A picture containing logo

Description automatically generated

**MEHRAN UNIVERSITY OF ENGINEERING AND TECHNOLOGY, JAMSHORO**

**PROBLEM BASED LEARNING ASSIGMENT**

**( 21SW – III )**

**DATA STRUCTURES AND ALGORITHMS**

**BY**

**UZAIR HUSSAIN SHAIKH 21SW085-III**

**SUBMITTED TO:**

**SIR MOHSIN MEMON**

**Tasks**

**Dataset**: <https://www.kaggle.com/datasets/usgs/earthquake-database>

**Step** **1**: Use <https://www.daniel-braun.com/technik/reverse-geocoding-library-for-java/> library to find out the city and country from the given coordinates and store them in yearly earthquake collection along with magnitude. (Collection of each year means 52 collections)

**Step 2**: Make a queue storing biggest (with highest magnitude) quake of each year with magnitude and country, starting from 1965 to 2016. (52 elements in the queue approx.).

**Step 3**: Make a stack from the collections, one for each country which stores earthquake and its magnitude in the order of the event (the most recent event on top).

**Step 4**: Make a linked list which saves the one most recent earthquake with magnitude and country name from each country (use the stack from step 3).

**Problem 1**: How to find the average number of earthquakes per year for each country and which country is most vulnerable to earthquakes (which country has the most number of earth quakes)?

**Problem 2**: Which are the biggest earthquakes from 2005 to 2015 and occurred and in which country (use step 2)?

**Problem 3**: How to determine the recent 5 earthquakes from each country?

**Problem 4**: How to find the most recent above 6 magnitude earthquakes (use step 4)

**Algorithms**

**Main Class:**

import java.io.IOException;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws IOException {

        Scanner s = new Scanner(System.in);

        Collections list = new Collections();       // object of collection class ( step 1 )

        LinkedQueue queue= new LinkedQueue();       // object of linked queue class ( step 2 )

        Stacks stack = new Stacks();               // object of stacks class ( step 3 & 4 )

        int ch=0;

        do{

            ch=list.menu();          // calls menu method from collection class

            switch(ch){

                case 1:

                    list.display\_Collections();    // method to display collection yearly

                    break;

                case 2:

                    queue.display\_All();         //  method to display collection

                    break;

                case 3:

                    stack.display\_Yearly\_Stacks();   // display country stacks country wise

                    break;

                case 4:

                    stack.recent\_stacks();         //  display stacks recent ones

                    break;

                case 5:

                    stack.average\_earthquake();    // returns most vulnerable countries

                    break;

                case 6:

                    queue.displayFrom\_05\_15();       // shows earthquakes of 2005 to 2015

                    break;

                case 7:

                    stack.display\_recent\_5();      // displays recent five from each stacks

                    break;

                case 8:

                    stack.recent\_above\_6();    // display recenet earthquakes with magnitude greater than 6

                    break;

                case 0:

                    ch=0;

                    break;

                default: System.out.println("\nInvalid Choice");

            }

            if(ch==0){

                System.out.println("Thank You!");

                break;

            }

            System.out.print("\nPress any no. to continue \n      or 0 to exit \n\nChoice:");

            ch=s.nextInt();

        }while(ch!=0);

    }

}

**Step 1:**

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;

import com.opencsv.CSVReader;

import com.opencsv.exceptions.CsvValidationException;

class LinkedNode{           // node class use in collections

    LinkedNode next;

    int year;

    String country,state;

    float magnitude;

    Object date;

    Object time;

    LinkedNode(){

    }

    // constructor to add data

    public LinkedNode(int year, String country,String state, Object date, float magnitude, Object time){

        this.year=year;

        this.country=country;

        this.magnitude=magnitude;

        this.state=state;

        this.time=time;

        this.date=date;

    }

    //method to display element in nodes

    public String display(){

        return year +" , "+country + " , " + state+" , "+ date + " , " + magnitude + " , " + time;

    }

}

class Collections {

    ArrayList<LinkedNode> earthquake = new ArrayList<>();       // collection to store earthquakes yearly

    // arrays to store data from columns and parsing

    float[] highestmagnitudes;

    String[] country;

    int[] years;

    // contructor of collection class to initialize process by just calling

    Collections() throws IOException{

    int Year;

    int count=0;

    String filepath= "E:\\2. Muet material\\3. 3rd Sem SW\\1. DSA\\4. DSA\_PBL\_SEM\_ASSIGNMENT\\PBL\_21SW085\\PBL\_21SW85\\Cleaned\_Data.csv";

    String line;

    String splitBy = ",";

    String add;

    float magnitude;

    // Read the file and extract the year from the date in column 1

        try {

            BufferedReader br = new BufferedReader(new FileReader(filepath));

            br.readLine();      // reada header line

            while ((line = br.readLine()) != null) // returns a Boolean value

            {

                add="";

                String[] parts = line.split(splitBy); // use comma as separator

                magnitude= Float.parseFloat(parts[4]);  // reads magnitude column

                Year = Integer.parseInt(parts[0].substring(parts[0].length()-4));   //reads year from date column

                // add data in collection

                if(earthquake.size()!=0){

                    for(int i=0;i<earthquake.size();i++){

                        if(Year==(earthquake.get(i).year)){

                            LinkedNode temp = earthquake.get(i);

                            while(temp.next!=null){

                                temp=temp.next;

                            }

                            temp.next= new LinkedNode(Year, parts[2], parts[3], parts[0], magnitude, parts[0]);

                            continue;

                        }

                        else if(i==earthquake.size()-1){

                            LinkedNode temp= new LinkedNode(Year, parts[2], parts[3], parts[0], magnitude, parts[0]);

                            earthquake.add(temp);

                            break;

                        }

                    }

                }

                else if(earthquake.size()==0){

                    LinkedNode temp = new LinkedNode(Year, parts[2], parts[3], parts[0], magnitude, parts[0]);

                    earthquake.add(temp);

                }

            }

        br.close();

        count++;

        } catch (FileNotFoundException e) {

            System.out.println("File not found: ");

            e.printStackTrace();

        }

    }

    void display\_Collections(){

        int Year;

        Scanner s=new Scanner(System.in);

        System.out.print("Enter Year :  ");

        Year=s.nextInt()-1965;

        LinkedNode n=earthquake.get(Year);

        int x=1;

        while(n!=null){

            System.out.println(x + " : " + n.time + " : " + n.year + " : " + n.country + " : " + n.state + " : " + n.magnitude);

            n.display();

            n=n.next;

            x++;

        }

    }

    void display\_All\_Collections(){

        for(int i=0;i<52;i++){     //  by using line 107, 108 we can see each year collection with thier count

        LinkedNode n=earthquake.get(i);

        int x=1;

        while(n!=null){

            System.out.println(x + " : " + n.time + " : " + n.year + " : " + n.country + " : " + n.state + " : " + n.magnitude);

            n.display();

            n=n.next;

            x++;

        }

    }

    }

    public int menu(){

        Scanner s = new Scanner(System.in);

        System.out.println("\n\tPBL Tasks ");

        System.out.println("1. Step 1 (Yearly Collections of Earthquakes.) ");

        System.out.println("2. Step 2 (Yearly Queue of Earthquakes.) ");

        System.out.println("3. Step 3 (Country Stacks of Earthquakes.) ");

        System.out.println("4. Step 4 (Recent Earthquake of each year.) ");

        System.out.println("5. Problem 1 (Most Vulnerable Country.) ");

        System.out.println("6: Problem 2 (Biggest Earthquakes from 2005 to 2015.) ");

        System.out.println("7: Problem 3 (Recent 5 Earthquakes.) ");

        System.out.println("8: Problem 4 (Most recent Earthquakes with magnitude above 6.) ");

        System.out.println("0. Exit");

        System.out.print("\nEnter Your Choice: ");

        int Choice=s.nextInt();

        return Choice;

    }

    void highest() {

        int j=0;

        highestmagnitudes = new float[52]; // Array to store highest magnitudes

        country = new String[52];

        years = new int[52];

        for(int i = 1965; i <=2016; i++) {

            highestmagnitudes[i-1965] = 0; // Initialize to smallest possible float value

        }

        // Loop through each year and find the highest magnitude for that year

        for(int i = 0; i < earthquake.size(); i++) {

            LinkedNode n = earthquake.get(i);

            while(n != null) {

                float high=highestmagnitudes[n.year-1965];

                if(n.magnitude > highestmagnitudes[n.year - 1965]) { // Update highest magnitude for the year

                    highestmagnitudes[n.year - 1965] = Math.max(n.magnitude, high);

                    country[i]=(String)n.country;

                    years[i]= n.year;

                }

                n = n.next;

            }

        }

        for(int i = 0; i < 52; i++,j++) {

            LinkedNode n = earthquake.get(i);

            System.out.println(years[i]  + " : " + country[i] + " : " + highestmagnitudes[i]);

        }

        System.out.println(j);

    }

    int size(){

        return earthquake.size();

    }

    LinkedNode get(int index){

        return earthquake.get(index);

    }

}

**Step 2:**

import java.io.IOException;

interface Queue{

    public void addQueue(int year, String country, float magnitude);

    public int size();

}

class LinkedQueue implements Queue{

    private DataNode head= new DataNode(0, null, 0);

    private int Queuesize;

    private static class DataNode{  // inner node class to add data

        int year;

        String country;

        float magnitude;

        DataNode prev=this;

        DataNode next=this;

        public DataNode(int year,String country,float magnitude){

            this.year=year;

            this.country=country;

            this.magnitude=magnitude;

        }

        // constructor to add data

        public DataNode(int year,String country,float magnitude,DataNode n, DataNode p){

            this.year=year;

            this.country=country;

            this.magnitude=magnitude;

            next=n;

            prev=p;

        }

        // method to display data

        public void display(){

            System.out.println( year +" : "+country + " : " + magnitude);

        }

    }

    LinkedQueue(){

    }

    public int size(){

        return Queuesize;

    }

    public void addQueue(int year, String country,float magnitude) {

        head.prev.next = new DataNode(year, country, magnitude, head, head.prev);

        head.prev = head.prev.next;

        ++Queuesize;

    }

    public boolean isEmpty(){

        return (head==null);

    }

    // returns higest all yearly

    void  highest() throws IOException {

        int num=1;

        LinkedQueue q=new LinkedQueue();

        int year=0;

        float highest=0;

        String Country="";

        Collections earthquake = new Collections();

        for(int i = 0; i < earthquake.size(); i++) {

            LinkedNode n = earthquake.get(i);

            highest=0;

            while(n != null) {

                if(n.magnitude > highest) { // Update highest magnitude for the year

                    highest=n.magnitude;

                    Country=n.country;

                    year=n.year;

                }

                n = n.next;

            }

            q.addQueue(year, Country, highest);

        }

        for(DataNode t=q.head.next;t!=q.head;t=t.next){

            System.out.print(num+ " : ");

            t.display();

            num++;

        }

    }

    // returns highest from 2005 to 2016

    void  highest2() throws IOException {

        int num=1;

        LinkedQueue q=new LinkedQueue();

        int year=0;

        float highest=0;

        String Country="";

        Collections earthquake = new Collections();

        for(int i = 0; i < earthquake.size(); i++) {

            LinkedNode n = earthquake.get(i);

            highest=0;

            while(n != null) {

                if(n.magnitude > highest) { // Update highest magnitude for the year

                    highest=n.magnitude;

                    Country=n.country;

                    year=n.year;

                }

                n = n.next;

            }

            q.addQueue(year, Country, highest);

        }

        for(DataNode t=q.head.next;t!=q.head;t=t.next){

            if(t.year>2004 && t.year<2016){

                System.out.print(num+ " : ");

                t.display();

                num++;

            }

        }

    }

    // method to display earthquake all yearly

    void display\_All() {

        try {

            highest();

        } catch (Exception e) {

            // TODO: handle exception

        }

    }

    /// Problem # 02

    // method to display earthquake from 2005 to 2015

    public void displayFrom\_05\_15() {

        try {

            highest2();

        } catch (Exception e) {

            // TODO: handle exception

        }

    }

}

**Step 3:**

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

import java.util.Stack;

class StackNode{        // node to store stacks data

    int year;

    int size;

    String country;

    float magnitude;

    StackNode prev=null;

    StackNode next=null;

    StackNode top=null;

    StackNode(){

    }

    StackNode(int year,String country,float magnitude){     // constructor

        this.year=year;

        this.country=country;

        this.magnitude=magnitude;

    }

    StackNode(int year,float magnitude){

        this.year=year;

        this.magnitude=magnitude;

    }

    public void push1(int year2, String country2, float magnitude2) {

        this.year=year2;

        this.country=country2;

        this.magnitude=magnitude2;

    }

    // displays elements in node

    public void display3() {

        System.out.println( year +" : "+ country+" : "+ magnitude);

    }

}

class Stacks{

    StackNode head;

    ArrayList<Stacks> countries= new ArrayList<>();         // arraylist of stacks to store country wise earthquakes

    int size;

    public void push2(int year,String Country, float magnitude) {

        StackNode temp = new StackNode( year, Country, magnitude);

        temp.next=head;

        head = temp;

        size++;

    }

    public void peek(){

        StackNode n=head;

        while(n!=null){

            System.out.println(n.country+ " : " + n.year + " : " + n.magnitude);

            n=n.next;

        }

    }

    boolean isEmpty(){

        return (head==null);

    }

    public void CountryStacks() throws IOException{

        int Year;

        String country;

        String filepath= "E:\\2. Muet material\\3. 3rd Sem SW\\1. DSA\\4. DSA\_PBL\_SEM\_ASSIGNMENT\\PBL\_21SW085\\PBL\_21SW85\\Cleaned\_Data.csv";

        String line;

        String splitBy = ",";

        String add;

        float magnitude;

        // Read the file and extract the year from the date in column 1

        try {

            BufferedReader br = new BufferedReader(new FileReader(filepath));

            br.readLine();      // reada header line

            while ((line = br.readLine()) != null) // returns a Boolean value

            {

                add="";

                String[] parts = line.split(splitBy); // use comma as separator

                magnitude= Float.parseFloat(parts[4]);

                Year = Integer.parseInt(parts[0].substring(parts[0].length()-4));

                country= parts[2];

                if(countries.size()!=0){

                    for(int i=0;i<countries.size();i++){

                        if(countries.get(i).head.country.equals(country)){

                            countries.get(i).push2(Year, country, magnitude);

                            break;

                        }

                        else if (i==countries.size()-1){

                            Stacks temp = new Stacks();

                            temp.push2(Year, country, magnitude);

                            countries.add(temp);

                            break;

                        }

                    }

                    continue;

                }

                else{

                    Stacks temp=new Stacks();

                    temp.push2(Year, country, magnitude);

                    countries.add(temp);

                    continue;

                }

            }

        br.close();

        } catch (FileNotFoundException e) {

            System.out.println("File not found: ");

            e.printStackTrace();

        }

    }

    // method to print stacks recent ones first but all

    void recent\_stacks(){

        try {

            CountryStacks();

        } catch (Exception e) {

            // TODO: handle exception

        }

        StackNode t=countries.get(0).head;

        StackNode r=new StackNode(t.year, t.country, t.magnitude);

        StackNode temp=r;

        for(int i=1;i<countries.size();i++){

            StackNode x=countries.get(i).head;

            temp.next=new StackNode(x.year, x.country, x.magnitude);

            temp=temp.next;

        }

        for(StackNode z=r;z!=null;z=z.next){

            z.display3();

        }

    }

    // method to print stacks yearly but recent one first

    void display\_Yearly\_Stacks(){

        int num=0;

        try {

            CountryStacks();

        } catch (Exception e) {

            // TODO: handle exception

        }

        StackNode t=countries.get(0).head;

        StackNode r=new StackNode(t.year, t.country, t.magnitude);

        StackNode temp=r;

        for(int i=1;i<countries.size();i++){

            StackNode x=countries.get(i).head;

            temp.next=new StackNode(x.year, x.country, x.magnitude);

            temp=temp.next;

        }

        for(int i=0;i<countries.size();i++){

            StackNode a = countries.get(i).head;

            for(num=0;num<countries.size();num++){

                a.display3();

                a=a.next;

                if(a==null){

                   break;

                }

            }

        }

    }

    // method to print stacks recent but having magnitude above 6

    void recent\_above\_6(){

        int num=1;

        try {

            CountryStacks();

        } catch (Exception e) {

            // TODO: handle exception

        }

        StackNode t=countries.get(0).head;

        StackNode r=new StackNode(t.year, t.country, t.magnitude);

        StackNode temp=r;

        for(int i=1;i<countries.size();i++){

            StackNode x=countries.get(i).head;

            temp.next=new StackNode(x.year, x.country, x.magnitude);

            temp=temp.next;

        }

        System.out.println("\n\tPrinting magnitudes from country stacks having magnitude above 6.\n");

        for(StackNode z=r;z!=null;z=z.next){

            if(z.magnitude>6 && z.next!=null){

                System.out.print(num+ " : ");

                z.display3();

                num++;

            }

        }

    }

    // method to vulnerablitlity of coutnries

    void average\_earthquake(){

        int count=0;

        String country\_vulnerable = "";

        float result=0;

        try {

            CountryStacks();

        } catch (Exception e) {

            // TODO: handle exception

        }

        StackNode t=countries.get(0).head;

        StackNode r=new StackNode(t.year, t.country, t.magnitude);

        StackNode temp=r;

        for(int i=1;i<countries.size();i++){

            StackNode x=countries.get(i).head;

            // x.display3();

            temp.next=new StackNode(x.year, x.country, x.magnitude);

            temp=temp.next;

        }

        for(int i=0;i<countries.size();i++){

            count=0;

            // System.out.println(countries.get(i).size);

            StackNode a = countries.get(i).head;

            country\_vulnerable=a.country;

            a=a.next;

            count=countries.get(i).size;

            if(a==null){

                break;

            }

            result=(float)(count/52);

            System.out.println(country\_vulnerable  + " : "+count + " : " + result );

        }

    }

    // method to print 5 recent stacks of each country

    void display\_recent\_5(){

        int num=1;

        try {

            CountryStacks();

        } catch (Exception e) {

            // TODO: handle exception

        }

        StackNode t=countries.get(0).head;

        StackNode r=new StackNode(t.year, t.country, t.magnitude);

        StackNode temp=r;

        for(int i=1;i<countries.size();i++){

            StackNode x=countries.get(i).head;

            // x.display3();

            temp.next=new StackNode(x.year, x.country, x.magnitude);

            temp=temp.next;

        }

        for(int i=0;i<countries.size();i++){

            StackNode a = countries.get(i).head;

            for(num=0;num<6;num++){

                a.display3();

                a=a.next;

                if(a==null){

                   break;

                }

            }

        }

    }

}