Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ**

по дисциплине «Системный подход в разработке программного обеспечения»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

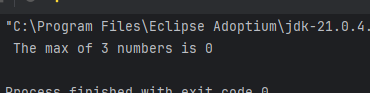
Санкт-Петербург

2024

1. **По занятию 8.4**
2. **Упражнение 1**

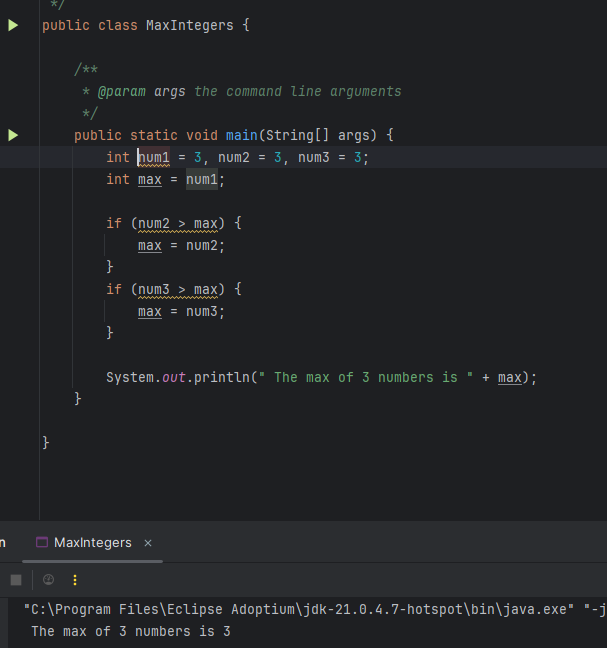
Импортируйте и откройте проект DebuggingEx.

* Просмотрите MaxIntegers.java.
* Можете ли вы определить, что Ричи пропустил во время своей проверки?

****

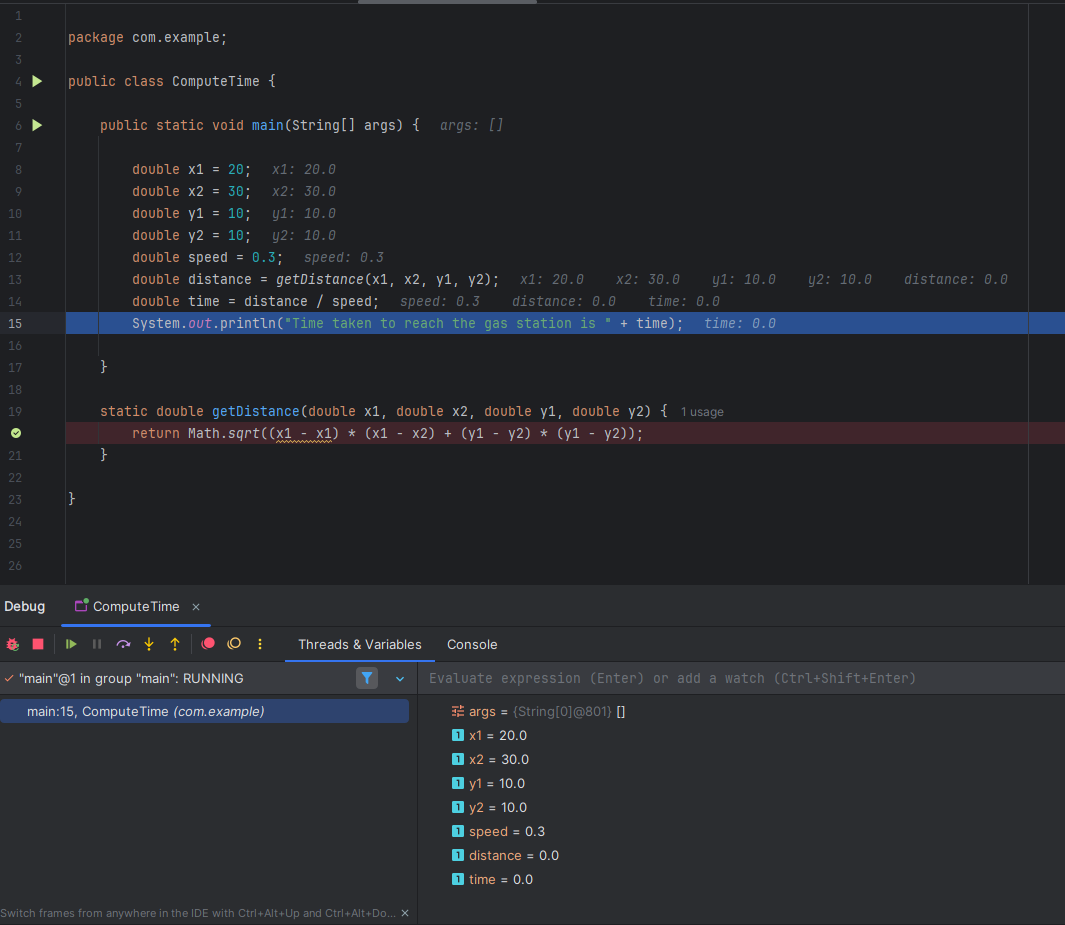
Во время проверки не были учтены дублирующиеся (3,3,3) значения переменных. При таком раскладе программа работает некорректно и выдает нулевой результат.

Исправим код:

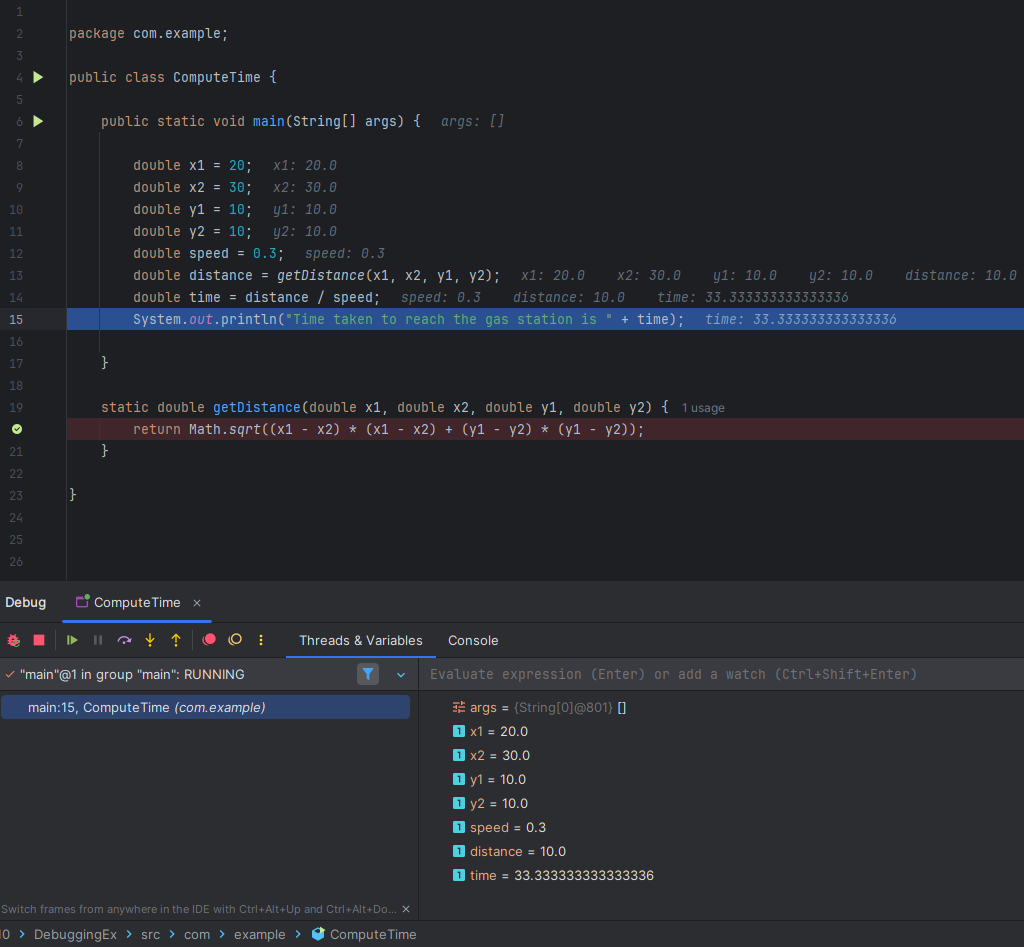


1. **Упражнение 2**

* Импортируйте и откройте проект DebuggingEx.
* Просмотрите ComputeTime.java.
* Запустите программу с отладчиком NetBeans для отладки этой программы:
* Установите точку останова в методе getDistance.
* Нажмите Ввод для перехода к следующей строке .
* Просмотрите значения переменных x1, x2, y1, y2, speed,distance и time.
* Можно ли выявить ошибку?

****

Исправим:

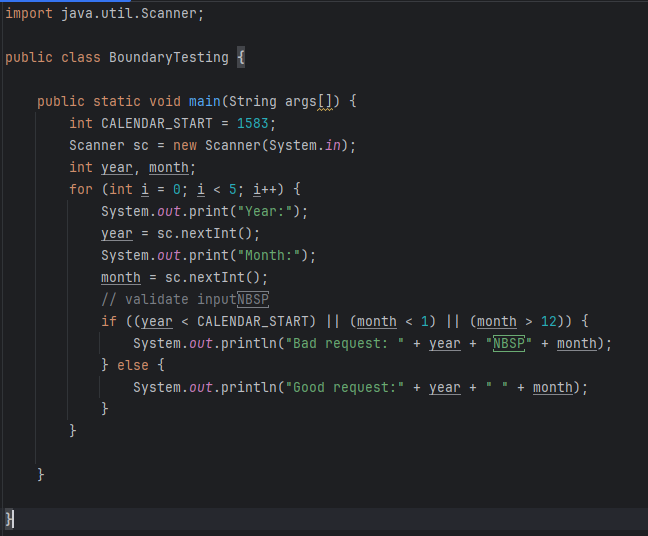


1. **Упражнение 3**

* Импортируйте и откройте проект DebuggingEx.
* Просмотрите BoundaryTesting.java.
* Проверьте ввод с помощью выполнения программы со следующими граничными тестовыми значениями для года и месяца:

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание





1. **Практика 8.**

Напишите программу, которая моделирует футбольную лигу и отслеживает статистику сезона. Тщательно рассмотрите, какие данные следует хранить в массиве, а какие данные следует хранить в ArrayList. На основе описания лиги разработайте классы с полями и методами. Вам также потребуется тестовый класс, который содержит главный метод. Все поля должны иметь значение private. Предоставьте все необходимые методы чтения и устанавливающие методы.

Main:

public class Main {  
 public static void main(String[] args) {  
 Scheduler scheduler = new Scheduler();  
 scheduler.startSeason();  
 }  
}

Scheduler:

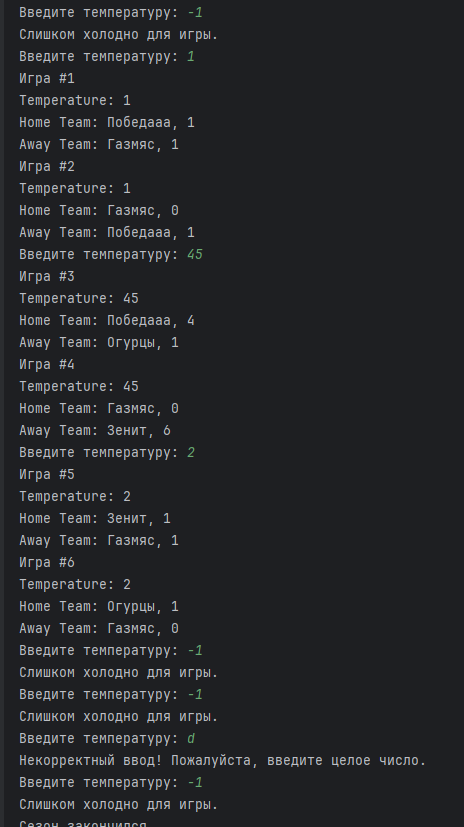
import java.util.\*;  
  
public class Scheduler {  
 private final List<Team> teams;  
 private final List<Game> games;  
 private int coldWeeksCount = 0;  
 private int totalTemperature = 0;  
 private int gamesPlayed = 0;  
 private int hottestTemp = Integer.*MIN\_VALUE*;  
  
 public Scheduler() {  
 teams = new ArrayList<>();  
 games = new ArrayList<>();  
 teams.add(new Team("Огурцы"));  
 teams.add(new Team("Газмяс"));  
 teams.add(new Team("Зенит"));  
 teams.add(new Team("Победааа"));  
 }  
  
 public void startSeason() {  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (coldWeeksCount < 3) {  
 int temperature = getTemperature();  
  
 if (temperature < 0) {  
 coldWeeksCount++;  
 System.*out*.println("Слишком холодно для игры.");  
 continue;  
 }  
  
 coldWeeksCount = 0;  
 scheduleGames(temperature);  
  
 }  
 printSeasonStatistics();  
 }  
  
 public void scheduleGames(int temperature) {  
 Random random = new Random();  
 totalTemperature += temperature;  
 gamesPlayed += 2;  
 if (temperature > hottestTemp) {  
 hottestTemp = temperature;  
 }  
  
 for (int i = 0; i < 2; i++) {  
 Team homeTeam = teams.get(random.nextInt(teams.size()));  
 Team awayTeam;  
 do {  
 awayTeam = teams.get(random.nextInt(teams.size()));  
 } while (homeTeam == awayTeam);  
  
 Game game = new Game(temperature, homeTeam, awayTeam);  
 games.add(game);  
 System.*out*.println(game.getGameStatistics());  
 }  
 }  
  
 public void printSeasonStatistics() {  
 System.*out*.println("Сезон закончился");  
 System.*out*.println("\*\*\*\*\*\*\*\*\*РЕЗУЛЬТАТЫ\*\*\*\*\*\*\*\*\*");  
 for (Team team : teams) {  
 System.*out*.println(team.getStatistics());  
 }  
  
 for (Game game : games) {  
 System.*out*.println(game.getGameStatistics());  
 }  
  
 double averageTemp = (double) totalTemperature / gamesPlayed;  
 System.*out*.println("Самая горячая температура: " + hottestTemp);  
 System.*out*.println("Средняя температура: " + averageTemp);  
 }  
  
  
 private int getTemperature() {  
 Scanner scanner = new Scanner(System.*in*);  
 int temperature;  
  
 while (true) {  
 System.*out*.print("Введите температуру: ");  
 try {  
 temperature = scanner.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.*out*.println("Некорректный ввод! Пожалуйста, введите целое число.");  
 scanner.next();  
 }  
 }  
 return temperature;  
 }  
}

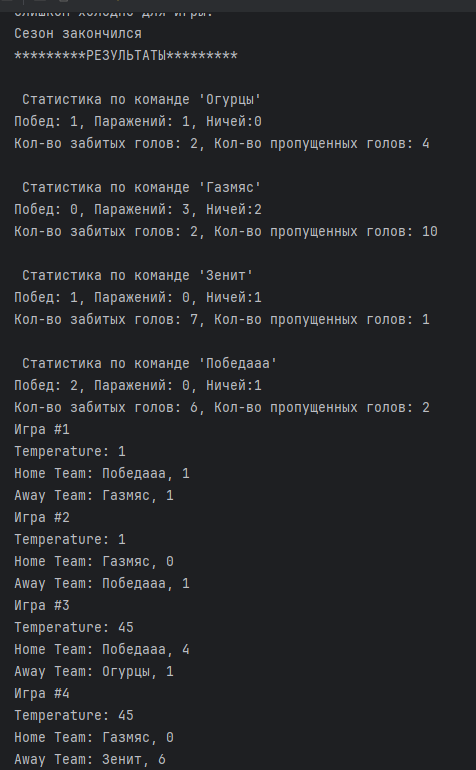
Game:

public class Game {  
 private static int *gameCounter* = 1;  
 private final int gameId;  
 private final int temperature;  
 private final Team homeTeam;  
 private final Team awayTeam;  
 private final int homeScore;  
 private final int awayScore;  
  
 public Game(int temperature, Team homeTeam, Team awayTeam) {  
 this.gameId = *gameCounter*++;  
 this.temperature = temperature;  
 this.homeTeam = homeTeam;  
 this.awayTeam = awayTeam;  
 this.homeScore = generateScore(temperature);  
 this.awayScore = generateScore(temperature);  
 updateTeamStatistics();  
 }  
  
 private int generateScore(int temperature) {  
 return (int) (Math.*random*() \* (temperature / 5 + 1));  
 }  
  
 private void updateTeamStatistics() {  
 if (homeScore > awayScore) {  
 homeTeam.recordWin(homeScore, awayScore);  
 awayTeam.recordLoss(awayScore, homeScore);  
 } else if (homeScore < awayScore) {  
 homeTeam.recordLoss(homeScore, awayScore);  
 awayTeam.recordWin(awayScore, homeScore);  
 } else {  
 homeTeam.recordTie(homeScore, awayScore);  
 awayTeam.recordTie(awayScore, homeScore);  
 }  
 }  
  
 public String getGameStatistics() {  
 return "Игра #" + gameId + "\nTemperature: " + temperature +  
 "\nHome Team: " + homeTeam.getName() + ", " + homeScore +  
 "\nAway Team: " + awayTeam.getName() + ", " + awayScore;  
 }  
}

Team:

public class Team {  
 private final String name;  
 private int wins;  
 private int losses;  
 private int ties;  
 private int pointsScored;  
  
 public int getPointsAllowed() {  
 return pointsAllowed;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getWins() {  
 return wins;  
 }  
  
 public int getLosses() {  
 return losses;  
 }  
  
 public int getTies() {  
 return ties;  
 }  
  
 public int getPointsScored() {  
 return pointsScored;  
 }  
  
 private int pointsAllowed;  
  
 public Team(String name) {  
 this.name = name;  
 this.wins = 0;  
 this.losses = 0;  
 this.ties = 0;  
 this.pointsScored = 0;  
 this.pointsAllowed = 0;  
 }  
  
  
 public void recordWin(int scored, int allowed) {  
 wins++;  
 pointsScored += scored;  
 pointsAllowed += allowed;  
 }  
  
 public void recordLoss(int scored, int allowed) {  
 losses++;  
 pointsScored += scored;  
 pointsAllowed += allowed;  
 }  
  
 public void recordTie(int scored, int allowed) {  
 ties++;  
 pointsScored += scored;  
 pointsAllowed += allowed;  
 }  
  
 public String getStatistics() {  
 return "\n Статистика по команде '" + name + "'\nПобед: " + wins + ", Паражений: " + losses + ", Ничей:" + ties +  
 "\nКол-во забитых голов: " + pointsScored + ", Кол-во пропущенных голов: " + pointsAllowed;  
 }  
}





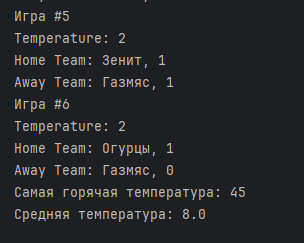


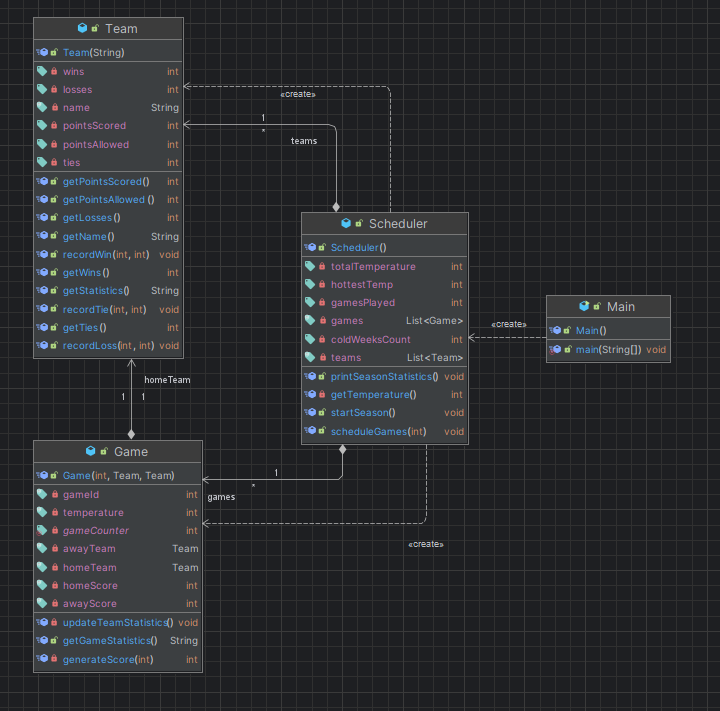
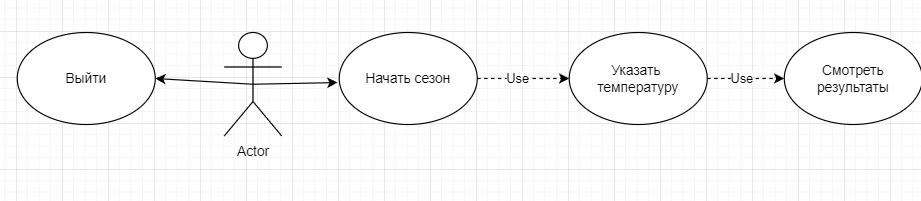
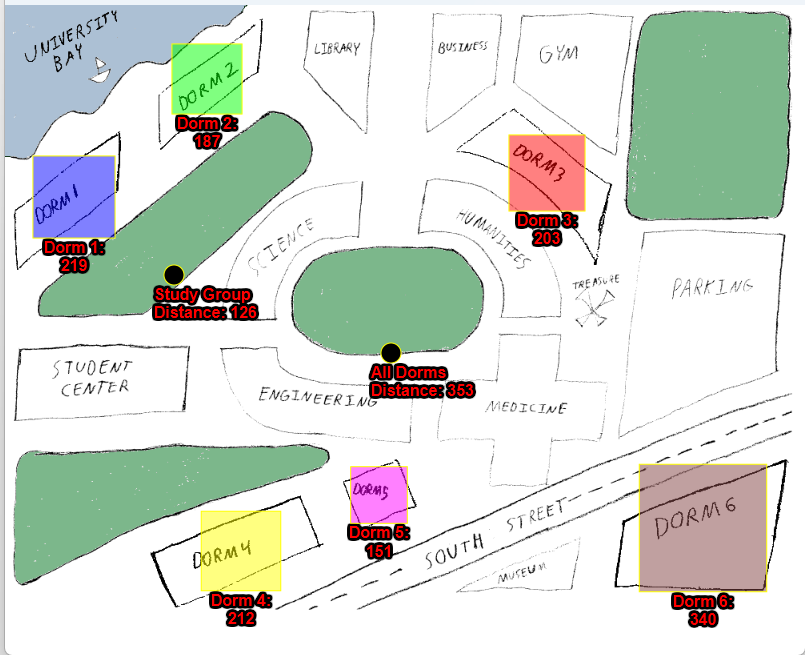
Диаграмма классов:   


Диаграмма вариантов использования:



1. **По занятию 9.1**
2. **Упражнение 1**

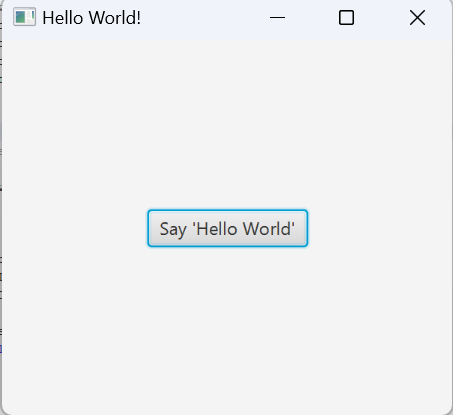
* Запустите CampusMap.jar.
* Сопоставьте каждый квадрат с правильным корпусом на карте.
* Оцените и настройте количество проживающих в корпусе
* Нажмите и перетащите текст под каждым квадратом.
* Просмотрите изменения в следующих центральных точках:
* Все студенты во всех корпусах
* Студенческая группа из трех друзей проживает в корпусах 1, 2 и 4

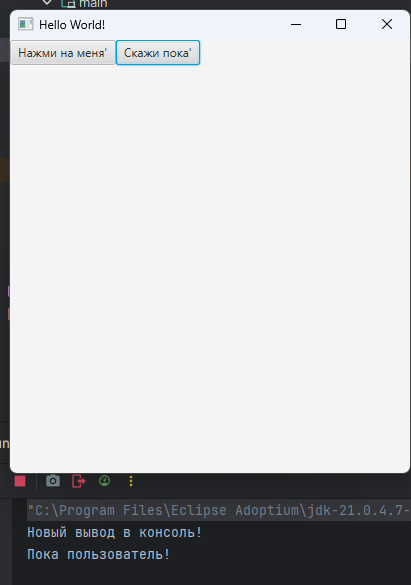


1. **Упражнение 2**

* Создайте проект JavaFX.
* Java должна предоставить программу по умолчанию.
* Поэкспериментируйте с программой. Можете ли вы внести данные изменения?
* Измените метку кнопки.
* Измените то, что выводится при нажатии кнопки.
* Создайте другую кнопку и отобразите обе кнопки.
* Измените стандартный размер окна приложения.

По умолчанию:





1. **Упражнение 3**

• Измените текущий проект JavaFX.

– Проведем небольшой эксперимент.

• После добавления кнопки в корневой узел попробуйте изменить расположение.

– btn1.setLayoutY(100);

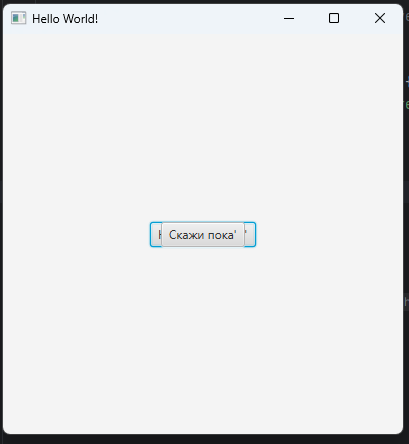
• Изменится ли расположение кнопки, если для корневого узла не было указано значение StackPane? Попробуйте выполнить следующие альтернативные способы:

– TilePane

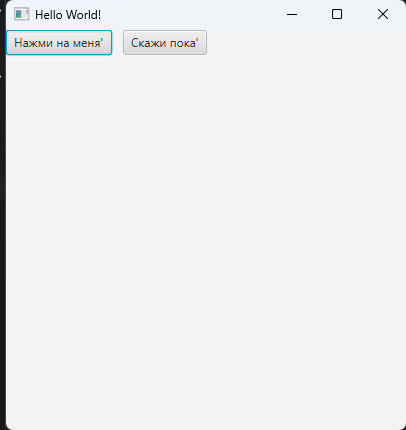
– VBox

– Group

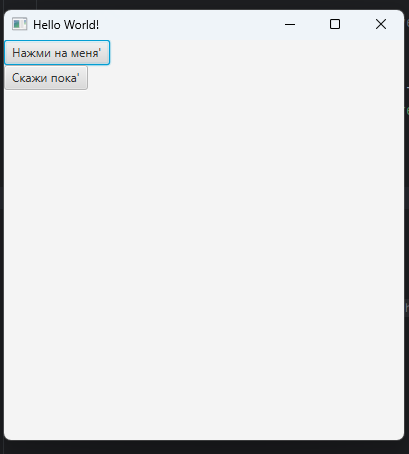
setLayoutY  не будет иметь эффекта, т.к. StackPane центрирует элементы.



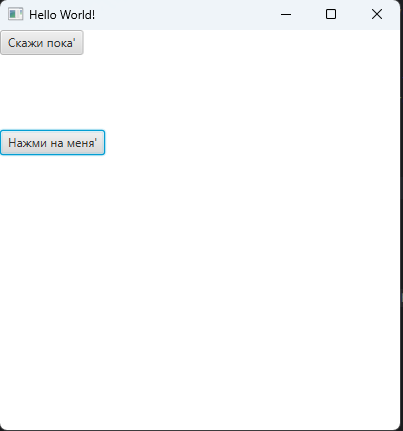
TilePane:



VBox:



Group: setLayoutY работает



1. **Упражнение 4**

• Измените текущий проект JavaFX.

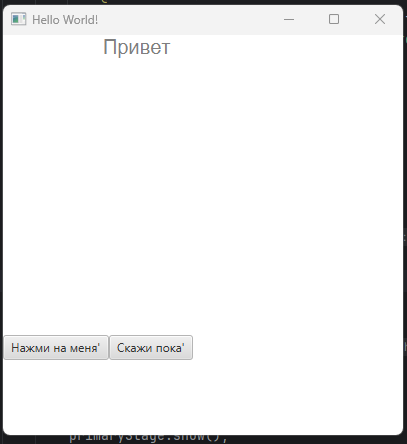
– Можно еще поэкспериментировать.

• Можете ли вы сообразить, как сделать следующее?

– Создать панель HBox и добавить в нее несколько кнопок.

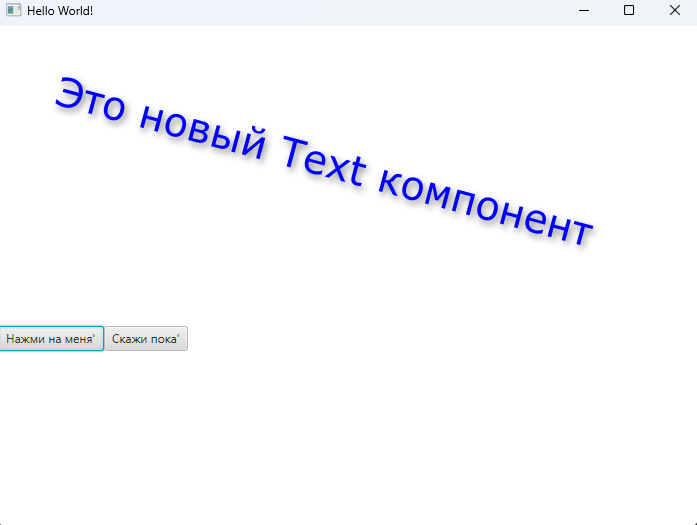
– Добавить панель HBox в корневой узел Group.

– Расположить HBox ближе к нижней части окна.

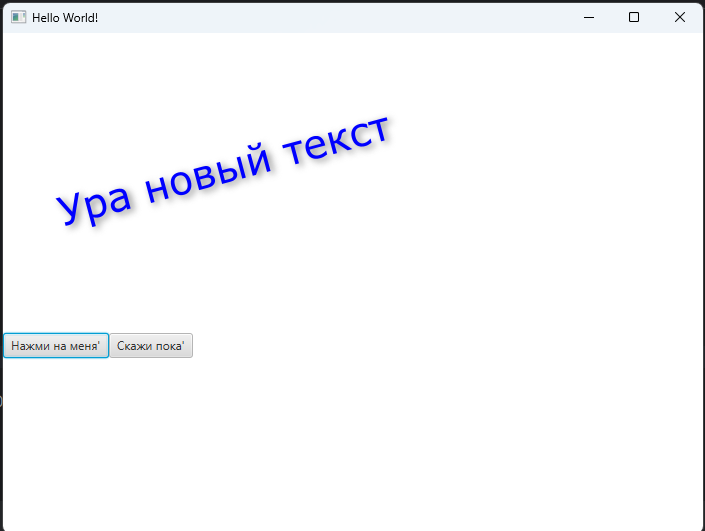


Дополните результаты упражнения 4, добавив объект класса Text (слайд 31 и статья "Создание форм в JavaFx") над горизонтальной панелью с кнопками. По нажатию разных кнопок меняйте выводимый текст надписи (метод класса Text: setText()) вместо печати в консоль. Поэкспериментируйте со шрифтами и размером текста.

Программа до нажатия на кнопки:



Программа после нажатия на кнопку “Нажми на меня”



Программа после нажатия на кнопку “Скажи пока”. Надпись плавно исчезает, вращаясь по часовой стрелке на 360 градусов с задержкой в 500 миллисекунд.

