בס"ד   
  
קריאה מקובץ:

כל שורה בקובץ זה אובייקט - כמות העמודות בשורה זה כמות השדות

שלב 1) יוצרים מחלקה שמייצגת את האובייקט - אדם/לקוח

שלב 2)לדאוג שהפונקציה שקוראת מהקובץ תחזיר רשימה של אוביקטים

שלב ג) עכשיו שיש רשימה מסודרת של האובייקטים לעבור עליה בלולאה למצוא את הסעיף א ,עם GET , לסכום וכו

שלב ד) לשלוח את התשובה בבקשת POST API - שימו לב ,לנתיב ול3 הפרמטרים - שזה המזהה ,התשובה ,והשאלה

שאלה 2 ) תמונה - או ע"י דוגמאות עוברים על תמונה , או עם צאט ,או לבד

את התשובה מעבירים בבקשת POST - שאלה ,תשובה ,ומזהה

שאלה 3- בדר"כ שאלת חשיבה - אם זה קשור בזמן reads.sleep T - זה עובד על זמן

שימו לב - תמיד לשנות ערכים של API - שאתם שולחים ערכים נכונים - המזהה שלכם ,מספר השאלה, והתשובה

יש לכם כאן קובץ של שאלה 1 ושאלה 2 , תשתמשו - לשנות לרלוונטי לכם

והכי חשוב!!!!!!!!!

לעשות את המבחן מועד א, ו-ב לפחות פעמיים

תשובות לשאלה ראשונה:  
  
  
import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.utils.URIBuilder;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.util.EntityUtils;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.URI;

import java.util.\*;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

public class Main {

public static final String PATH= "https://app.seker.live/fm1/answer-file";

public static final String MAGIC= "Hd5GB0C";

public static void main(String[] args) {

List<Costumer> lines = readFile();

//מהו ההפרש בין סכום הרכישה הממוצע של גברים לסכום הרכישה הממוצע של נשים?

// חישוב סכום הרכישה הממוצע של גברים

double totalPurchaseMen = 0;

double countMen = 0;

// חישוב סכום הרכישה הממוצע של נשים

double totalPurchaseWomen = 0;

double countWomen = 0;

// עובר על כל הלקוחות ומחשב את סכום הרכישה לכל מגדר

for (Costumer costumer : lines) {

if ("Male".equals(costumer.getGender())) {

totalPurchaseMen += costumer.getPurchase();

countMen++;

} else if ("Female".equals(costumer.getGender())) {

totalPurchaseWomen += costumer.getPurchase();

countWomen++;

}

}

// חישוב הממוצע של סכום הרכישה לכל מגדר

double averagePurchaseMen = totalPurchaseMen / countMen;

double averagePurchaseWomen = totalPurchaseWomen / countWomen;

// חישוב ההפרש בין הממוצעים

double difference = averagePurchaseMen - averagePurchaseWomen;

// הדפסת התוצאה

System.out.println("ההפרש בין סכום הרכישה הממוצע של גברים לסכום הרכישה הממוצע של נשים הוא: " + difference);

sendAnswer(MAGIC,"1", " 0.7513162785661507");

// יצירת מפת ספירה עבור ראשי התיבות

Map<String, Integer> initialsCountMap = new HashMap<>();

// עובר על כל הלקוחות ומחשב את ראשי התיבות

for (Costumer costumer : lines) {

String initials = String.valueOf(costumer.getFirstName().charAt(0)) +

String.valueOf(costumer.getLastName().charAt(0));

// עדכון הספירה של ראשי התיבות

initialsCountMap.put(initials, initialsCountMap.getOrDefault(initials, 0) + 1);

}

// חיפוש ראשי התיבות הנפוצים ביותר

String mostCommonInitials = null;

int maxCount = 0;

for (Map.Entry<String, Integer> entry : initialsCountMap.entrySet()) {

if (entry.getValue() > maxCount) {

mostCommonInitials = entry.getKey();

maxCount = entry.getValue();

}

}

// הדפסת ראשי התיבות הנפוצים ביותר

if (mostCommonInitials != null) {

System.out.println("הראשי תיבות הנפוצים ביותר הם: " + mostCommonInitials);

sendAnswer(MAGIC,"1","MS");

} else {

System.out.println("לא נמצאו ראשי תיבות.");

}

// משתנה לספירת נשים שהשם הפרטי מתחיל באותה אות כמו שם המשפחה

int count = 0;

// עובר על כל הלקוחות ומחפש נשים שהשם הפרטי מתחיל באותה אות כמו שם המשפחה

for (Costumer costumer : lines) {

// אם הלקוחה היא אישה

if (costumer.getGender().equalsIgnoreCase("Female")) {

// בודק אם השם הפרטי ושם המשפחה מתחילים באותה אות

if (costumer.getFirstName().charAt(0) == costumer.getLastName().charAt(0)) {

count++;

}

}

}

// הדפסת התוצאה

System.out.println("מספר הנשים שהשם הפרטי שלהן מתחיל באותה האות כמו שם המשפחה: " + count);

sendAnswer(MAGIC,"1","511");

// מפת סכום הרכישות לפי עיר

Map<String, Double> cityPurchases = new HashMap<>();

// מעדכן את סכום הרכישה של כל עיר

for (Costumer costumer : lines) {

String city = costumer.getCityLiving();

double purchase = costumer.getPurchase();

cityPurchases.put(city, cityPurchases.getOrDefault(city, 0.0) + purchase);

}

// מיון הערים לפי סכום הרכישה

List<Map.Entry<String, Double>> sortedCities = new ArrayList<>(cityPurchases.entrySet());

sortedCities.sort((entry1, entry2) -> entry2.getValue().compareTo(entry1.getValue()));

// הדפסת העיר השלישית בסכום הרכישה הגבוה ביותר

if (sortedCities.size() >= 3) {

String thirdCity = sortedCities.get(2).getKey();

System.out.println("העיר השלישית בסכום הרכישה הכי גבוה היא: " + thirdCity);

sendAnswer(MAGIC,"1","South Tracy");

} else {

System.out.println("אין מספיק ערים כדי למצוא את העיר השלישית בסכום הרכישה הגבוה ביותר.");

}

int currentYear = 2024; // שנה נוכחית

// מפת סכום הגיל לפי עיר

Map<String, Double> cityAgeSum = new HashMap<>();

Map<String, Integer> cityCount = new HashMap<>(); // מפת ספירת הלקוחות בכל עיר

// מעדכן את גיל כל לקוח ומוסיף אותו לסך הגיל של העיר

for (Costumer costumer : lines) {

String city = costumer.getCityLiving();

int age = currentYear - costumer.getBirth();

// עדכון סך הגיל של העיר

cityAgeSum.put(city, cityAgeSum.getOrDefault(city, 0.0) + age);

// עדכון ספירת הלקוחות בעיר

cityCount.put(city, cityCount.getOrDefault(city, 0) + 1);

}

// חישוב הגיל הממוצע עבור כל עיר

double maxAverageAge = -1;

String oldestCity = "";

for (String city : cityAgeSum.keySet()) {

double totalAge = cityAgeSum.get(city);

count = cityCount.get(city);

double averageAge = totalAge / count; // חישוב גיל ממוצע

// עדכון העיר עם הגיל הממוצע הגבוה ביותר

if (averageAge > maxAverageAge) {

maxAverageAge = averageAge;

oldestCity = city;

}

}

// הדפסת העיר עם הלקוחות המבוגרים ביותר

if (!oldestCity.isEmpty()) {

System.out.println("העיר שבה הלקוחות הם המבוגרים ביותר היא: " + oldestCity);

sendAnswer(MAGIC,"1","Josueton");

} else {

System.out.println("לא נמצאו ערים.");

}

currentYear = 2024; // שנה נוכחית

// מפת סכום הגיל לפי עיר

cityAgeSum = new HashMap<>();

cityCount = new HashMap<>();

// מעדכן את גיל כל לקוח ומוסיף אותו לסך הגיל של העיר

for (Costumer costumer : lines) {

String city = costumer.getCityLiving();

int age = currentYear - costumer.getBirth();

// עדכון סך הגיל של העיר

cityAgeSum.put(city, cityAgeSum.getOrDefault(city, 0.0) + age);

// עדכון ספירת הלקוחות בעיר

cityCount.put(city, cityCount.getOrDefault(city, 0) + 1);

}

currentYear = 2024;

// מפת סכום הגיל לפי עיר

cityAgeSum = new HashMap<>();

cityCount = new HashMap<>();

// מעדכן את גיל כל לקוח ומוסיף אותו לסך הגיל של העיר

for (Costumer costumer : lines) {

String city = costumer.getCityLiving();

int age = currentYear - costumer.getBirth();

// עדכון סך הגיל של העיר

cityAgeSum.put(city, cityAgeSum.getOrDefault(city, 0.0) + age);

// עדכון ספירת הלקוחות בעיר

cityCount.put(city, cityCount.getOrDefault(city, 0) + 1);

}

// חישוב הגיל הממוצע עבור כל עיר

double minAverageAge = Double.MAX\_VALUE;

String youngestCity = "";

for (String city : cityAgeSum.keySet()) {

double totalAge = cityAgeSum.get(city);

count = cityCount.get(city);

double averageAge = totalAge / count; // חישוב גיל ממוצע

// עדכון העיר עם הגיל הממוצע הנמוך ביותר (הצעירה ביותר)

if (averageAge < minAverageAge) {

minAverageAge = averageAge;

youngestCity = city;

}

}

// סופרים את התו הנפוץ ביותר בעיר הצעירה ביותר

if (!youngestCity.isEmpty()) {

Map<Character, Integer> charCountMap = new HashMap<>();

for (Costumer costumer : lines) {

if (costumer.getCityLiving().equals(youngestCity)) {

// תו ראשון בשם פרטי ושם משפחה

char firstChar = costumer.getFirstName().charAt(0);

char lastChar = costumer.getLastName().charAt(0);

// עדכון המונה עבור כל תו

charCountMap.put(firstChar, charCountMap.getOrDefault(firstChar, 0) + 1);

charCountMap.put(lastChar, charCountMap.getOrDefault(lastChar, 0) + 1);

}

}

// מציאת התו הנפוץ ביותר

char mostCommonChar = ' ';

maxCount = 0;

for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

mostCommonChar = entry.getKey();

}

}

// הדפסת התו הנפוץ ביותר

System.out.println("התו הנפוץ ביותר בעיר " + youngestCity + " הוא: " + mostCommonChar);

sendAnswer(MAGIC,"1","H");

} else {

System.out.println("לא נמצאו ערים.");

}

}

//Write your code here!

public static List<Costumer> readFile () {

List<Costumer> lines = new ArrayList<>();

try {

File file = new File("C:\\Users\\danie\\Downloads\\ws\_2024\_a\_helper-main (23)\\ws\_2024\_a\_helper-main\\data.csv");

if (file.exists()) {

Scanner scanner = new Scanner(file);

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

String[] tokens = line.split(",");

Costumer costumer = new Costumer(Integer.parseInt(tokens[0]), tokens[1],

tokens[2], tokens[3], Integer.parseInt(tokens[4]), tokens[5], tokens[6],

Double.parseDouble(tokens[7]));

lines.add(costumer);

}

}

} catch (FileNotFoundException e) {

throw new RuntimeException(e);

}

return lines;

}

public static void sendAnswer(String magic, String question, String answer) {

try (CloseableHttpClient client = HttpClients.createDefault()) {

URI uri = new URIBuilder(PATH)

.setParameter("magic", magic)

.setParameter("question",question)

.setParameter("answer", answer)

.build();

HttpPost post = new HttpPost(uri);

post.setHeader("Content-Type", "application/json");

try (CloseableHttpResponse response = client.execute(post)) {

int responseCode = response.getStatusLine().getStatusCode();

System.out.println("Response Code: " + responseCode);

String responseBody = EntityUtils.toString(response.getEntity());

System.out.println("Response: " + responseBody);

if (responseCode == 200) {

System.out.println("Answer sent successfully!");

} else {

System.out.println("Error code: " + responseCode);

}

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}  
  
  
שאלה 2:  
import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.utils.URIBuilder;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.util.EntityUtils;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.net.URI;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class Image {

public static final String PATH = "https://app.seker.live/fm1/answer-image";

public static final String MAGIC = "Hd5GB0C";

public static void main(String[] args) {

try {

// נתיב התמונה

File file = new File("C:\\Users\\danie\\Downloads\\ws\_2024\_a\_helper-main (18)\\ws\_2024\_a\_helper-main\\images\\14.png"); // שנה את הנתיב לתמונה שלך

BufferedImage image = ImageIO.read(file); // קריאת התמונה

// יצירת מפת ספירה עבור צבעי RGB

Map<Color, Integer> colorCountMap = new HashMap<>();

// עובר על כל הפיקסלים בתמונה

for (int y = 0; y < image.getHeight(); y++) {

for (int x = 0; x < image.getWidth(); x++) {

Color pixelColor = new Color(image.getRGB(x, y));

// מוודא שהתמונה אינה צבע לבן (אופציונלי, אפשר לשנות את זה או להסיר)

if (!(pixelColor.getRed() == 255 && pixelColor.getGreen() == 255 && pixelColor.getBlue() == 255)) {

// מוסיף את הצבע למפה ומעדכן את ספירתו

colorCountMap.put(pixelColor, colorCountMap.getOrDefault(pixelColor, 0) + 1);

}

}

}

// המרת המפה לרשימה כדי למיין אותה לפי הערכים (הכמות)

List<Map.Entry<Color, Integer>> sortedList = new ArrayList<>(colorCountMap.entrySet());

// מיון הרשימה לפי כמות ההופעות (הערך) בסדר יורד

sortedList.sort((entry1, entry2) -> entry2.getValue() - entry1.getValue());

// הדפסת הצבעים והכמויות בסדר יורד לפי הכמות

for (Map.Entry<Color, Integer> entry : sortedList) {

Color color = entry.getKey();

int count = entry.getValue();

System.out.println("צבע: RGB(" + color.getRed() + ", " + color.getGreen() + ", " + color.getBlue() + ") - כמות: " + count);

}

} catch (IOException e) {

e.printStackTrace();

}

sendAnswer(MAGIC,"193","50","210");

// לחשב את הצבע הכי פחות דומיננטי

}

public static void sendAnswer(String magic, String red, String green, String blue) {

try (CloseableHttpClient client = HttpClients.createDefault()) {

URI uri = new URIBuilder(PATH)

.setParameter("magic", magic)

.setParameter("red",red)

.setParameter("green", green)

.setParameter("blue",blue)

.build();

HttpPost post = new HttpPost(uri);

post.setHeader("Content-Type", "application/json");

try (CloseableHttpResponse response = client.execute(post)) {

int responseCode = response.getStatusLine().getStatusCode();

System.out.println("Response Code: " + responseCode);

String responseBody = EntityUtils.toString(response.getEntity());

System.out.println("Response: " + responseBody);

if (responseCode == 200) {

System.out.println("Answer sent successfully!");

} else {

System.out.println("Error code: " + responseCode);

}

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}

// נתיב התמונה