/\*\*

\* Roberto Andino

\* Version 1

\* Program is done in Java programming language

\*

\* This is the Testerclass with the main method, It has all the search and sort

\* methods to display the data according to directions.

\*

\*/

package tvshow;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

public class TvShowTester {

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

//filename

String fileName = "tv\_shows.csv";

String line = "";

//tvShow array attributes

String name;

String yearPremier;

String seasons;

String episodes;

String network;

String genre;

String maleLead;

String femaleLead;

int count = 0; //for arrayLocaton

int count2 = 0; //to skip first row in file

//tvShow array

tvShow[] shows = new tvShow[20];

//reading file

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

//storing values in array of values and creating tvShow objects

while((line = br.readLine()) != null)

{

//storing data in array

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

//storing array data in variables

name = values[0];

yearPremier = values[1];

seasons = values[2];

episodes = values[3];

network = values[4];

genre = values[5];

maleLead = values[6];

femaleLead = values[7];

//if statement to skip first row in file

if(count2 > 0)

{

//creating tvShow objects

shows[count] = new tvShow(name, yearPremier, seasons, episodes,

network, genre, maleLead, femaleLead);

count++; //for next array

}

count2++;

}

System.out.println("\nSorted by TV Show name with insertion sort - ascending: \n"

+ "--------------------------------");

shows = insertionSorts(shows, 1);

printTvShows(shows, 1);

System.out.println("\n\nSorted by number of seasons with insertion sort - descending: \n"

+ "--------------------------------");

shows = insertionSorts(shows, 2);

printTvShows(shows, 2);

System.out.println("\n\nSorted by year show premiered with selection sort - ascending: \n"

+ "--------------------------------");

selectionSorts(shows, 1);

printTvShows(shows, 2);

System.out.println("\n\nSorted by genre with selection sort - descending: \n"

+ "--------------------------------");

selectionSorts(shows, 2);

printTvShows(shows, 3);

System.out.println("\n\nSorted by network with merge sort - descending: \n"

+ "--------------------------------");

mergeSort1(shows, 0, shows.length-1 );

printTvShows(shows, 4);

System.out.println("\n\nSorted by number of episodes with merge sort - ascending: \n"

+ "--------------------------------");

mergeSort2(shows, 0, shows.length-1 );

printTvShows(shows, 5);

System.out.println("\n\nLinear search to find shows that lasted 2 years \n"

+ "--------------------------------");

linearSearch(shows, "2", 1);

System.out.println("\n\nLinear search to find shows that had more than 100"

+ " episodes\n --------------------------------");

linearSearch(shows, "100", 2);

System.out.println("\n\nBinary search to find and cound all shows in the"

+ " array that aired on CBS Network \n--------------------------------");

//sorting array by network name for binary search - ascending order

shows = insertionSorts(shows, 3);

//getting index location of show aired on CBS from binary search

int index = binarySearch(shows, "CBS", 1);

//calling method matchAndPrint to match the result returned by the

//binary search and check if there are other shows in same network

matchAndPrint(shows, index, 1);

System.out.println("\n\nBinary search to find and cound all shows in the"

+ " array that are on the Drama Genre \n--------------------------------");

//sorting array by genre name for binary search - ascending order

selectionSorts(shows, 3);

//getting index of in the Drama genre from binary search

int index2 = binarySearch(shows, "Drama", 2);

//calling method matchAndPrint to match the result returned by the

//binary search and check if there are other shows in same genre

matchAndPrint(shows, index2, 2);

System.out.println("\n\nBinary search to find and cound all shows in the"

+ " array that are on the Fanatasy Genre \n--------------------------------");

//sorting array by genre name for binayr search - ascending

mergeSort3(shows, 0, shows.length-1 );

//getting index of in the Fantasy genre from binary search

int index3 = binarySearch(shows, "Fantasy", 2);

//System.out.println("index2 : " + index3);

//calling method matchAndPrint to match the result returned by the

//binary search and check if there are other shows in same genre

matchAndPrint(shows, index3, 2);

}

public static void printTvShows(tvShow[] shows, int order)

{

//switch statatement to print message according to directions

switch (order) {

case 1:

System.out.println("Name \t\t\t Year premiered \t\t Male Lead \t\t Female Lead \n");

break;

case 2:

System.out.println("Name \t\t\t Year premiered \t\t Number of seasons \n");

break;

case 3:

System.out.println("Name \t\t\t Number of episodes \t\t Genre \n");

break;

case 4:

System.out.println("Name \t\t\t Year premiered \t\t Number of seasons "

+ "\t\t Network\n");

break;

case 5:

System.out.println("Name \t\t\t Year premiered \t\t Number of episodes "

+ "\t\t Network\n");

break;

default:

break;

}

//switch statement to print array data according to directions

switch (order) {

case 1:

for (tvShow show : shows) {

System.out.printf("%-20s%9s%38s%28s%n", show.getName(),

show.getYearPremier(), show.getMaleLead(),

show.getFemaleLead() );

}

break;

case 2:

for (tvShow show : shows) {

System.out.printf("%-20s%9s%30s%n", show.getName(),

show.getYearPremier(), show.getSeasons());

}

break;

case 3:

for (tvShow show : shows) {

System.out.printf("%-20s%9s%35s%n", show.getName(),

show.getEpisodes(), show.getGenre());

}

break;

case 4:

for (tvShow show : shows) {

System.out.printf("%-20s%9s%30s%33s%n", show.getName(),

show.getYearPremier(), show.getSeasons(), show.getNetwork());

}

break;

case 5:

for (tvShow show : shows) {

System.out.printf("%-20s%9s%30s%35s%n", show.getName(),

show.getYearPremier(), show.getEpisodes(), show.getNetwork());

}

break;

default:

break;

}

}

public static tvShow[] insertionSorts(tvShow[] source, int order)

{

tvShow[] dest = new tvShow[ source.length ];

switch (order) {

//ascending order by name

case 1:

for ( int i = 0 ; i < source.length ; i++ )

{

tvShow next = source[ i ];

int insertindex = 0;

int k = i;

while ( k > 0 && insertindex == 0 )

{

if ( next.getName().compareTo(dest[ k - 1 ].getName()) > 0 )

{

insertindex = k;

}

else

{

dest[ k ] = dest[ k - 1 ];

}

k--;

}

dest[ insertindex ] = next;

} // end of for

break;

//descending order by number of seasons

case 2:

for ( int i = 0 ; i < source.length ; i++ )

{

tvShow next = source[ i ];

int insertindex = 0;

int k = i;

while ( k > 0 && insertindex == 0 )

{

if ( next.getSeasons().compareTo(dest[ k - 1 ].getSeasons()) < 0)

{

insertindex = k;

}

else

{

dest[ k ] = dest[ k - 1 ];

}

k--;

}

dest[ insertindex ] = next;

} // end of for

break;

//ascending order by Network

case 3:

for ( int i = 0 ; i < source.length ; i++ )

{

tvShow next = source[ i ];

int insertindex = 0;

int k = i;

while ( k > 0 && insertindex == 0 )

{

if ( next.getNetwork().compareTo(dest[ k - 1 ].getNetwork()) > 0 )

{

insertindex = k;

}

else

{

dest[ k ] = dest[ k - 1 ];

}

k--;

}

dest[ insertindex ] = next;

} // end of for

break;

default:

break;

}

return dest;

}

public static void selectionSorts(tvShow[] source, int order)

{

int i;

int k;

int posmax;

tvShow temp;

switch (order) {

//ascending by year premiered

case 1:

for ( i = source.length - 1 ; i >= 0 ; i-- )

{

// find largest element in the i elements

posmax = 0;

for ( k = 0 ; k <= i ; k++ )

{

if ( source[ k ].getYearPremier().compareTo(source[ posmax ].getYearPremier()) > 0 )

posmax = k;

}

// swap the largest with the position i

// now the item is in its proper location

temp = source[ i ];

source[ i ] = source[posmax ];

source[ posmax ] = temp;

} break;

//descending by genre

case 2:

for ( i = source.length - 1 ; i >= 0 ; i-- )

{

// find largest element in the i elements

posmax = 0;

for ( k = 0 ; k <= i ; k++ )

{

if ( source[ k ].getGenre().compareTo(source[ posmax ].getGenre()) < 0 )

posmax = k;

}

// swap the largest with the position i

// now the item is in its proper location

temp = source[ i ];

source[ i ] = source[posmax ];

source[ posmax ] = temp;

} break;

//ascending by genre

case 3:

for ( i = source.length - 1 ; i >= 0 ; i-- )

{

// find largest element in the i elements

posmax = 0;

for ( k = 0 ; k <= i ; k++ )

{

if ( source[ k ].getGenre().compareTo(source[ posmax ].getGenre()) > 0 )

posmax = k;

}

// swap the largest with the position i

// now the item is in its proper location

temp = source[ i ];

source[ i ] = source[posmax ];

source[ posmax ] = temp;

} break;

default:

break;

}

}

//descending merge sort by genre

public static void mergeSort1(tvShow[] a, int low, int high)

{

if ( low == high )

return;

int mid = ( low + high ) / 2;

mergeSort1( a, low, mid );

mergeSort1( a, mid + 1, high);

mergeNetwork( a, low, mid, high);

}

//descending merge sort by genre

public static void mergeNetwork( tvShow[] a, int low, int mid, int high )

{

tvShow[] temp = new tvShow[ high - low + 1 ];

int i = low, j = mid + 1, n = 0;

while ( i <= mid || j <= high )

{

if ( i > mid )

{

temp[ n ] = a[ j ];

j++;

}

else if ( j > high )

{

temp[ n ] = a[ i ];

i++;

}

else if ( a[ i ].getNetwork().compareTo(a[ j ].getNetwork()) > 0)

{

temp[ n ] = a[ i ];

i++;

}

else

{

temp[ n ] = a[ j ];

j++;

}

n++;

}

for ( int k = low ; k <= high ; k++ )

a[ k ] = temp[ k - low ];

} // end of merge

//ascending merge sort by number of episodes

public static void mergeSort2(tvShow[] a, int low, int high)

{

if ( low == high )

return;

int mid = ( low + high ) / 2;

mergeSort2( a, low, mid );

mergeSort2( a, mid + 1, high);

mergeEpisodes( a, low, mid, high);

}

//ascending merge sort by number of episodes

public static void mergeEpisodes( tvShow[] a, int low, int mid, int high )

{

tvShow[] temp = new tvShow[ high - low + 1 ];

int i = low, j = mid + 1, n = 0;

while ( i <= mid || j <= high )

{

if ( i > mid )

{

temp[ n ] = a[ j ];

j++;

}

else if ( j > high )

{

temp[ n ] = a[ i ];

i++;

}

else if ( a[ i ].getEpisodes().compareTo(a[ j ].getEpisodes()) < 0 )

{

temp[ n ] = a[ i ];

i++;

}

else

{

temp[ n ] = a[ j ];

j++;

}

n++;

}

for ( int k = low ; k <= high ; k++ )

a[ k ] = temp[ k - low ];

} // end of merge

//ascending merge sort by Genre

public static void mergeSort3(tvShow[] a, int low, int high)

{

if ( low == high )

return;

int mid = ( low + high ) / 2;

mergeSort3( a, low, mid );

mergeSort3( a, mid + 1, high);

mergeGenre( a, low, mid, high);

}

//ascending merge sort by Genre

public static void mergeGenre( tvShow[] a, int low, int mid, int high )

{

tvShow[] temp = new tvShow[ high - low + 1 ];

int i = low, j = mid + 1, n = 0;

while ( i <= mid || j <= high )

{

if ( i > mid )

{

temp[ n ] = a[ j ];

j++;

}

else if ( j > high )

{

temp[ n ] = a[ i ];

i++;

}

else if ( a[ i ].getGenre().compareTo(a[ j ].getGenre()) < 0 )

{

temp[ n ] = a[ i ];

i++;

}

else

{

temp[ n ] = a[ j ];

j++;

}

n++;

}

for ( int k = low ; k <= high ; k++ )

a[ k ] = temp[ k - low ];

} // end of merge

public static void linearSearch(tvShow [] source, String target, int order)

{

boolean match = false;

if(order == 1) //finding shows that lasted 2 years

{

System.out.println("Show \t\t\t Year Premiered \t\t Seasons "

+ "\t\t Network \n");

for (tvShow source1 : source) {

if (target.equals(source1.getSeasons())) {

System.out.printf("%-20s%9s%30s%27s%n", source1.getName(),

source1.getYearPremier(), source1.getSeasons(),

source1.getNetwork());

match = true;

}

}

if(!match)

System.out.println("No shows that lasted 2 years found");

}

else if(order == 2) //finding shows that had more than 100 episodes

{

System.out.println("Show \t\t\t Seasons \t\t Episodes \n");

for (tvShow source1 : source) {

//converting strings to ints to compare

int x = Integer.valueOf(target);

int y = Integer.valueOf(source1.getEpisodes());

if (y >= x) {

System.out.printf("%-20s%9s%25s%n", source1.getName(),

source1.getSeasons(), source1.getEpisodes());

match = true;

}

}

if(!match)

System.out.println("No shows with more than 100 episodes found");

}

}

public static int binarySearch(tvShow [] source, String target, int order)

{

if(order == 1)

{

int low = 0;

int high = source.length - 1;

while (high >= low)

{

//int mid = (low + high) / 2;

int mid = low + high / 2 ;

//int count = 0;

if (target.equals(source[mid].getNetwork()))

{

return mid;

}

else if(target.compareTo(source[mid].getNetwork()) < 0)

{

high = mid -1;

}

else

{

low = mid + 1;

}

}

return -1 - low;

//return result;

}

else if(order == 2)

{

int low = 0;

int high = source.length - 1;

while (high >= low)

{

int mid = (low + high) / 2;

if (target.equals((source[mid].getGenre())))

{

return mid;

}

else if(target.compareTo((source[mid].getGenre())) < 0)

{

high = mid -1;

}

else

{

low = mid + 1;

}

}

return -1 - low;

}

return -1;

}

public static void matchAndPrint(tvShow[] source, int index, int order)

{

ArrayList<Integer> indexList = new ArrayList<>();

if(order == 1)

{

//checking if network name match with index returned and the index

//location to an arrayList

for(int i = 0; i < source.length; i++)

{

if(source[index].getNetwork().equals(source[i].getNetwork()))

{

indexList.add(i);

}

}

if(index == -1)

{

System.out.println("Not present");

}

else

{

System.out.println("Total shows counted: " + indexList.size());

//Using ArrayList to print network names and total number of shows

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

{

System.out.println("Show aired on CBS: " +

source[indexList.get(i)].getName());

}

}

}

else if(order == 2)

{

//checking if genre name match with index returned and the index

//location to an arrayList

for(int i = 0; i < source.length; i++)

{

if(source[index].getGenre().equals(source[i].getGenre()))

{

indexList.add(i);

}

}

if(index == -1)

{

System.out.println("Not present");

}

else

{

System.out.println("Total shows counted: " + indexList.size());

//Using ArrayList to print network names and total number of shows

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

{

System.out.println("Show on " + source[index].getGenre() + " Genre: " +

source[indexList.get(i)].getName());

}

}

}

}

}

/\*\*

\* Roberto Andino

\* Version 1

\* Program is done in Java programming language

\*

\* This is the Object class, it has a constructor that receives the data from

\* excel to create the tvShow objects.

\*

\*/

package tvshow;

public class tvShow {

private String name;

private String yearPremier;

private String seasons;

private String episodes;

private String network;

private String genre;

private String maleLead;

private String femaleLead;

//object constructor

tvShow(String name, String yearPremier, String seasons, String episodes, String network,

String genre, String maleLead, String femaleLead)

{

this.name = name;

this.yearPremier = yearPremier;

this.seasons = seasons;

this.episodes = episodes;

this.network = network;

this.genre = genre;

this.maleLead = maleLead;

this.femaleLead = femaleLead;

}

//getter methods

public String getName()

{

return name;

}

public String getYearPremier()

{

return yearPremier;

}

public String getSeasons()

{

return seasons;

}

public String getEpisodes()

{

return episodes;

}

public String getNetwork()

{

return network;

}

public String getGenre()

{

return genre;

}

public String getMaleLead()

{

return maleLead;

}

public String getFemaleLead()

{

return femaleLead;

}

}