FINAL EXAM

ВРЕМЕ:

ЗАДАЧА 1: - 1:00мин. 66/100

ЗАДАЧА 2: – 33 мин.

ЗАДАЧА 3: – 2:37мин 0/100

Общо: 166/300

Задача 1:

import java.util.Scanner;  
  
public class First {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String input = scanner.nextLine();  
  
 String command = scanner.nextLine();  
 while (!command.equals("Travel")) {  
 String[] tokens = command.split(":");  
 String action = tokens[0];  
 StringBuilder sb = new StringBuilder();  
  
 switch (action) {  
 case "Add Stop":  
 int index = Integer.*parseInt*(tokens[1]);  
 String string = tokens[2];  
  
 if (index >= 0 && index < input.length()) {  
 sb.append(input);  
 sb.insert(index, string);  
 input = sb.toString();  
 System.*out*.println(input);  
 }  
 break;  
  
 case "Remove Stop":  
 int startI = Integer.*parseInt*(tokens[1]);  
 int endI = Integer.*parseInt*(tokens[2]);  
  
 if (startI >= 0 && startI < input.length() && endI >= 0 && endI < input.length() && startI <= endI) {  
 sb = new StringBuilder(input);  
 sb.replace(startI, endI + 1, "");  
 input = sb.toString();  
 System.*out*.println(input);  
 }  
 break;  
  
 case "Switch":  
 String oldString = tokens[1];  
 String newString = tokens[2];  
 int findIndex = input.indexOf(oldString);  
  
 if (input.contains(oldString)) {  
 while (findIndex != -1) {  
 String start = input.substring(0, findIndex);  
 String switchString = input.substring(findIndex, oldString.length() + findIndex);  
 String end = input.substring(oldString.length() + findIndex);  
  
 sb = new StringBuilder();  
 sb.append(start);  
 sb.append(newString);  
 sb.append(end);  
  
 input = sb.toString();  
 findIndex = input.indexOf(oldString);  
  
 }  
 input = sb.toString();  
 System.*out*.println(input);  
 }  
 break;  
  
 }  
 command = scanner.nextLine();  
 }  
 System.*out*.printf("Ready for world tour! Planned stops: %s%n", input);  
 }  
}

Оправено решение:

import java.util.Scanner;  
  
public class WorldTour\_01 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String input = scanner.nextLine();  
  
 String command = scanner.nextLine();  
 while (!command.equals("Travel")) {  
 String[] tokens = command.split(":");  
 String action = tokens[0];  
 StringBuilder sb = new StringBuilder();  
  
 switch (action) {  
 case "Add Stop":  
 int index = Integer.*parseInt*(tokens[1]);  
 String string = tokens[2];  
  
 if (index >= 0 && index < input.length()) {  
 sb.append(input);  
 sb.insert(index, string);  
 input = sb.toString();  
 }  
 System.*out*.println(input);  
 break;  
  
 case "Remove Stop":  
 int startI = Integer.*parseInt*(tokens[1]);  
 int endI = Integer.*parseInt*(tokens[2]);  
  
 if (startI >= 0 && startI < input.length() && endI >= 0 && endI < input.length() && startI < endI) {  
 sb = new StringBuilder(input);  
 sb.replace(startI, endI + 1, "");  
 input = sb.toString();  
 }  
 System.*out*.println(input);  
 break;  
  
 case "Switch":  
 String oldString = tokens[1];  
 String newString = tokens[2];  
 int findIndex = input.indexOf(oldString);  
  
 if (input.contains(oldString)) {  
 while (findIndex != -1) {  
 String start = input.substring(0, findIndex);  
 String switchString = input.substring(findIndex, oldString.length() + findIndex);  
 String end = input.substring(oldString.length() + findIndex);  
  
 sb = new StringBuilder();  
 sb.append(start);  
 sb.append(newString);  
 sb.append(end);  
  
 input = sb.toString();  
 findIndex = input.indexOf(oldString);  
  
 }  
 input = sb.toString();  
 }  
 System.*out*.println(input);  
 break;  
 }  
 command = scanner.nextLine();  
 }  
 System.*out*.printf("Ready for world tour! Planned stops: %s%n", input);  
 }  
}

Задача 2:

import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Second {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String input = scanner.nextLine();  
 List<String> destinations = new ArrayList<>();  
 int points = 0;  
  
 String regex = "([=]|[\\/])([A-Z][A-Za-z]{2,})\\1";  
 Pattern pattern = Pattern.*compile*(regex);  
 Matcher matcher = pattern.matcher(input);  
 while (matcher.find()) {  
 destinations.add(matcher.group(2));  
 points += matcher.group(2).length();  
 }  
 System.*out*.print("Destinations: ");  
 for (int i = 0; i < destinations.size(); i++) {  
 if (i != destinations.size() - 1) {  
 System.*out*.print(destinations.get(i) + ", ");  
 } else {  
 System.*out*.print(destinations.get(i));  
 }  
 }  
 System.*out*.println();  
 System.*out*.printf("Travel Points: %d", points);  
 }  
}

Задача 3:

import java.util.\*;  
import java.util.zip.DeflaterOutputStream;  
  
public class Third {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 Map<String, Integer> registerForPlants = new LinkedHashMap<>();  
 Map<String, Double> ratings = new LinkedHashMap<>();  
  
 int numberOfPlants = Integer.*parseInt*(scanner.nextLine());  
 for (int i = 0; i < numberOfPlants; i++) {  
 String[] plants = scanner.nextLine().split("<->");  
 String plant = plants[0];  
 int rarity = Integer.*parseInt*(plants[1]);  
  
 if (!registerForPlants.containsKey(plant)) {  
 registerForPlants.put(plant, rarity);  
 } else {  
 registerForPlants.put(plant, registerForPlants.get(plant) + rarity);  
 }  
 }  
  
 String command = scanner.nextLine();  
 while (!command.equals("Exhibition")) {  
 String[] tokens = command.split("([\\:][\\s+])|([\\s+][\\-][\\s+])");  
 String action = tokens[0];  
 String plant = tokens[1];  
  
 switch (action) {  
 case "Rate":  
 double currentRating = Integer.*parseInt*(tokens[2]);  
 ratings.putIfAbsent(plant, currentRating);  
 //List<Double> gradesList = new ArrayList<>();  
 //grades.add(currentRating);  
  
 currentRating += currentRating / 2;  
 ratings.put(plant, currentRating);  
  
 break;  
  
 case "Update":  
 int newRarity = Integer.*parseInt*(tokens[2]);  
 if (registerForPlants.containsKey(plant)) {  
 registerForPlants.put(plant, newRarity);  
 } else {  
 System.*out*.println("error");  
 }  
  
 break;  
  
 case "Reset":  
 currentRating = 0.0;  
 if (ratings.containsKey(plant)) {  
 ratings.put(plant, currentRating);  
 } else {  
 System.*out*.println("error");  
 }  
  
 break;  
 }  
 command = scanner.nextLine();  
 }  
  
 System.*out*.println("Plants for the exhibition:");  
 registerForPlants  
 .entrySet()  
 .stream()  
 .sorted((f, s) -> {  
 int result = s.getValue().compareTo(f.getValue());  
 if (result == 0) {  
 }  
 return result;  
 })  
 .forEach(entry -> {  
 System.*out*.printf("- %s; Rarity: %d; Rating: %.2f%n", entry.getKey(), entry.getValue(), ratings.get(entry.getValue()));  
 });  
 }  
}

import java.util.\*;  
  
public class PlantDiscovery\_03 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 Map<String, Integer> registerForPlants = new LinkedHashMap<>();  
 Map<String, List<Double>> ratings = new LinkedHashMap<>();  
  
 int numberOfPlants = Integer.*parseInt*(scanner.nextLine());  
 for (int i = 0; i < numberOfPlants; i++) {  
 String[] plants = scanner.nextLine().split("<->");  
 String plant = plants[0];  
 int rarity = Integer.*parseInt*(plants[1]);  
  
 registerForPlants.put(plant, rarity);  
 ratings.put(plant, new ArrayList<>());  
 }  
  
 String command = scanner.nextLine();  
 while (!command.equals("Exhibition")) {  
 String[] tokens = command.split("([\\:][\\s+])|([\\s+][\\-][\\s+])");  
 String action = tokens[0];  
 String plant = tokens[1];  
  
 switch (action) {  
 case "Rate":  
 double currentRating = Double.*parseDouble*(tokens[2]);  
  
 if (!ratings.containsKey(plant)) {  
 System.*out*.println("error");  
 } else {  
 List<Double> gradesList = ratings.get(plant);  
 gradesList.add(currentRating);  
 ratings.put(plant, gradesList);  
 }  
 break;  
  
 case "Update":  
 int newRarity = Integer.*parseInt*(tokens[2]);  
 if (registerForPlants.containsKey(plant)) {  
 registerForPlants.put(plant, newRarity);  
 } else {  
 System.*out*.println("error");  
 }  
 break;  
  
 case "Reset":  
 if (ratings.containsKey(plant)) {  
 ratings.get(plant).clear();  
 } else {  
 System.*out*.println("error");  
 }  
 break;  
  
 default:  
 System.*out*.println("error");  
 }  
 command = scanner.nextLine();  
 }  
  
 System.*out*.println("Plants for the exhibition:");  
 registerForPlants  
 .entrySet()  
 .stream()  
 .sorted((f, s) -> {  
 int result = s.getValue().compareTo(f.getValue());  
 if (result == 0) {  
 if (ratings.get(s.getKey()).isEmpty() || ratings.get(f.getKey()).isEmpty()) {  
 return result;  
 } else {  
 double asd = ratings.get(s.getKey()).stream().mapToDouble(Double::doubleValue).average().orElse(0.0) -  
 ratings.get(f.getKey()).stream().mapToDouble(Double::doubleValue).average().orElse(0.0);  
 return (int) asd;  
 }  
 } else {  
 return result;  
 }  
 })  
 .forEach(entry -> {  
 System.*out*.printf("- %s; Rarity: %d; Rating: %.2f%n", entry.getKey(), entry.getValue(), ratings.get(entry.getKey()).stream().mapToDouble(Double::doubleValue).average().orElse(0.00));  
 });  
  
 }  
}