Pauta Certamen 2, Programación II

Prof. Rodrigo Olivares

Mayo 26, 2016

Instrucciones:

- El puntaje máximo del certamen es 100 %, siendo el 60 % el mínimo requerido para aprobar.
- El certamen es **individual**. Cualquier intento de copia, será sancionado con nota **1,0**.

1. *30pts.* De las siguentes afirmaciones, encierre en un círculo la o las alternativas correctas.

i. **La clase `ArrayList`**:

(a) Es un `List`.
(b) Es un `Vector`.
(c) Es `synchronized`.
(d) Pertenece al `java.lang`.
(e) Es un nodo.

ii. **La instancia `this`**:

(a) Invoca al constructor de una clase padre.
(b) Invoca a la instancia de la clase hija.
(c) Referencia al construtor de la clase.
(d) Referencia a los atributos de la clase.
(e) Referencia a los método de la clase.

iii. **Una clase `abstract`**:

(a) Posee métodos abstractos.
(b) Puede contener métodos no abstractos.
(c) Instancia objetos abstractos.
(d) Posee un constructor abstracto.
(e) En Java, se define con la palabra reservada `abstract`.

iv. **Respecto a las interfaces**:

(a) Su constructor es creado en compilación.
(b) Sus métodos pueden ser `protected`.
(c) Sus métodos son abstract.
(d) Se implementa.
(e) Se extiende.

v. **Un TDA Bean**.

(a) Sólo tiene atributos.
(b) Sólo tiene métodos.
(c) No debe incluir lógica.
(d) El constructor debe ser siempre incluido.
(e) Es posible agregar métodos como `equals`() y `toString`().

vi. **Sobre la herencia**:

(a) En java se realiza con la palabra reservada `extends`.
(b) En java se realiza con la palabra reservada `impelements`.
(c) Todas las clases heredan de la clase `Object`.
(d) La clase padre hereda el comportamiento de la clase hija.
(e) La clase hija hereda el comportamiento de la clase padre.

vii. **Sobre las interfaces**:

(a) Proveen un medio de comunicación entre componentes.
(b) Sus métodos no están implementados.
(c) Instancia objetos sin comportamiento.
(d) Permite simular la herencia múltiple.
(e) En Java, se definen con la palabra reservada `interfaz`.

viii. **En relación a la manipulación de archivos**.

(a) Se lee un archivo con la instancia `FileWriter`.
(b) Se lee un archivo con la instancia `FileReader`.
(c) Se leen sólo archivos con delimitar y de largo fijo.
(d) `StringTokenizer` se usa para archivos con delimitador.
(e) Se requiere de la clase `Scanner` para la lectura.

ix. **En relación a la manipulación de archivos**.

(a) Se escribe un archivo con la instancia `FileWriter`.
(b) Se escribe un archivo con la instancia `FileReader`.
(c) No es factible agregar contenido a un archivo existente.
(d) Es necesario utilizar la clase `InputStreamReader`.
(e) `FileReader`("f.txt", `false`) agrega contenido al final.

x. **Excepción/es a considerar al manipular archivos**.

(a) `IOExceptionFile`
(b) `IOExceptionArchive`
(c) `IOException`
(d) `ExceptionIO`
(e) `ExceptionEx`

2. *70pts.* Utilice los dataset publicados en el aula virtual y luego responda las siguientes preguntas:

   a) Listar todas las películas de género Adventure.

   b) Listar todas las películas de género Thriller y Crime (al mismo tiempo).

   c) Listar todas las películas de un año específico, ingresado por el usuario (entrada estándar).

   d) Listar todas las películas de Rating superior o igual a un valor ingresado por el usuario.

   e) Listar todas las películas de Rating superior o igual a un valor ingresado por el usuario y su género sea sólo Comedy.

Importante:

- Cada listado debe ser desarrollado en métodos independientes.
- Cada resultado obtenido en los listados debe ser almacenado en un archivo de resultados. **NO SE DEBE SOBRE ESCRIBIR EL ARCHIVO**.

¿Cómo seré evaluado en la pregunta 3?			
Tópico	**Logrado**	**Medianamente logrado**	**No logrado**
Manipulación de archivo.	*20pts* Lee correctamente los archivos, los mapea a entidad y escribe en el archivo.	*10pts* Realiza dos de las tres acciones del punto anterior.	*0pts* No realiza la acciones del punto anterior.
TDA Lista.	*20pts* Crea la clase TDA Lista e implementa todos los métodos de manera independiente.	*10pts* Crea la clase TDA Lista e implementa algunos métodos.	*0pts* No crea la clase TDA Lista.
TDA Bean / Entidad	*10pts* Crea la clase entidad para Movie y Rating. Para la clase Movie, separa en una Lista de `String` el o los géneros de la película.	*5pts* Crea la clase entidad para Movie o la Rating (no ambas).	*0pts* No crea las clases entidad.
Clase principal y método main.	*5pts* Crea la clase principal en un archivo independiente con el método main.	*3pts* Crea el método main en la misma clase.	*0pts* No crea el método main.
Paradigma Orientación a Objetos	*15pts* Resuelve el problema utilizando el POO.	*7pts* Utiliza parte del POO para resolver el problema.	*0pts* No utiliza el POO para dar solución al problema.
Total máximo puntaje pregunta 2	*70pts*	*35pts*	*0pts*

```java
import java.util.List;

public class Movie {
    private String movieId;
    private String title;
    private List<String> genres;
    private List<Rating> rating;

    public Movie() {}

    public String getMovieId() {
        return movieId;
    }

    public void setMovieId(String movieId) {
        this.movieId = movieId.trim();
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title.trim();
    }

    public List<String> getGenres() {
        return genres;
    }

    public void setGenres(List<String> genres) {
        this.genres = genres;
    }

    public List<Rating> getRating() {
        return rating;
    }

    public void setRating(List<Rating> rating) {
        this.rating = rating;
    }

    public double getRantingPromedio() {
        double rat = 0;
        int cant = 0;
        for (Rating r : rating) {
            try {
                rat += Double.parseDouble(r.getRating());
                cant++;
            } catch(NumberFormatException e) {}
        }
        return rat/cant;
    }

    @Override
    public String toString() {
        String generos = "";
        for (String genre : genres) {
            generos += (genre + " ");
        }
        String ratings = "";
        for (Rating r : rating) {
            ratings += (r + " ");
        }
        return String.format("%s %s %s %s", movieId, title, generos, ratings);
    }
}
```

```java
public class Rating {

    private String userId;
    private String movieId;
    private String rating;

    public Rating() {}

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId.trim();
    }

    public String getMovieId() {
        return movieId;
    }

    public void setMovieId(String movieId) {
        this.movieId = movieId.trim();
    }

    public String getRating() {
        return rating;
    }

    public void setRating(String rating) {
        this.rating = rating.trim();
    }

    @Override
    public String toString() {
        return String.format("%s %s",userId, rating);
    }
}

import java.util.ArrayList;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Lista {

    private List<Movie> peliculas;
    private List<Rating> ratings;
    private List<String> lineasPelicula;
    private List<String> lineasRating;
    private List<String> busquedas;

    public Lista() {
        lineasPelicula = FuenteDatos.leerArchivo("/Users/rolivares/Desktop/movies.csv");
        lineasRating = FuenteDatos.leerArchivo("/Users/rolivares/Desktop/ratings.csv");
        busquedas = new ArrayList<>();
        cargarRating();
        cargarPeliculas();
    }

    private void cargarRating() {
        ratings = new ArrayList<>();
        Rating rating;
        StringTokenizer str;
        for (String linea : lineasRating) {
            str = new StringTokenizer(linea, ";");
            if (str.hasMoreTokens()) {
                rating = new Rating();
```

```
                rating.setUserId(str.nextToken());
                rating.setMovieId(str.nextToken());
                rating.setRating(str.nextToken());
                ratings.add(rating);
            }
        }
    }

    private void cargarPeliculas() {
        peliculas = new ArrayList<>();
        List<Rating> rts;
        List<String> generos;
        Movie pelicula;
        StringTokenizer stm, str;
        for (String linea : lineasPelicula) {
            stm = new StringTokenizer(linea, ";");
            if (stm.hasMoreTokens()) {
                try {
                    pelicula = new Movie();
                    pelicula.setMovieId(stm.nextToken());
                    pelicula.setTitle(stm.nextToken());
                    str = new StringTokenizer(stm.nextToken(), "|");
                    generos = new ArrayList<>();
                    while (str.hasMoreTokens()) {
                        generos.add(str.nextToken());
                    }
                    pelicula.setGenres(generos);
                    rts = new ArrayList<>();
                    for (Rating r : ratings) {
                        if (r.getMovieId().equals(pelicula.getMovieId())) {
                            rts.add(r);
                        }
                    }
                    pelicula.setRating(rts);
                    peliculas.add(pelicula);
                } catch (NoSuchElementException ex) {
                }
            }
        }
    }

    private void peliculasAdventure() {
        for (Movie pelicula : peliculas) {
            if (pelicula.getGenres().contains("Adventure")) {
                busquedas.add(pelicula.toString());
            }
        }
    }

    private void peliculasThrillerCrime() {
        for (Movie pelicula : peliculas) {
            if (pelicula.getGenres().contains("Thriller") &&
                pelicula.getGenres().contains("Crime") ) {
                busquedas.add(pelicula.toString());
            }
        }
    }

    private String leerDato(String texto) {
        Scanner sc;
        String dato = null;
        try {
            sc = new Scanner(System.in);
            System.out.print("Ingrese " + texto + ": ");
            dato = sc.nextLine();
        } catch (Exception e) {return "0";}
        return dato;
    }
```

```
    private void peliculaAnio() {
        String anio = leerDato("anio");
        for (Movie pelicula : peliculas) {
            if (pelicula.getTitle().contains(anio) ) {
                busquedas.add(pelicula.toString());
            }
        }
    }

    private void peliculaRatingPromedio() {
        double rp = Double.parseDouble(leerDato("rating pomedio"));
        for (Movie pelicula : peliculas) {
            if (pelicula.getRantingPromedio() >= rp) {
                busquedas.add(pelicula.toString());
            }
        }
    }

    private void peliculaRatingPromedioSoloComedy() {
        double rp = Double.parseDouble(leerDato("rating pomedio"));
        for (Movie pelicula : peliculas) {
            if (pelicula.getRantingPromedio() >= rp &&
                    pelicula.getGenres().size() == 1 &&
                    pelicula.getGenres().contains("Comedy")) {
                busquedas.add(pelicula.toString());
            }
        }
    }

    private void registrarBusqueda() {
        FuenteDatos.escribirArchivo("/Users/rolivares/Desktop/busqueda.txt", busquedas, true
    );
    }

    public void ejecutar() {
        peliculasAdventure();
        peliculasThrillerCrime();
        peliculaAnio();
        peliculaRatingPromedio();
        peliculaRatingPromedioSoloComedy();
        registrarBusqueda();
    }
}

public class Main {

    public static void main(String[] args) {
        Lista l = new Lista();
        l.ejecutar();
    }
}
```