

Programación Orientada a Objetos
Taller JavaFX
2020 2T

Objetivos:

- Crear una aplicación con interfaz gráfica que tenga contenido dinámico

Descripción:

Se le ha pedido a usted que ayude a desarrollar una aplicación para consulta de películas dado un género de película.

Usted cuenta con un archivo de llamado generos.txt con la información de los géneros de las películas en el paquete data que tiene el siguiente formato:

nombre,codigo

```
accion;0001  
comedia;0002  
ciencia ficcion;0003  
animacion;0004
```

Además, cuenta con un archivo de texto llamado películas.txt que tiene la información de las películas de la aplicación y que tiene el siguiente formato:

Nombre película; codigo categoria; ano; rating; nombre director; nombre de la imagen

```
Mad Max; 0001; 2015; 4; George Miller; mad_max_2015.jpeg  
Intensamente; 0002; 2015; 3; Pete Docker; intensamente_2015.jpeg  
Star War; 0003; 2015; 5; J J Abrams; starwars_2015.jpeg
```

A usted se le da un proyecto de Netbeans que tiene la implementación del modelo y la capa de datos y se le pide que implemente la capa de presentación de acuerdo a las siguientes indicaciones

INDICACIONES

1. Al inicio de la aplicación se muestra un ComboBox con los géneros de las películas, como en la Figura 1

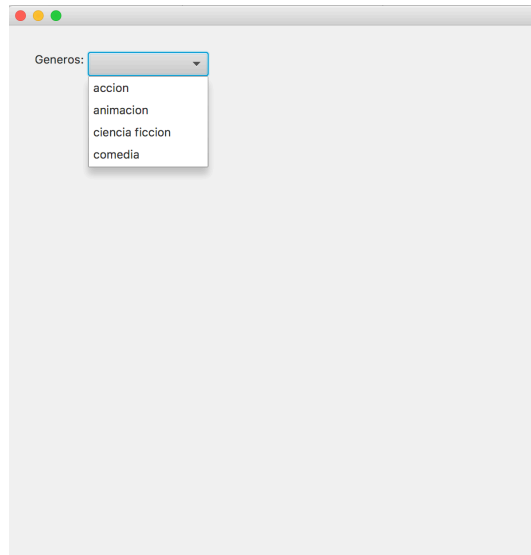


Figura 1

2. Cuando se selecciona un género aparecen las películas que pertenecen a dicho género. Por cada película se muestra su imagen, nombre y año de creación

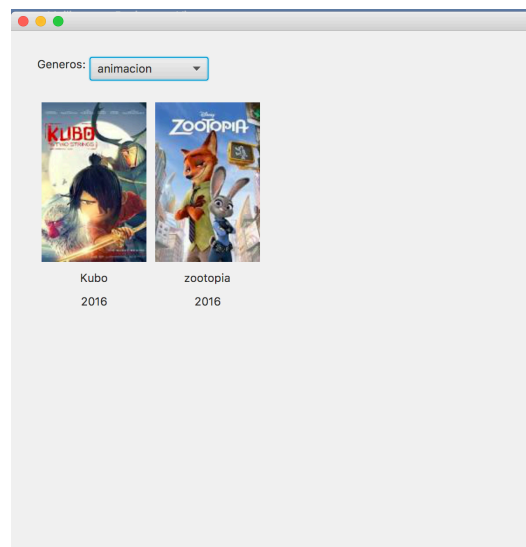


Figura 2.

3. Cuando se da click sobre el área de una película al lado derecho de la pantalla aparece la información detallada de la misma cómo se muestra en la Figura 3. El número de estrellas mostradas corresponde al rating de la película. El rating de va 1 a 5.

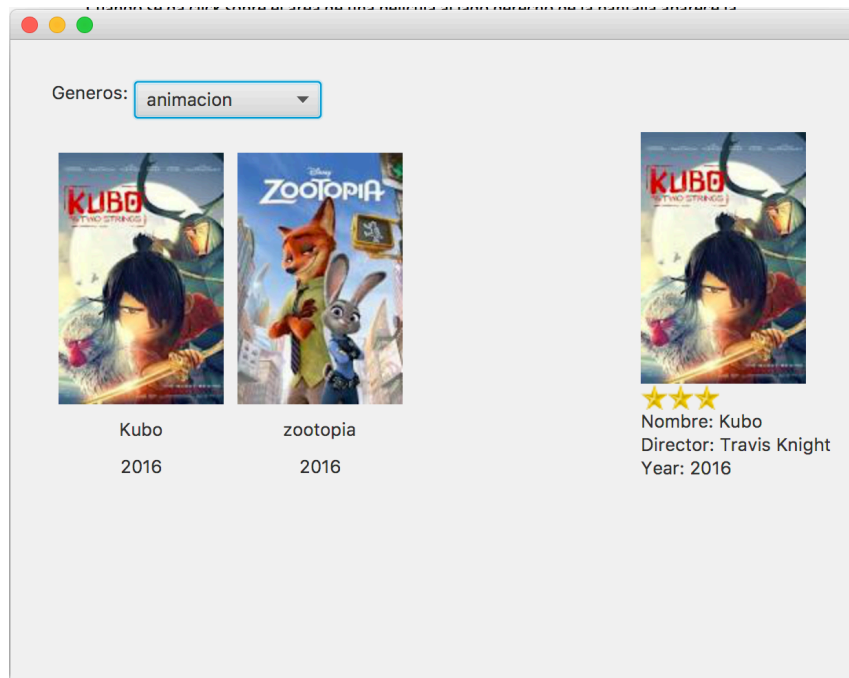


Figura 3.

PARTE 1 - Creación de la pantalla inicial

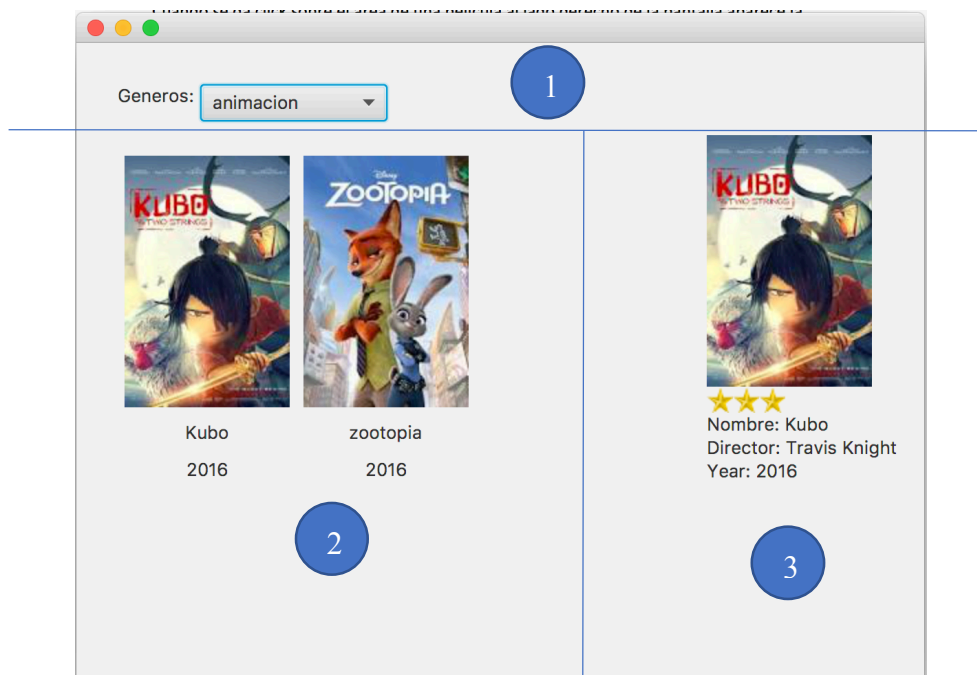
1. Selección del contenedor raíz de la aplicación

Como se puede apreciar en la Figura 4, la aplicación tiene tres secciones:

La sección donde se muestra el combo con la información de los géneros (1)

La sección donde se muestran las películas del género seleccionado (2)

La sección dónde se muestra información detallada de una película escogida (3)

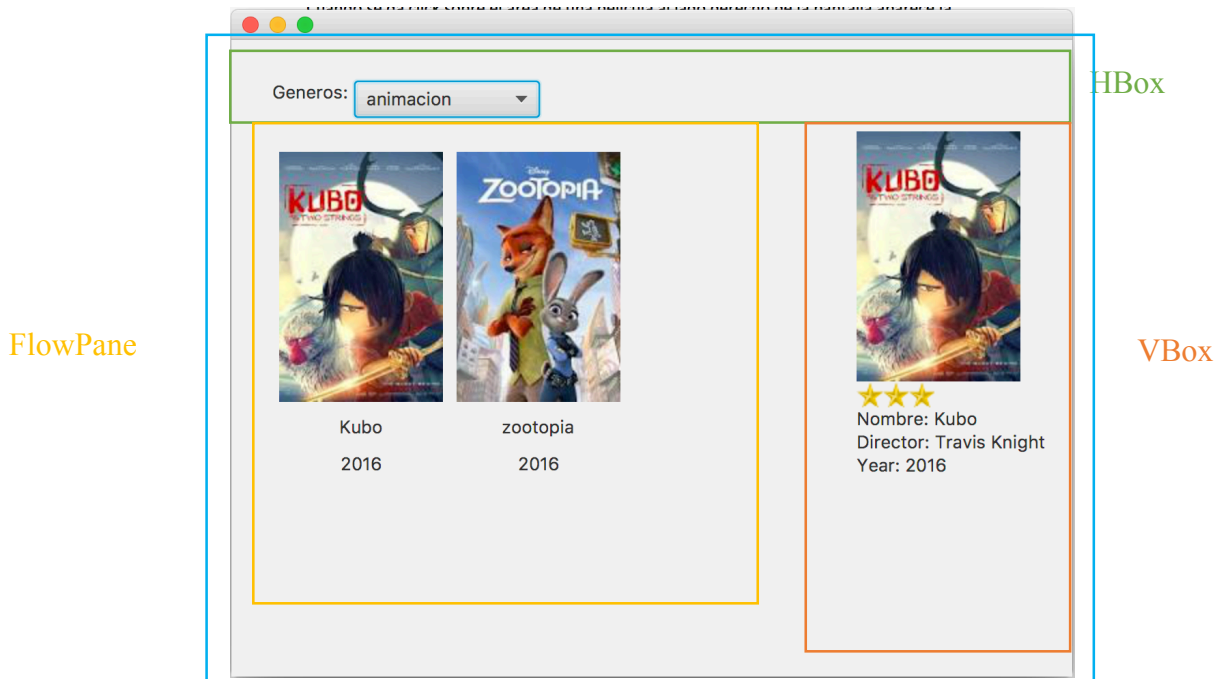


Para soportar esta distribución del contenido usaremos como contenedor raíz un `BorderPane`.

Distribución BorderLayout

- Top: HBox con un Label y ComboBox
- Center: FlowPane
- Right: VBox

BorderPane



2. La clase **VistaPelículas.java** en el paquete `com.mycompany.aplicacionpeliculas` creará los elementos gráficos de la vista. Actualmente la clase solo define el constructor raíz de la aplicación.

```
public class VistaPelículas{  
    //como root de nuestra aplicacion seleccionamos un BorderLayout  
    //porque en la aplicacion se visualiza claramente un Top donde  
    //se muestra el combo con los generos de peliculas  
    //un centro donde se muestran las peliculas que pertenecen  
    //al genero seleccionado y  
    //una seccion a la derecha donde se muestra informacion detallada  
    //de una pelicula seleccionada  
    public BorderLayout root;  
  
    public VistaPelículas(){  
        root = new BorderLayout();  
    }  
  
    public Pane getRoot(){  
        return root;  
    }  
}
```

- La clase **Principal.java** en el paquete `com.mycompany.aplicacionpeliculas` extiende de `Application` y será la clase principal de nuestra aplicación.

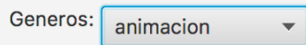
```

public class Principal extends Application{

    @Override
    public void start(Stage primaryStage) throws Exception {
        VistaPelículas pn = new VistaPelículas();
        Scene sc = new Scene(pn.getRoot(),600,600);
        primaryStage.setScene(sc);
        primaryStage.show();
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}

```

3. En la clase **VistaPelículas.java** implemente un método llamado crearTop que cree los elementos de la sección 1 de la figura 3.



Como contenedor de esta sección usaremos un HBox porque los elementos se posicionan uno a continuación del otro en sentido horizontal.

```

private void createTop() throws IOException{
    //HBox que tendrá el label y comboBox
    HBox paneTop = new HBox();
    paneTop.setPadding(new Insets(10,10,10,10));

    //creo un objeto de tipo ComboBox que tendrá objetos de tipo
    //genero
    ComboBox<Genero> cpelicula = new ComboBox<>();
    //el comboBox recibe un ObservableList con los elementos que
    //queremos mostrar.

    //el metodo setItems fija una colección dentro del comboBox
    cpelicula.setItems(
        FXCollections.observableArrayList(GeneroData.cargarGeneros()));

    paneTop.getChildren().addAll(new Label("Generos: "),
                                  cpelicula);

    //fijamos el HBox que tiene el label y el comboBox en el
    //top del BorderPane que es nuestro root.
    root.setTop(paneTop);
}

```

4. Llame al método creado en tres en el constructor de la clase.

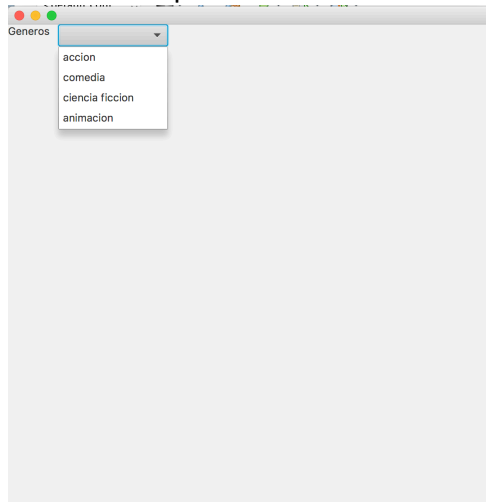
```

public VistaPeliculas(){
    try{
        root = new BorderPane();
        root.setPadding(new Insets(20,20,20,20));
        //llamamos al metodo que crea la seccion que contine el combo
        //y lo fija al top del root
        //este metodo puede lanzar una IOException
        createTop();
    }catch(IOException ex){
        //si encontramos una excepcion de tipo IOException
        //quitamos todos los elementos del contenedor raiz
        root.getChildren().clear();
        //fijamos en el centro un label con el mensaje de error
        String s = "Ha ocurrido un error\n";
        s += ex.getMessage();
        root.setCenter(new Label(s));
    }
}

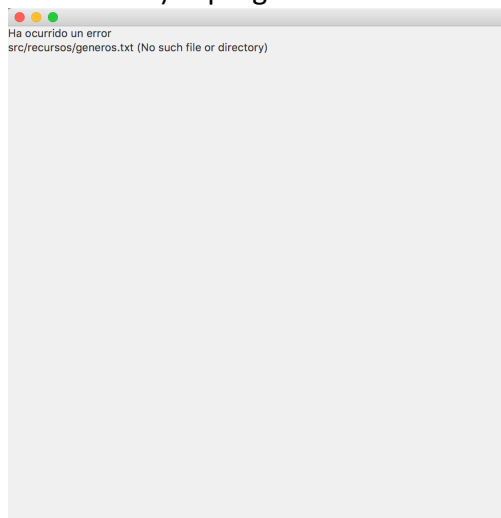
```

5. Pruebe lo realizado

Al correr la aplicación debe mostrarse la pantalla inicial con solo la sección 1.



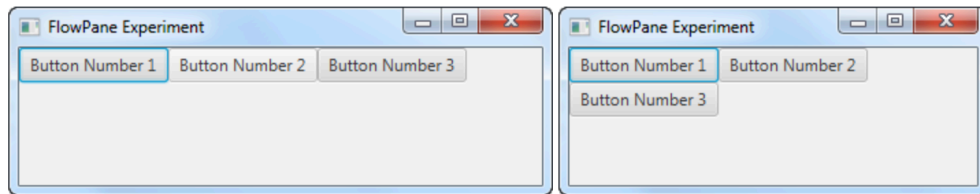
En caso de producirse algún error de IOException (por ejemplo, no exista alguno de los archivos) el programa no debe caerse y debe mostrar el mensaje de error.



6. Creación la sección del centro

Para la sección donde se muestran la información de las películas usaremos un **FlowPane**. Un **FlowPane** es un contenedor que coloca sus hijos uno a continuación del otro en sentido vertical u o horizontal (ver orientación), y automáticamente manda el elemento a la siguiente fila o columna (dependiendo de la orientación) si esta está llena

<http://tutorials.jenkov.com/javafx/flowpane.html>



7. Declaremos dos nuevas variables de instancias en la clase que tendrán referencia a al contenedor de las películas y al contenedor del detalle de una película al darle click. Y en el constructor de la clase inicializaremos estas variables de instancia.

```
public VistaPelículas(){
    try{
        root = new BorderPane();
        root.setPadding(new Insets(20,20,20,20));
        //llamamos al metodo que crea la seccion que contine el combo
        //y lo fija al top del root
        //este metodo puede lanzar una IOException
        createTop();

        paneListPelículas = new FlowPane();
        root.setCenter(paneListPelículas);

        detallePelícula = new VBox();
        root.setRight(detallePelícula);
    }catch(IOException ex){
        //si encontramos una excepcion de tipo IOException
        //quitamos todos los elementos del contenedor raiz
        root.getChildren().clear();
        //fijamos en el centro un label con el mensaje de error
        String s = "Ha ocurrido un error\n";
        s += ex.getMessage();
        root.setCenter(new Label(s));
    }
}
```

PARTE 2 – Fijar acción al ComboBox de Géneros y mostrar las películas del género seleccionado

Queremos que cuando seleccionemos una de las opciones del combobox con los géneros de las películas parezca la información de todas las películas de ese género. Para ello debemos fijar un manejador para el evento que se produce cuando se selecciona un elemento del combo.

Los manejadores son objetos de tipo `EventHandler` (es decir objetos de clases que implementan la interfaz `EventHandler`).

button.setOnAction((e) -> {codigo});

La interfaz EventHandler declara un único método llamado handle y es este método que debemos sobre-escribir y en dónde debemos colocar el código que queremos que se ejecute cuando ocurre el evento que estamos manejando

(Revisar diapositivas JavaFX – Eventos para más detalles)

8. **Modificamos el método crearTop** para agregar el código para manejar el evento – por el momento lo único que haremos cuando ocurra el evento es mostrar el valor seleccionado por el usuario en el contenedor que tendrá las películas.

```
//el metodo setOnAction fija el manejador para el evento que ocurre
//cuando se selecciona un nuevo elemento en el ComboBox
cpelicula.setOnAction(←
    (e)->{
        //el metodo getValue() retorna el objeto seleccionado en el Combo
        Genero ge = cpelicula.getValue();
        //por el momento lo que hacemos es agregar un Label
        //con el nombre de la categoria seleccionada en paneListPelículas
        Label l = new Label(ge.getNombre());
        paneListPelículas.getChildren().add(l);
    }
);
```

9. Pruebe lo realizado.

10. Para mostrar las películas debemos hacer lo siguiente

1. Debemos obtener las películas que pertenecen al género seleccionado
2. Por cada película que obtuvimos debemos:
 - a. Crear un objeto de tipo ImageView con la imagen de la película (ver apéndice)
 - b. Crear un objeto de tipo Label con el nombre de la película
 - c. Crear un objeto de tipo Label con el año de la película
 - d. Crear un VBox y agregar a él los elementos creados en a,b y c
 - e. Agregar el VBox al contenedor sección películas.
3. Cree un nuevo método llamado **mostrarPelículas**(Genero g) que encapsula el procedimiento anterior, y luego llame a ese método en el manejador que definimos anteriormente.

11. Pruebe lo realizado

PARTE 3 – Mostrar Detalle de cada película

12. Cada vez que se da click sobre alguna de las películas debe aparecer en el contenedor **secciondetalle** información detallada de la película en la que se dió click, como se ve en la Imagen3.

Fije un manejador para el evento de tipo **MouseClicked** sobre el contenedor que tiene la información de cada película y realice lo pedido.

APÉNDICE

Clase ImageView

- ImageView es un control que nos permite mostrar imágenes en la GUI.
- El constructor de este control recibe como parámetro un objeto de tipo Image.

```
ImageView(Image g);
```

Clase Image

- Permite cargar una imagen
- Tiene varios constructores:

```
Image(String pathimage)
```

```
Image(InputStream in)
```

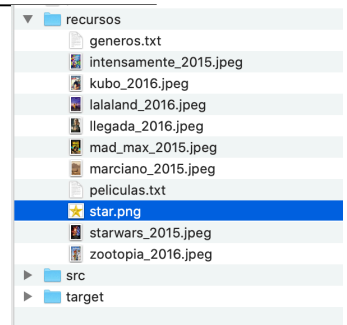
```
Image(String url, double width, double height, boolean perserveRatio,  
boolean smoth)
```

```
Image(InputStream, double width, double height, boolean perserveRatio,  
boolean smoth)
```

Ejemplo:

En el ejemplo se crea en la GUI con un imageView que muestre la imagen **start.png** que encuentra en la carpeta recursos. Este carpeta está localizada al mismo nivel que src.

```
public class EjemploImageView extends Application{  
    public void start(Stage primaryStage) {  
        primaryStage.setTitle("ImageView Experiment 1");  
  
        HBox hbox = new HBox();  
  
        FileInputStream input;  
        try {  
            input = new FileInputStream("recursos/star.png");  
            Image image = new Image(input);  
            ImageView imageView = new ImageView(image);  
            hbox.getChildren().add(imageView);  
        } catch (FileNotFoundException ex) {  
            hbox.getChildren().add(new Label("no se pudo cargar la  
imagen"));  
        }  
  
        Scene scene = new Scene(hbox, 600, 600);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
    public static void main(String[] args){  
        launch(args);  
    }  
}
```



Más detalles: https://www.tutorialspoint.com/javafx/javafx_images