Exam - 23 Oct 2019

ВРЕМЕ:

ЗАДАЧА 1: - 0:17 мин.

ЗАДАЧА 2: – 0:28 мин.

ЗАДАЧА 3 : – 1:50 мин

Общо: 2:40 мин

Задача 1:

import java.util.ArrayDeque;  
import java.util.Arrays;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
  
public class Task1 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 ArrayDeque<Integer> tasksStack = new ArrayDeque<>();  
 Arrays.*stream*(scanner.nextLine().split(", "))  
 .mapToInt(Integer::*parseInt*)  
 .forEach(e -> tasksStack.push(e));  
  
 ArrayDeque<Integer> threadsQueue = Arrays.*stream*(scanner.nextLine().split("\\s+"))  
 .map(Integer::*parseInt*)  
 .collect(Collectors.*toCollection*(ArrayDeque::new));  
  
 int killedTask = Integer.*parseInt*(scanner.nextLine());  
 int threadThatKilledTheTask = 0;  
  
 while (tasksStack.peek() != killedTask) {  
 if (threadsQueue.peek() >= tasksStack.peek()) {  
 threadsQueue.poll();  
 tasksStack.pop();  
 } else {  
 threadsQueue.poll();  
 }  
 }  
  
 threadThatKilledTheTask = threadsQueue.peek();  
  
 System.*out*.printf("Thread with value %d killed task %d%n", threadThatKilledTheTask, killedTask);  
 while (!threadsQueue.isEmpty()) {  
 System.*out*.print(threadsQueue.poll() + " ");  
 }  
 }  
}

Задача 2:

import java.util.Scanner;  
  
public class Task2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 int rows = scanner.nextInt();  
 int cols = scanner.nextInt();  
 scanner.nextLine();  
  
 int[][] matrix = *getMatrix*(scanner, rows, cols);  
  
 String command = scanner.nextLine();  
 while (!command.equals("Bloom Bloom Plow")) {  
 String[] tokens = command.split("\\s+");  
 int rowFlower = Integer.*parseInt*(tokens[0]);  
 int colFlower = Integer.*parseInt*(tokens[1]);  
  
 if (*isOutOfBounds*(matrix, rowFlower, colFlower)) {  
 System.*out*.println("Invalid coordinates.");  
 continue;  
 }  
  
 // по редове  
 matrix[rowFlower][colFlower] += 1;  
 for (int r = 0; r < rowFlower; r++) {  
 matrix[r][colFlower]++;  
 }  
 for (int r = rowFlower + 1; r < rows; r++) {  
 matrix[r][colFlower]++;  
 }  
 //по колони  
 for (int c = 0; c < colFlower; c++) {  
 matrix[rowFlower][c]++;  
 }  
 for (int c = colFlower + 1; c < cols; c++) {  
 matrix[rowFlower][c]++;  
 }  
 command = scanner.nextLine();  
 }  
 *printMatrix*(matrix, rows, cols);  
 }  
  
 private static int[][] getMatrix(Scanner scanner, int rows, int cols) {  
 int[][] matrix = new int[rows][cols];  
  
 for (int r = 0; r < rows; r++) {  
 for (int c = 0; c < cols; c++) {  
 matrix[r][c] = 0;  
 }  
 }  
 return matrix;  
 }  
  
 private static boolean isOutOfBounds(int[][] matrix, int r, int c) {  
 return r < 0 || r >= matrix.length || c < 0 || c >= matrix[r].length;  
 }  
  
 public static void printMatrix(int[][] matrix, int rows, int cols) {  
 for (int r = 0; r < rows; r++) {  
 for (int c = 0; c < cols; c++) {  
 System.*out*.print(matrix[r][c] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}

Задача 3:

package classroom;  
  
public class Main {  
 public static void main(String[] args){  
 // Initialize the repository  
 Classroom classroom = new Classroom(10);  
// Initialize entities  
 Student student = new Student("Peter", "Parker", "Geometry");  
 Student studentTwo = new Student("Sarah", "Smith", "Algebra");  
 Student studentThree = new Student("Sam", "Winchester", "Algebra");  
 Student studentFour = new Student("Dean", "Winchester", "Music");  
 Student studentFive = new Student("Ellie", "Goulding", "Music");  
// Print Student  
 System.*out*.println(student);  
 // Student: First Name = Peter, Last Name = Parker, Subject = Geometry  
// Register Student  
 String register = classroom.registerStudent(student);  
 System.*out*.println(register); // Added student Peter Parker  
 String registerTwo = classroom.registerStudent(studentTwo);  
 String registerThree = classroom.registerStudent(studentThree);  
 String registerFour = classroom.registerStudent(studentFour);  
// Dismiss Student  
 String dismissed = classroom.dismissStudent(student);  
 System.*out*.println(dismissed); // Removed student Peter Parker  
 String dismissedTwo = classroom.dismissStudent(studentFive);  
 System.*out*.println(dismissedTwo); // Student not found  
// Subject info  
 String subjectInfo = classroom.getSubjectInfo("Algebra");  
 System.*out*.println(subjectInfo);  
// Subject: Algebra  
// Students:  
// Sarah Smith  
// Sam Winchester  
 String anotherInfo = classroom.getSubjectInfo("Art");  
 System.*out*.println(anotherInfo); // No students enrolled for the subject  
// Get Student  
 System.*out*.println(classroom.getStudent("Dean", "Winchester"));  
// Student: First Name = Dean, Last Name = Winchester, Subject = Music  
  
 }  
}

package classroom;  
  
public class Student {  
 public String firstName;  
 public String lastName;  
 public String bestSubject;  
  
 public Student(String firstName, String lastName, String bestSubject) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.bestSubject = bestSubject;  
 }  
  
 public String getFirstName() {  
 return this.firstName;  
 }  
  
 public String getLastName() {  
 return this.lastName;  
 }  
  
 public String getBestSubject() {  
 return this.bestSubject;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Student: First Name= %s, Last Name= %s, Best Subject= %s"  
 , this.firstName, this.lastName, this.bestSubject);  
 }  
}

package classroom;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Classroom {  
 public int capacity;  
 public List<Student> students;  
  
 public Classroom(int capacity) {  
 this.capacity = capacity;  
 this.students = new ArrayList<>();  
 }  
  
 public int getCapacity() {  
 return this.capacity;  
 }  
  
 public List<Student> getStudents() {  
 return this.students;  
 }  
  
 public int getStudentCount() {  
 return this.students.size();  
 }  
  
 public String registerStudent(Student student) {  
 if (this.students.size() < capacity && !this.students.contains(student)) {  
 this.students.add(student);  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < this.students.size(); i++) {  
 if (this.students.get(i).equals(student)) {  
 sb.append("Added student ").append(student.getFirstName()).append(" ").append(student.getLastName());  
 }  
 }  
 return sb.toString();  
 } else if (this.students.size() < capacity && this.students.contains(student)) {  
 return "Student is already in the classroom";  
 } else if (this.students.size() >= capacity){  
 return "No seats in the classroom";  
 }  
 return null;  
 }  
  
 public String dismissStudent(Student student) {  
 if (this.students.contains(student)) {  
 this.students.remove(student);  
 return String.*format*("Removed student %s %s", student.getFirstName(), student.getLastName());  
 } else {  
 return "Student not found";  
 }  
 }  
  
 public String getSubjectInfo(String subject) {  
  
 boolean isEqualSubject = false;  
  
 for (Student student : this.students) {  
 if (student.getBestSubject().equals(subject)) {  
 isEqualSubject = true;  
 break;  
 }  
 }  
  
 if (!isEqualSubject) {  
 return "No students enrolled for the subject";  
 } else {  
 StringBuilder sb = new StringBuilder();  
 sb.append("Subject: ").append(subject).append(System.*lineSeparator*());  
 sb.append("Students:").append(System.*lineSeparator*());  
 for (int i = 0; i < this.students.size(); i++) {  
 if (this.students.get(i).getBestSubject().equals(subject)) {  
 if (i != this.students.size() - 1) {  
 sb.append(this.students.get(i).getFirstName()).append(" ").append(this.students.get(i).getLastName()).append(System.*lineSeparator*());  
 } else {  
 sb.append(this.students.get(i).getFirstName()).append(" ").append(this.students.get(i).getLastName());  
 }  
 }  
 }  
 return sb.toString();  
 }  
 }  
  
 public String getStudent(String firstName, String lastName) {  
 for (Student student : this.students) {  
 if (student.getFirstName().equals(firstName) && student.getLastName().equals(lastName)) {  
 return String.*format*("Student: First Name= %s, Last Name= %s, Best Subject= %s", student.getFirstName(), student.getLastName(), student.getBestSubject());  
 }  
 }  
 return null;  
 }  
  
 public String getStatistics() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("Classroom size: ").append(this.students.size()).append(System.*lineSeparator*());  
 for (Student student : this.students) {  
 sb.append("==Student: First Name= ").append(student.getFirstName()).append(", ")  
 .append("Last Name= ").append(student.getLastName()).append(", ")  
 .append("Best Subject= ").append(student.getBestSubject()).append(System.*lineSeparator*());  
 }  
 return sb.toString().trim();  
 }  
}