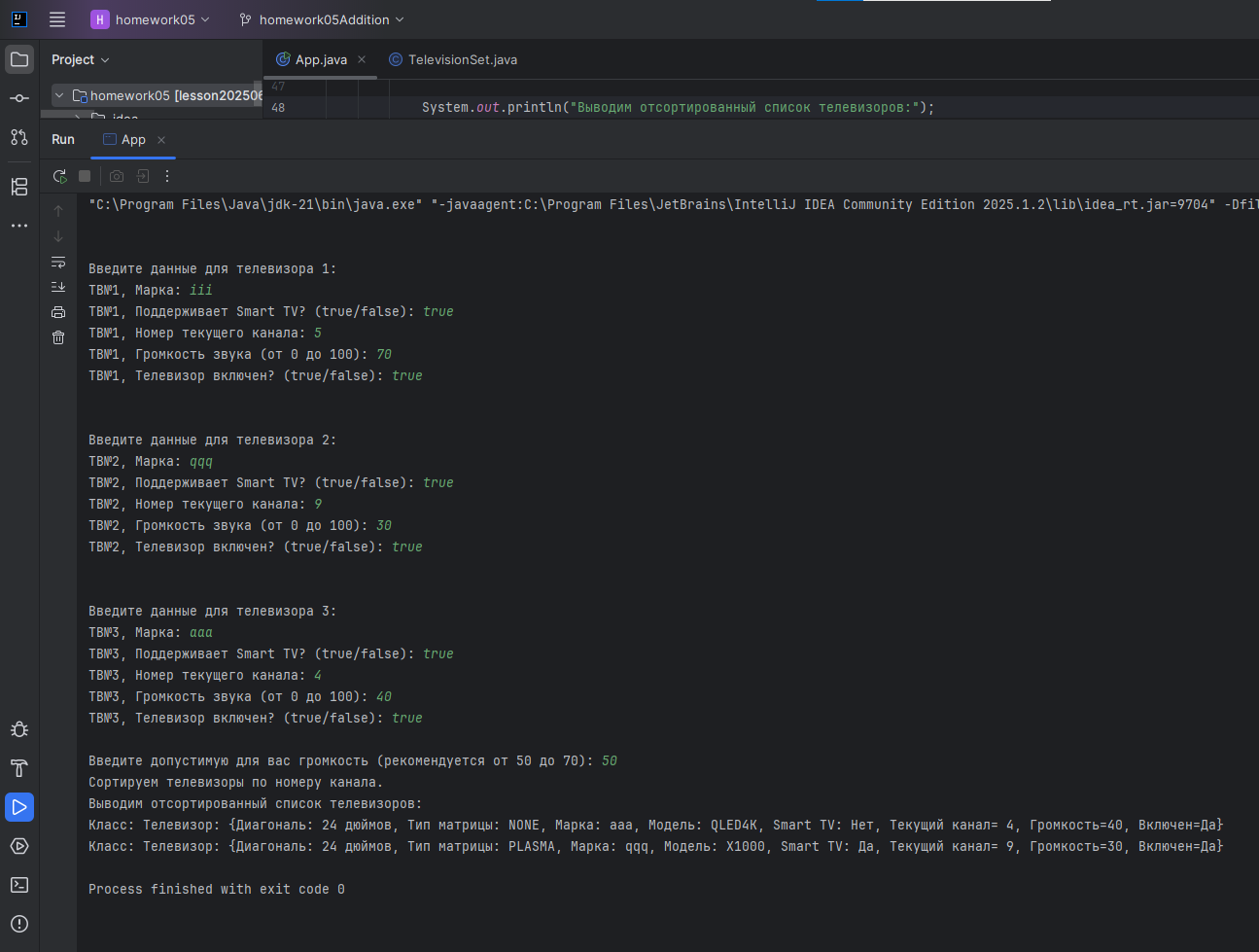
# git push --set-upstream origin homework05Addition

Повторяем несколько раз после доработки дополнительных задач

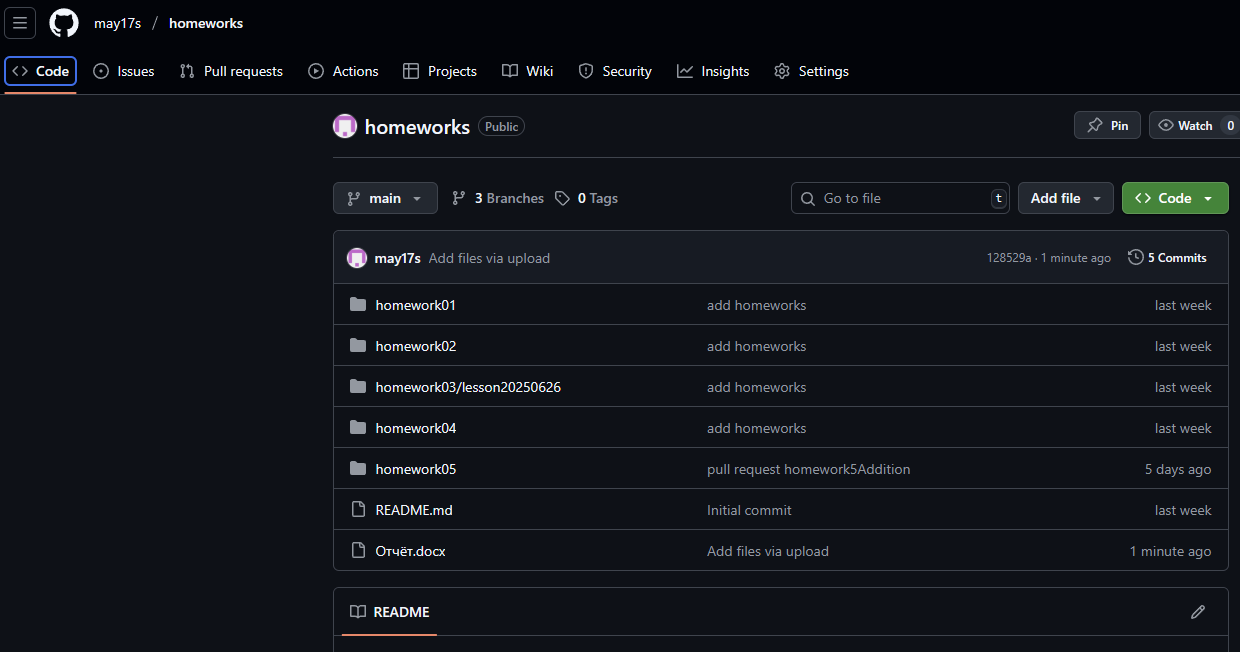




Выгрузка программы в удаленный репозиторий



Результат работы дополнительных заданий



Структура папок в github.

Код дополнительного задания

App.java

import java.util.Arrays;  
import java.util.Random;  
import java.util.Scanner;  
  
public class App {  
 public static void main(String[] args) {  
  
 try (Scanner scanner = new Scanner(System.*in*)) {  
 TelevisionSet[] tvsetArray = new TelevisionSet[3];  
 Random random = new Random();  
 int[] screenSizes = {24, 25, 27, 32};  
 String[] models = {"X1000", "UltraHD", "OLED55", "QLED4K", "SmartView"};  
  
 for (int i = 0; i < tvsetArray.length; i++) {  
 System.*out*.println("\n");  
 System.*out*.println("Введите данные для телевизора " + (i + 1) + ":");  
 System.*out*.print("ТВ№" + (i + 1) + ", Марка: ");  
 String brandName = scanner.nextLine();  
 System.*out*.print("ТВ№" + (i + 1) + ", Поддерживает Smart TV? (true/false): ");  
 boolean hasSmartTV = scanner.nextBoolean();  
 scanner.nextLine(); // чистим буфер  
 System.*out*.print("ТВ№" + (i + 1) + ", Номер текущего канала: ");  
 int currentChannel = scanner.nextInt();  
 scanner.nextLine(); // чистим буфер  
 System.*out*.print("ТВ№" + (i + 1) + ", Громкость звука (от 0 до 100): ");  
 int volume = scanner.nextInt();  
 scanner.nextLine(); // чистим буфер  
 System.*out*.print("ТВ№" + (i + 1) + ", Телевизор включен? (true/false): ");  
 boolean tvSwitch = scanner.nextBoolean();  
 scanner.nextLine(); // чистим буфер  
  
 TelevisionSet tvSet = new TelevisionSet(  
 screenSizes[random.nextInt(screenSizes.length)],  
 TelevisionSet.DisplayTechnology.*values*()[random.nextInt(TelevisionSet.DisplayTechnology.*values*().length)],  
 brandName, models[random.nextInt(models.length)], random.nextBoolean());  
 tvSet.setCurrentChannel(currentChannel);  
 tvSet.setVolume(volume);  
 if (tvSwitch) tvSet.setTvSwitch();  
 tvsetArray[i] = tvSet;  
 }  
  
 System.*out*.print("\nВведите допустимую для вас громкость (рекомендуется от 50 до 70): ");  
 int maxVolume = scanner.nextInt();  
  
 System.*out*.println("Сортируем телевизоры по номеру канала.");  
 Arrays.*sort*(tvsetArray, (tv1, tv2) -> Integer.*compare*(tv1.getCurrentChannel(), tv2.getCurrentChannel()));  
  
 System.*out*.println("Выводим отсортированный список телевизоров:");  
 for(TelevisionSet tvset:tvsetArray) {  
 if(tvset.getTvSwitch() && tvset.getVolume() <= maxVolume) {  
 System.*out*.println(tvset);  
 }  
 }  
 }  
 }  
}

TelevisionSet.java

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 currentChannel; // текущий канал  
 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.currentChannel = 1;  
 this.volume = 25;  
 this.tvSwitch = false;  
 }  
  
 public TelevisionSet(int screenSize, DisplayTechnology displayTechnology, String brandName, String modelName, boolean hasSmartTV) {  
 this.screenSize = screenSize;  
 this.displayTechnology = displayTechnology;  
 this.brandName = brandName;  
 this.modelName = modelName;  
 this.hasSmartTV = hasSmartTV;  
 this.currentChannel = 1;  
 this.volume = 25;  
 this.tvSwitch = false;  
 }  
  
 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 getCurrentChannel() { return currentChannel; }  
  
 public void setCurrentChannel(int currentChannel) {  
 this.currentChannel = currentChannel;  
 }  
  
 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 = true; }  
  
 @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  
 && currentChannel == televisionSet.currentChannel  
 && volume == televisionSet.volume  
 && tvSwitch == televisionSet.tvSwitch  
 && Objects.*equals*(brandName, televisionSet.brandName)  
 && Objects.*equals*(modelName, televisionSet.modelName);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(brandName, modelName, screenSize, hasSmartTV, currentChannel, volume, tvSwitch);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Класс: Телевизор: " +  
 "{Диагональ: %d дюймов, Тип матрицы: %s, Марка: %s, Модель: %s, Smart TV: %s, Текущий канал= %d, Громкость=%d, Включен=%s}",  
 screenSize, displayTechnology.toString(), brandName, modelName, (hasSmartTV ? "Да" : "Нет"), currentChannel, volume, (tvSwitch ? "Да" : "Нет"));  
 }  
}