Interfaces Persona Computador (IPC)

Examen de Prácticas – 10 de Junio de 2015

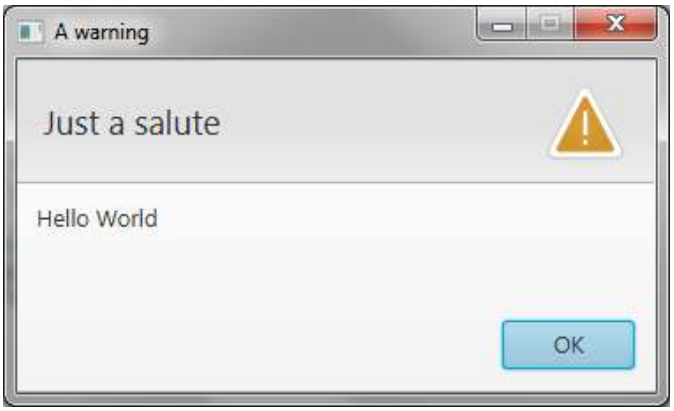
Alumno:	Grupo de prácticas:

*Este examen consta de **4 cuestiones**. Contestar a cada cuestión en los recuadros habilitados para ello. En la última hoja de este examen hay un anexo con información de la API útil para contestar las cuestiones. Duración del examen: Una hora y 15 minutos.*

1. Considérese una aplicación cuya ventana se muestra en la figura de la derecha. En su clase **`Main.java`** se ha escrito un método **`start`** donde se carga un fichero de vista, **`Sample.fxml`**, y se muestra la escena correspondiente. Y su clase controlador, **`SampleController.java`**, tiene el siguiente código:


```
public class SampleController implements Initializable {
   @FXML Button b;
   @Override
   public void initialize(URL location, ResourceBundle resources){
      b.setOnAction(new EventHandler<ActionEvent>() {
         public void handle(ActionEvent event) {
            Alert a = new Alert(AlertType.WARNING);
            a.setTitle("A warning");
            a.setHeaderText("Just a salute");
            a.setContentText( b.getText() );
            a.showAndWait();
      }});
   }
}
```

a) Dibujar la ventana que se mostrará en pantalla al pulsarse el botón. (***0.25 puntos***)



b) Indicar qué modificaciones tendrían que hacerse en la aplicación (en **`Main.java`**, **`SampleController.java`**, y **`Sample.fxml`**) para que, al ejecutarse, presentara la ventana mostrada a la derecha (que sería la nueva ventana principal), y tuviera la siguiente funcionalidad: si el usuario escribe un texto en el control tipo **`TextField`** y, a continuación, pulsa el botón, el texto en el **`TextField`** se mostrará en el área de texto del cuadro de mensaje.

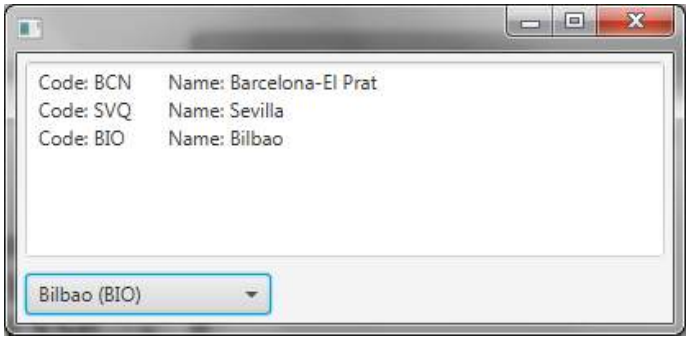
(***0.25 puntos***)

- En **Main.java** no sería necesario modificar nada.

- En **SampleController.java:**
  - se añadiría un atributo: `@FXML TextField tf;`
  - se sustituiría la línea: `a.setContentText( b.getText() );`
    por la línea: `a.setContentText( tf.getText() );`

- En **Sample.fxml** se añadiría un contenedor **VBox**, se colocarían en él un **Label**, un **TextField** y el **Button**, y se indicaría que el **fx:id** del **TextField** fuera **tf**.

2. Considérese una aplicación cuya ventana se muestra en la figura de la derecha.



Se supone que la implementación de la interfaz gráfica ya se ha hecho en el correspondiente fichero FXML, y que éste ya se ha conectado adecuadamente con su controlador, cuyo código (incompleto) es el siguiente:

```
   public class SampleController
implements Initializable {
      @FXML ComboBox<Airport> combo;
      @FXML TextArea ta;
      @Override
      public void initialize(URL location, ResourceBundle resources) {
         Data data = Data.getInstance();
         List<Airport> ports = data.getAirportList();
         ObservableList<Airport> la = FXCollections.observableArrayList(ports);
         combo.setItems(la);
         combo.getSelectionModel().selectFirst();
      }
   }
```

Se quiere implementar la siguiente funcionalidad: cada vez que se seleccione un aeropuerto en el control **ComboBox**, se añadirá una línea de texto en el control **TextArea** con la información del código y nombre del aeropuerto seleccionado. En la figura se muestra el contenido del área de texto después de seleccionar en el combo los aeropuertos de Barcelona, Sevilla y Bilbao.

Se pide añadir el código necesario en el controlador para dotar a la aplicación de la funcionalidad descrita.

(0.4 puntos)

Posibles soluciones:
- a) Añadir, en el método **initialize**, el siguiente código:

```
    combo.getSelectionModel().selectedItemProperty()
        .addListener(new ChangeListener<Airport>() {
            @Override
            public void changed(
                ObservableValue<? extends Airport> arg0,
                Airport arg1, Airport arg2) {
                    ta.appendText("Code: "+arg2.getCode()+
                    "\tName: "+arg2.getName()+"\n");
        }
    });
```

- b) Añadir, en el método **initialize**, el siguiente código:

```
    combo.getSelectionModel().selectedItemProperty()
        .addListener((observable, oldValue, newValue) -> {
            ta.appendText("Code: "+newValue.getCode()+
                        "\tName: "+newValue.getName()+"\n");
    });
```

- c) Asociar el siguiente método (a añadir en el controlador) con el control comboBox en SceneBuilder:

```
@FXML
public void comboAction(ActionEvent e) {
    Airport a = combo.getSelectionModel().getSelectedItem();
    ta.appendText("Code: "+a.getCode()+"\tName: "+a.getName()+"\n");
}
```

- d) Añadir, en el método **initialize**, el siguiente código:

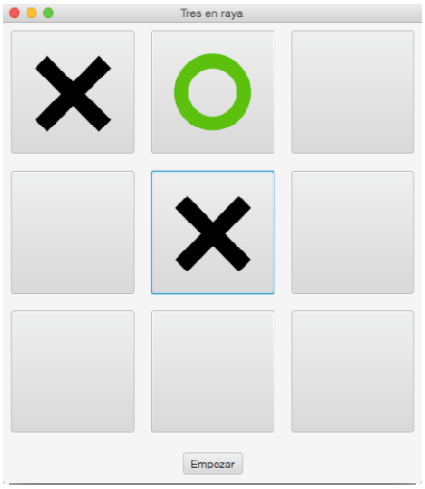
```
@FXML
combo.setOnAction(new EventHandler <ActionEvent>() {
    public void handle(ActionEvent e) {
    Airport a = combo.getSelectionModel().getSelectedItem();
    ta.appendText("Code: "+a.getCode()+"\tName: "+a.getName()+"\n");
}});
```

3. Implementa un método que reciba un array de **String** y un array de **int** (teniendo ambos arrays la misma longitud), y devuelva una lista observable con los datos a mostrar en un diagrama tipo **PieChart**.

(0.4 puntos)

```
ObservableList<PieChart.Data> setPieValues (String[] codes, int[] flights)
{
    ObservableList<PieChart.Data> pValues =
        FXCollections.observableArrayList();
    for (int i=0; i < flights.length; i++)
        pValues.add(new PieChart.Data(codes[i], flights[i]));
    return pValues;
}
```

4. Se desea implementar el juego del tres en raya, tal y como se muestra en la figura de la derecha.



La interfaz del juego está compuesta por 9 botones para las 9 casillas del juego y un botón adicional para reiniciar el juego. Se ha diseñado la pantalla con SceneBuilder y se ha implementado el siguiente controlador:

```
package view;
// Aquí van los imports correspondientes

public class Controller {
    @FXML private Button b21;
    @FXML private Button b10;
    @FXML private Button b20;
    @FXML private Button b01;
    @FXML private Button b12;
    @FXML private Button b00;
    @FXML private Button b11;
    @FXML private Button b22;
    @FXML private Button b02;
    private Image tic, tac;
    private boolean nextIsTic;

    @FXML
    void initialize() {
        tic = new Image(getClass().getResourceAsStream("tic.png"));
        tac = new Image(getClass().getResourceAsStream("tac.png"));
        nextIsTic = true;
    }
```

```
    @FXML
    void onClick(ActionEvent ae) {
       ImageView iv;
       if (nextIsTic) {
             iv = new ImageView(tic);
       } else {
             iv = new ImageView(tac);
       }
         iv.setFitWidth(100);
         iv.setPreserveRatio(true);
       ((Button)ae.getSource()).setGraphic(iv);
       nextIsTic = !nextIsTic;
    }
}
```

Contesta a las siguientes preguntas:

1. ¿Cómo harías para que al pulsar cualquier botón se ejecutara el método `onClick` del controlador? ***(0.1 puntos)***

Asignar a todos los botones dentro de SceneBuilder, en el apartado Código y opción OnAction, el método onClick.

2. Si tuvieras que usar un único contenedor para organizar los 9 botones de juego, ¿qué tipo de contenedor usarías? ***(0.1 puntos)***

Un GridPane de 3x3 elementos.

3. El código anterior tiene el problema de que cada vez que se pulsa un botón, se le asigna un nuevo icono, aunque ya tuviera uno asignado. ¿Cómo se podría solucionar este problema? ***(0.25 puntos)***

Se podría comprobar si el botón que se acaba de pulsar ya tiene un gráfico, y entonces no hacer nada, o deshabilitar el botón justo después de asignarle un gráfico.

4. Indica todos los pasos que deberías dar para implementar la acción de volver a empezar una partida (imagina que sólo se ha añadido el botón Empezar al FXML, debes conectarlo e implementar el código necesario para reiniciar la partida). ***(0.25 puntos)***

Añadir el siguiente método a la clase Controller:
 	
`@FXML`
`void onReset() {`
`b00.setGraphic(null);`
`b01.setGraphic(null);`
`b02.setGraphic(null);`
`b10.setGraphic(null);`
`b11.setGraphic(null);`
`b12.setGraphic(null);`
`b20.setGraphic(null);`
`b21.setGraphic(null);`
`b22.setGraphic(null);`
`nextIsTic = true;`
`}`
 	
y en el apartado Código\onAction del botón Empezar, seleccionar el método onReset.

ANEXO

```
class Airport
        public String getCode()
        public String getName()
interface ChangeListener<T>
        void changed(ObservableValue<? extends T> observable,
                        T oldValue, T newValue)
class ComboBox<T>
        public ComboBox(ObservableList<T> items)
        public final ObservableList<T> getItems()
        public final SingleSelectionModel<T> getSelectionModel()
        public final void setItems(ObservableList<T> value)
        public final ObjectProperty<ListCell<T>> buttonCellProperty()
        public final ObjectProperty<ObservableList<T>> itemsProperty()
class FXCollections
        public static <E> ObservableList<E> observableArrayList()
        public static <K,V> ObservableMap<K,V> observableMap(Map<K,V> map)
        public static ObservableFloatArray     observableFloatArray()
        public static ObservableIntegerArray observableIntegerArray()
        public static <E> ObservableSet<E>     observableSet(E... elements)
interface ObservableList<E>
        boolean add(E e)
        boolean addAll(Collection<? extends E> c)
        void addListener(ListChangeListener<? super E> listener)
        void remove(int from, int to)
        boolean removeAll(E... elements)
interface ObservableValue<T>
        void addListener(ChangeListener<? super T> listener)
        void removeListener(ChangeListener<? super T> listener)
        T getValue()
class PieChart
        public PieChart(ObservableList<PieChart.Data> data)
        public final ObservableList<PieChart.Data> getData()
        public final void setData(ObservableList<PieChart.Data> value)
        public final ObjectProperty<ObservableList<PieChart.Data>> dataProperty()
        public final void setLabelsVisible(boolean value)
        public final boolean getLabelsVisible()
class PieChart.Data
        public Data(String name, double value)
        public final String getName()
        public final double getPieValue()
        public final void setName(String value)
        public final void setPieValue(double value)
        public final DoubleProperty pieValueProperty()
        public final StringProperty nameProperty()
class TextArea
        public TextArea(String text)
        public void appendText(String text)
        public final void setText(String value)
        public final void setPrefColumnCount(int value)
        public final int getPrefColumnCount()
        public final int getPrefRowCount()
        public final void setPrefRowCount(int value)
        public final double getScrollTop()
        public final void setScrollTop(double value)
        public final double getScrollLeft()
        public final void setScrollLeft(double value)
        public final boolean isWrapText()
        public final void setWrapText(boolean value)
class TextField
        public TextField(String text)
        public final String getText()
        public final void setText(String value)
        public final int getPrefColumnCount()
        public final void setPrefColumnCount(int value)
        public final void setOnAction(EventHandler<ActionEvent> value)
        public final EventHandler<ActionEvent> getOnAction()
        public final void setAlignment(Pos value)
        public final Pos getAlignment()
class Button
        public Button(String text)
```

```
public Button(String text, Node graphic)
public final void setDefaultButton(boolean value)
public final void setCancelButton(boolean value)
public final String getText()
public final void setText(String value)
public final void setWrapText(boolean value)
public final void setFont(Font value)
public final void setOnAction(EventHandler<ActionEvent> value)
```