**IT ion** **Solutions**

**Assignment**

**Question 1.**

The Bitcoin rate is available as a public REST API that gives response in json format here:

https://api.coindesk.com/v1/bpi/currentprice.json Write a program that uses this REST API to

get the current rate of bitcoin and prints it in words. You can ignore the decimal part.

for example: if the rate is as follows:

"rate":"22,616.3987"

Your program should print: Twenty Two Thousand Six Hundred and Sixteen

**Solution:**

package ITionSolutions.BitcoinRateRestApi.src;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

import org.json.JSONObject;

public class BitcoinRateRestApi {

public static void main(String[] args) {

try {

String apiUrl = "https://api.coindesk.com/v1/bpi/currentprice.json";

String json = sendGET(apiUrl);

double rate = extractBitcoinRate(json);

String rateInWords = convertToWords(rate);

System.out.println(rateInWords);

} catch (IOException e) {

System.out.println("An error occurred while retrieving the Bitcoin rate.");

e.printStackTrace();

}

}

private static String sendGET(String apiUrl) throws IOException {

URL url = new URL(apiUrl);

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("GET");

int responseCode = connection.getResponseCode();

if (responseCode == HttpURLConnection.HTTP\_OK) {

BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));

String inputLine;

StringBuilder response = new StringBuilder();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

return response.toString();

} else {

throw new IOException("Failed to retrieve data from the API. Response code: " + responseCode);

}

}

private static double extractBitcoinRate(String json) {

JSONObject jsonObject = new JSONObject(json);

JSONObject bpi = jsonObject.getJSONObject("bpi");

JSONObject usd = bpi.getJSONObject("USD");

String rateString = usd.getString("rate");

return Double.parseDouble(rateString.replace(",", ""));

}

private static String convertToWords(double rate) {

int rateInInt = (int) rate;

String[] units = {

"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten",

"Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"

};

String[] tens = {

"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"

};

if (rateInInt == 0) {

return "Zero";

}

if (rateInInt < 0) {

return "Minus " + convertToWords(Math.abs(rateInInt));

}

String words = "";

if ((rateInInt / 1000) > 0) {

words += convertToWords(rateInInt / 1000) + " Thousand ";

rateInInt %= 1000;

}

if ((rateInInt / 100) > 0) {

words += convertToWords(rateInInt / 100) + " Hundred ";

rateInInt %= 100;

}

if (rateInInt > 0) {

if (rateInInt < 20) {

words += units[rateInInt];

} else {

words += tens[rateInInt / 10];

if ((rateInInt % 10) > 0) {

words += " " + units[rateInInt % 10];

}

}

}

return words;

}

}

**Output:**

**In Deferent‑ Deferent Time**

Thirty Thousand Six Hundred Forty Two

Thirty Thousand Six Hundred Twenty Four

Thirty Thousand Six Hundred Thirty Five

**Question 2.**

Given a matrix of characters representing a place on Earth, where the value 'T' indicates the presence of a Tree at that location and 'O' represents there is no tree at that point. An orchard is a region with tress connected vertically, horizontally, or diagonally. The size of the orchard is the total number of connected trees. Write a method to compute the sizes of all orchards in the matrix.

Example input:

[

['O','T','O','O'],

['O','T','O','T'],

['T','T','O','T'],

['O','T','O','T']

]

Note: Input can be in the code itself (it doesn't have to be supplied at runtime)

Output: 5, 3

If you are not able to get the exact output, still go ahead and upload the code. But ensure that your code is commented well with the comments explaining the approach.

**Solution:**

package ITionSolutions;

import java.util.ArrayList;

import java.util.List;

public class OrchardSizes {

public static void main(String[] args) {

char[][] matrix = {

{'O','T','O','O'},

{'O','T','O','T'},

{'T','T','O','T'},

{'O','T','O','T'}

};

List<Integer> orchardSizes = computeOrchardSizes(matrix);

for (int size : orchardSizes) {

System.out.print(size + " ");

}

}

public static List<Integer> computeOrchardSizes(char[][] matrix) {

List<Integer> orchardSizes = new ArrayList<>();

int rows = matrix.length;

int cols = matrix[0].length;

boolean[][] visited = new boolean[rows][cols];

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (matrix[i][j] == 'T' && !visited[i][j]) {

int orchardSize = dfs(matrix, i, j, visited);

orchardSizes.add(orchardSize);

}

}

}

return orchardSizes;

}

private static int dfs(char[][] matrix, int row, int col, boolean[][] visited) {

int rows = matrix.length;

int cols = matrix[0].length;

if (row < 0 || row >= rows || col < 0 || col >= cols || matrix[row][col] == 'O' || visited[row][col]) {

return 0;

}

visited[row][col] = true;

int size = 1;

// Check all neighboring cells (horizontal, vertical, diagonal)

int[] dr = {-1, -1, -1, 0, 0, 1, 1, 1};

int[] dc = {-1, 0, 1, -1, 1, -1, 0, 1};

for (int i = 0; i < 8; i++) {

int newRow = row + dr[i];

int newCol = col + dc[i];

size += dfs(matrix, newRow, newCol, visited);

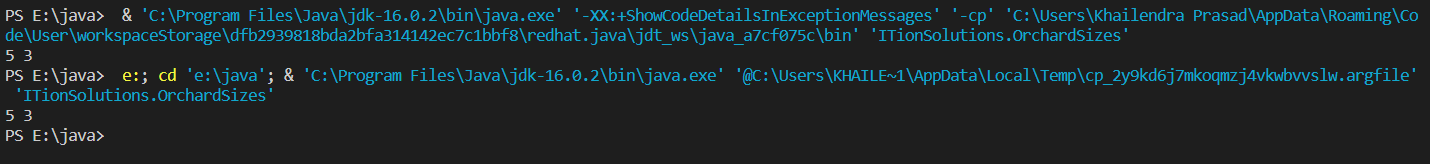
}

return size;

}

}

**Output:**



**GitHub Link: -** https://github.com/KumarKhailendra/ITionSolutionsAssignment