**­­­­­­­­Assignment-2**

**Student ID: 1098648 Student Name: Wang Xinyu Section: W01**

A. How to use MyPriorityQueue to implement a movie rating system to help user select the top movie based on the rating, popularity, or release time.

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Collections;

class MyPriorityQueue<E extends Comparable<E>> {

*private* ArrayList<E> list = *new* ArrayList<>();

*public* void enqueue(E newObject) {

list.add(newObject);

Collections.sort(this.list);

}

*public* E dequeue() {

E removed = (E) list.get(list.size() - 1);

list.remove(list.size() - 1);

*return* removed;

}

*public* int getSize() {

*return* list.size();

}

}

*public* class Playground {

*public* *static* void main(String[] args) {

Movie movie1 = *new* Movie("A", 2.0, 3.0, 2011, 2, 1);

Movie movie2 = *new* Movie("B", 3.0, 2.0, 2011, 2, 1);

Movie movie3 = *new* Movie("C", 4.0, 1.0, 2011, 1, 1);

Movie movie4 = *new* Movie("D", 5.0, 0.0, 2012, 1, 1);

Movie movie5 = *new* Movie("E", 5.0, 1.0, 2012, 1, 1);

Movie movie6 = *new* Movie("F", 5.0, 0.0, 2012, 2, 1);

Movie movie7 = *new* Movie("G", 5.0, 1.0, 2012, 2, 1);

MyPriorityQueue<Movie> priorityQueue = *new* MyPriorityQueue();

priorityQueue.enqueue(movie1);

priorityQueue.enqueue(movie2);

priorityQueue.enqueue(movie3);

priorityQueue.enqueue(movie4);

priorityQueue.enqueue(movie5);

priorityQueue.enqueue(movie6);

priorityQueue.enqueue(movie7);

*while* (priorityQueue.getSize() > 0) {

System.out.println(priorityQueue.dequeue());

}

}

*static* class Movie implements Comparable<Movie> {

String name;

double rate;

double popularity;

int releaseYear;

int releaseMonth;

int releaseDay;

*public* Movie(

String name,

double rate,

double popularity,

int releaseYear,

int releaseMonth,

int releaseDay

) {

this.name = name;

this.rate = rate;

this.popularity = popularity;

this.releaseYear = releaseYear;

this.releaseMonth = releaseMonth;

this.releaseDay = releaseDay;

}

@Override

*public* String toString() {

*return* String.format("%s rate: %.1f, pop: %.1f, %d-%d-%d", name, rate, popularity, releaseYear, releaseMonth, releaseDay);

}

@Override

*public* int compareTo(Movie movie) {

int ret = 0;

ret += 10000 \* Double.compare(this.rate, movie.rate);

ret += 1000 \* Double.compare(this.popularity, movie.popularity);

ret += 100 \* Integer.compare(this.releaseYear, movie.releaseYear);

ret += 10 \* Integer.compare(this.releaseMonth, movie.releaseMonth);

ret += 1 \* Integer.compare(this.releaseDay, movie.releaseDay);

*return* ret;

}

}

}

Terminal Output:

Text

Description automatically generated

B. Write a program that (1) counts the occurrences of words in a text and displays the words and their occurrences in ascending order of the words. Must use your implementation of MyHashedSet and MyHashedMap. (2) Count the occurrence of the keywords in a Java source file. Store Java keywords in MyHashSet. Write a program that uses MyHashMap to store a pair consisting of a word and its count. For each word, check whether it is already a key in the map. If not, add the key and value 1 to the map. Otherwise, increase the value for the word (key) by 1 in the map. To sort the map, convert it to a tree map. Note: keywords inside a String are not counted.

import java.io.\*;

import java.util.\*;

class MyHashSet<E extends Comparable<E>> {

*private* Set<E> set = *new* HashSet<>();

MyHashSet(E[] objects) {

this.set = *new* HashSet<>(Arrays.asList(objects));

}

boolean contains(E element) {

*return* set.contains(element);

}

Object[] elements() {

*return* set.toArray();

}

}

class MyHashMap<E extends Comparable<E>, F extends Comparable<F>> {

*private* Map<E, F> map = *new* HashMap<>();

boolean containsKey(E key) {

*return* map.containsKey((Object) key);

}

boolean containsValue(F value) {

*return* map.containsValue(value);

}

void updateKeyValuePair(E key, F value) {

map.replace(key, value);

}

void addKeyValuePair(E key, F value) {

map.put(key, value);

}

F get(E key) {

*return* map.get(key);

}

Set<Map.Entry<E, F>> entrySet() {

*return* this.map.entrySet();

}

}

class WordCounter {

*private* MyHashMap<String, Integer> map;

*private* *static* MyHashSet<Character> ignoredCharacters = *new* MyHashSet<>(toCharacterArray(";.,{}()/ "));

*private* *static* Character[] toCharacterArray(String s) {

*if* ( s == null )

*return* null;

int len = s.length();

Character[] array = *new* Character[len];

*for* (int i = 0; i < len ; i++)

array[i] = s.charAt(i);

*return* array;

}

*private* *static* String stringFormat(String str) {

StringBuilder stringBuilder = *new* StringBuilder();

*for* (int i = 0; i < str.length(); i++) {

*final* char currentCharacter = str.charAt(i);

*if* (!ignoredCharacters.contains(currentCharacter)) {

stringBuilder.append(currentCharacter);

} *else* {

stringBuilder.append(" ");

}

}

System.out.println(stringBuilder.toString());

*return* stringBuilder.toString();

}

WordCounter(String str) {

MyHashMap<String, Integer> tMap = *new* MyHashMap<>();

String[] data = stringFormat(str).split(" ");

*for* (String element*:* data)

*if* (tMap.containsKey(element))

tMap.updateKeyValuePair(element, map.get(element) + 1);

*else*

tMap.addKeyValuePair(element, 1);

this.map = tMap;

}

WordCounter(MyHashSet<String> preset) {

this.map = *new* MyHashMap<>();

*for* (Object element*:* preset.elements())

this.map.addKeyValuePair((String) element, 0);

}

*public* MyHashMap<String, Integer> getMap() {

*return* map;

}

*public* void process(String str) {

String[] data = stringFormat(str).split(" ");

*for* (String element*:* data)

*if* (map.containsKey(element))

map.updateKeyValuePair(element, map.get(element) + 1);

}

*public* void processFromFileOnPath(String path) {

*try* {

FileReader file = *new* FileReader(path);

Scanner input = *new* Scanner(file);

StringBuilder stringBuilder = *new* StringBuilder();

*while* (input.hasNextLine())

stringBuilder.append(input.nextLine());

input.close();

this.process(stringBuilder.toString());

} *catch* (FileNotFoundException e) {

e.printStackTrace();

}

}

*public* void printInTerminal() {

System.out.println();

*for* (Map.Entry<String, Integer> entry *:* this.map.entrySet()) {

System.out.printf("%s: %d; ", entry.getKey(), entry.getValue());

}

System.out.println();

}

}

*public* class Playground {

*public* *static* void main(String[] args) {

String string = "Hello World, \"this\" is Wendell.";

WordCounter counter = *new* WordCounter(string);

counter.printInTerminal();

*final* String[] JavaKeywords = {

"abstract", "assert", "boolean",

"break", "byte", "case", "catch", "char", "class", "const",

"continue", "default", "do", "double", "else", "enum",

"extends", "for", "final", "finally", "float", "goto",

"if", "implements", "import", "instanceof", "int",

"interface", "long", "native", "new", "package", "private",

"protected", "public", "return", "short", "static",

"strictfp", "super", "switch", "synchronized", "this",

"throw", "throws", "transient", "try", "void", "volatile",

"while", "true", "false", "null"

};

MyHashSet<String> myHashedSet = *new* MyHashSet<>(JavaKeywords);

WordCounter keywordCounter = *new* WordCounter(myHashedSet);

Scanner scanner = *new* Scanner(System.in);

System.out.print("Enter the file name or path: ");

String currentFilePath = scanner.nextLine();

scanner.close();

keywordCounter.processFromFileOnPath(currentFilePath);

keywordCounter.printInTerminal();

}

}

Terminal Output of the java code above:

