# REPORTE – PROYECTO SEGUNDO PARCIAL-PAR 4

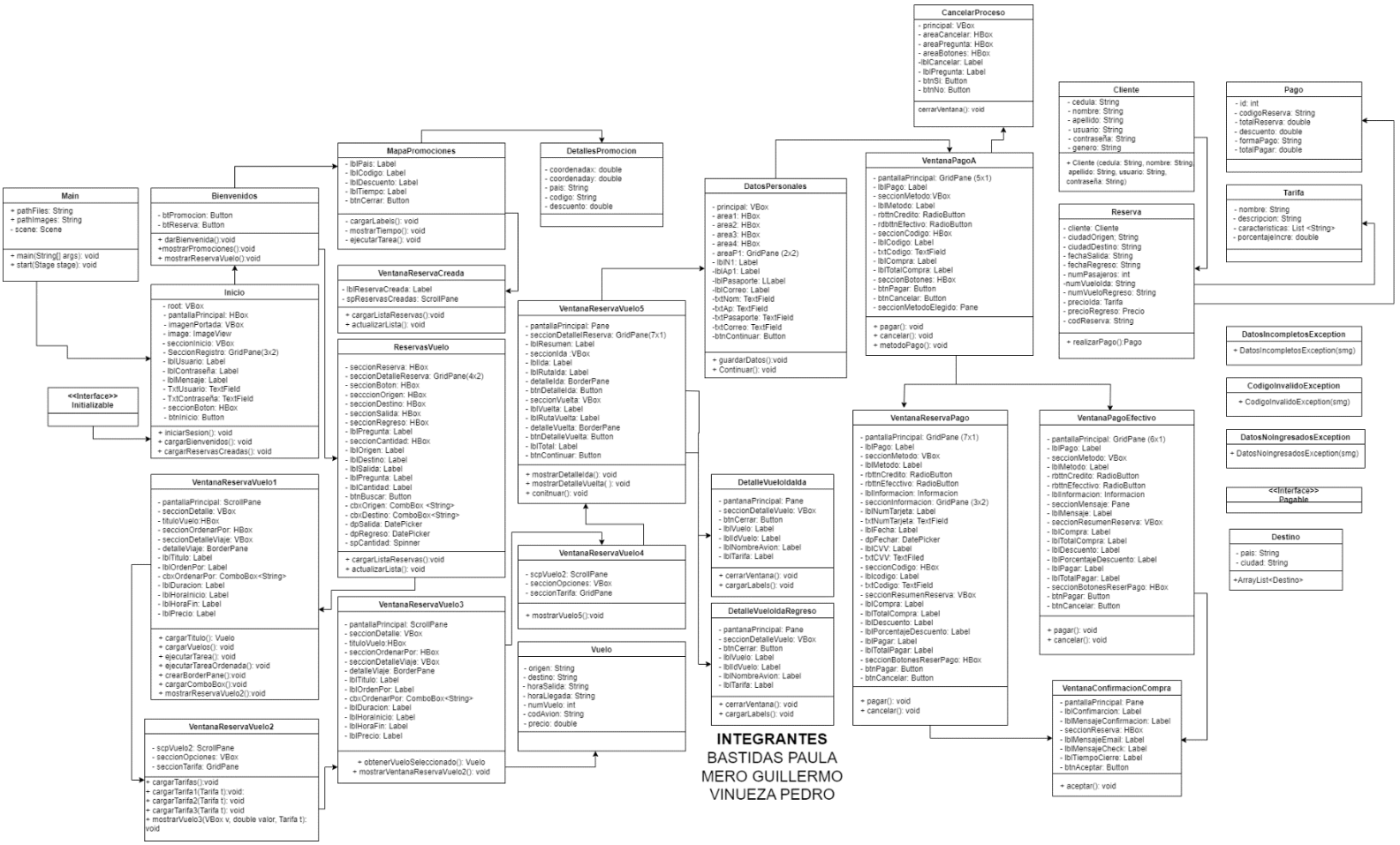
PROGRAMACIÓN ORIENTADA A OBJETOS
TÉRMINO II 2023-2024

Integrantes:
**Paula Bastidas**
**Guillermo Mero**
**Pedro Vinueza**

URL Repositorio:
https://github.com/GuillermoMero/POO5_2P_BASTIDAS_MERO_VINUEZA.git

Fecha:

# 1. Diagrama de clases




https://app.diagrams.net/#Wb!43UAEtuwd0udYG9kM2_2aecEBzt-hrlFgJknVulzyCkUY-YR7t-qRpOyRI9-wrSq%2F01GOM277JIREQQQA7W2BEJC74DZLX2XMWY

# 2. Tareas

En esta sección incluirán el detalle de las tareas que se asignó a cada estudiante.

### Estudiante (Paula Bastidas):

1. UML
2. Reporte y Java Doc. (documentación)
3. Diseño aplicación (Ventanas)
4. Clase Reserva Vuelo 5 (Manejo de excepciones)

### Estudiante (Guillermo Mero):

1. Desarrollar los métodos de clases (crear contenido de los métodos e interfaz)
2. Interfaces (Comparable y Pagable)
3. Creación de hilos en las clases
4. FXML

### Estudiante (Pedro Vinueza):

1. UML

# 3. Evidencias de Tareas

Estudiante (Paula Bastidas):

## Commit 1

**Avance en ReservaVuelo5**

main

**polipopi** committed 3 days ago

Showing **10 changed files** with **194 additions** and **40 deletions**.

10 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/ReservaVuelo4Controller.java

```
@@ -304,18 +304,18 @@ public void mostrarVuelo3(VBox v){
304 304             @Override
305 305             public void handle(MouseEvent m){
306 306                 Stage s = (Stage) v.getScene().getWindow();
307     -           FXMLLoader fl = new FXMLLoader(Main.class.getResource("ReservaVuelo3.fxml"));
    307 +           FXMLLoader fl = new FXMLLoader(Main.class.getResource("ReservaVuelo5.fxml"));
308 308               ReservaVuelo3Controller.origenSeleccionado = ReservaVuelo1Controller.destinoSeleccionado
309 309               ReservaVuelo3Controller.destinoSeleccionado = ReservaVuelo1Controller.origenSeleccionado
310     -           Parent rootVuelo3 = null;
    310 +           Parent rootVuelo5 = null;
311 311               try{
312     -               rootVuelo3 = fl.load();
    312 +               rootVuelo5 = fl.load();
313 313               }catch(IOException e){
314 314
315 315               }
316     -           Scene scene = new Scene(rootVuelo3);
```

## Commit 2

**Imagen de bienvenido**

main

**polipopi** committed 2 weeks ago

Showing **8 changed files** with **21 additions** and **7 deletions**.

BIN +217 KB Proyecto2P_Grupo2/imgBienvenido.png

12 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/BienvenidosController.java

```
@@ -4,12 +4,16 @@
4  4   */
5  5  package com.mycompany.proyecto2p_grupo2;
6  6
   7 + import java.io.FileInputStream;
   8 + import java.io.IOException;
7  9  import java.net.URL;
8  10 import java.util.ResourceBundle;
9  11 import javafx.fxml.FXML;
10 12 import javafx.fxml.Initializable;
11 13 import javafx.scene.control.Button;
12 14 import javafx.scene.control.Label;
   15 + import javafx.scene.image.Image;
   16 + import javafx.scene.image.ImageView;
```

## Commit 3

Commit

Manejo Excepciones

main

polipopi committed 2 weeks ago

Showing **4 changed files** with **7 additions** and **7 deletions**.

2 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/BienvenidosController.java

```
@@ -35,7 +35,7 @@ public class BienvenidosController implements Initializable {
35 35     @Override
36 36     public void initialize(URL url, ResourceBundle rb) {
37 37         try(FileInputStream input = new FileInputStream("src/main/resources/images/imgBienvenido.png")){
38    -             Image i = new Image(input,578,369,false,false);
   38 +             Image i = new Image(input,614,402,false,false);
39 39             imgBienvenido.setImage(i);
40 40         }catch(IOException e){
41 41             System.out.println("No se encuentra el archivo");
```

## Estudiante (Guillermo Mero):

## Commit 1

Creacion de la ventana Reservas creadas Y Mapa promociones

main

GuillermoMero committed 2 weeks ago

Showing **27 changed files** with **568 additions** and **72 deletions**.

42 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/BienvenidosController.java

```
@@ -8,13 +8,19 @@
8  8  import java.io.IOException;
9  9  import java.net.URL;
10 10 import java.util.ResourceBundle;
   11 + import javafx.event.ActionEvent;
11 12 import javafx.fxml.FXML;
   13 + import javafx.fxml.FXMLLoader;
12 14 import javafx.fxml.Initializable;
   15 + import javafx.scene.Parent;
   16 + import javafx.scene.Scene;
13 17 import javafx.scene.control.Button;
14 18 import javafx.scene.control.Label;
15 19 import javafx.scene.image.Image;
```

## Commit 2

Creacion de ventana Detalles Promocion

main

GuillermoMero committed last week

Showing **30 changed files** with **552 additions** and **44 deletions**.

Whitespace

5 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/BienvenidosController.java

```
@@ -83,4 +83,9 @@ void mostrarPromociones(ActionEvent e){
83 83         s.setScene(scene);
84 84         s.show();
85 85     }
   86 +
   87 +    @FXML
   88 +    void mostrarReservaVuelo(ActionEvent e){
   89 +
   90 +    }
```

## Commit 3

**Creacion de la clase Vuelo y ventana ReservaVuelo1**

main

**GuillermoMero** committed last week

Showing **30 changed files** with **724 additions** and **28 deletions**.

15 Proyecto2P_Grupo2/src/main/java/com/mycompany/proyecto2p_grupo2/BienvenidosController.java

```
@@ -70,7 +70,7 @@ public void darBienvenida(String genero, String nombre){
70                                                          70
71          @FXML                                           71          @FXML
72          void mostrarPromociones(ActionEvent e){         72          void mostrarPromociones(Ac
73  -           Stage s = (Stage)btnPromo.getScene().getWindow();   73  +           Stage s = new Stage();
74              FXMLLoader fl = new                         74              FXMLLoader fl = new
    FXMLLoader(Main.class.getResource("MapaPromociones.fxml"));     FXMLLoader(Main.class.getResou
75              Parent rootPromo = null;                    75              Parent rootPromo = nul
76              try{                                        76              try{
@@ -86,6 +86,17 @@ void mostrarPromociones(ActionEvent e){
86                                                          86
```

## Estudiante (Pedro Vinueza):

(screenshots)

# 4. Identificación de teoría aplicada en programación orientada a objetos.

## Manipulación de objetos

1)

```
*/
@Override
public void initialize(URL url, ResourceBundle rb) {
    try(FileInputStream input = new FileInputStream("src/main/resources/images/mapa.png")){
        Image i = new Image(input);
        imgMapa.setImage(i);
    }catch(IOException e){
        System.out.println("No se encuentra el archivo");
    }
    mostrarPuntos();
```

**Explicación**

Se manipulan objetos de la interfaz gráfica mediante el uso de un ImageView (imgMapa), un Pane (seccionMapa), y un AnchorPane (root). Además, se crean objetos ImageView dinámicamente en el método cargarImagen y se utilizan para mostrar puntos en el mapa.

2)

```
private Button btnCerrar;

public void cargarLabels(String pais, String codigo, String descuento){
    lblPais.setText(pais);
    lblCodigo.setText(codigo);
    lblDescuento.setText(descuento);
}
```

**Explicación**

Se manipulan objetos de la interfaz gráfica mediante el uso de etiquetas (Label), botones (Button), y ventanas (Stage). Por ejemplo, lblPais, lblCodigo, lblDescuento, lblTiempo, btnCerrar, y Stage s.

3)

```
public void actualizarLista(){
    ArrayList<String> b = new ArrayList<>();
    b.add("Cambio");
    b.add("cambio 2");
    listReservas.getItems().clear();
    listReservas.getItems().addAll(b);
}
```

**Explicación**

Se manipulan objetos de la interfaz gráfica mediante el uso de ImageView (imgAvion) y ListView<String> (listReservas). Se utiliza la clase Image para cargar una imagen en el ImageView.

# Threads

1)

```
public void mostrarTiempo(){
    Thread t = new Thread(new Runnable(){
        @Override
        public void run() {
            System.out.println("Empezo hilo "+Thread.currentThread());
            ejecutarTarea();
            Stage s = (Stage)lblTiempo.getScene().getWindow();
            cerrarVentana(s);
            System.out.println("Termina hilo "+Thread.currentThread());
        }
    });
    t.setName("Tiempo de cierre");
    t.setDaemon(true);
    t.start();

}
```

## Explicación

Se utiliza un hilo (Thread) en el método mostrarTiempo() para realizar una tarea en segundo plano (actualización del tiempo de cierre). También se utiliza Platform.runLater para actualizar componentes de la interfaz gráfica desde el hilo de JavaFX.

2)

```
private AnchorPane root;

public void mostrarPuntos(){
    ArrayList<Promocion> promos = Promocion.leerPromociones();
    Random rd = new Random();
    int numAleatorio = rd.nextInt(10) + 1;
    Thread t = new Thread(new Runnable(){
        @Override
        public void run(){
            System.out.println("Empieza hilo "+Thread.currentThread());
            for(Promocion p: promos){
                ImageView imgUbi = new ImageView();
                System.out.println("Mostrando promoción en "+p.getPais());
                cargarImagen(imgUbi,p);
                try{
                    Thread.sleep(numAleatorio*1000);
                }catch(InterruptedException e){
```

## Explicación

Se utiliza un hilo (Thread) en el método mostrarPuntos() para realizar una tarea en segundo plano (mostrar promociones en el mapa).

3)

```
public void ejecutarTarea(){
    Thread t = new Thread(new Runnable(){
        @Override
        public void run(){
            System.out.println("Empieza hilo "+Thread.currentThread());
            ArrayList<Vuelo> vuelos = Vuelo.leerVuelos();
            for(int i=0; i<vuelos.size(); i++){
                Vuelo v = vuelos.get(i);
                if(origenSeleccionado.equals(v.getOrigen()) && destinoSeleccionado.equals(v.getDestino())){
                    try{
                        Thread.sleep(1000);
                    }catch(InterruptedException e){
```

## Explicación

Se utilizan hilos (Thread) en los métodos ejecutarTarea y ejecutarTareaOrdenada para realizar tareas en segundo plano (mostrar vuelos de manera asíncrona).

## Manejo de Excepciones

1)

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    try(FileInputStream input = new FileInputStream("src/main/resources/images/imc
        Image i = new Image(input,614,402,false,false);
        imgBienvenido.setImage(i);
    }catch(IOException e){
        System.out.println("No se encuentra el archivo");
    }
```

### Explicación

El código maneja excepciones, por ejemplo, en la lectura de un archivo de imagen (FileInputStream input = new FileInputStream). Se utiliza el bloque catch (IOException e) para manejar posibles problemas durante la lectura.

2)

```
public void cargarBienvenidos(Cliente c, Stage s){
    FXMLLoader f = new FXMLLoader(Main.class.getResource("Bienvenidos.fxml"));
    Parent root = null;
    try{
        root = f.load();
    }catch(IOException e2){

    }
    BienvenidosController bc = f.getController();
    bc.darBienvenida(c.getGenero(),c.getNombre());
```

### Explicación

Se manejan excepciones en el bloque try-catch al intentar cargar una imagen desde un archivo (FileInputStream input = new FileInputStream). También hay un bloque try-catch al intentar cargar FXMLLoader en los métodos cargarBienvenidos y cargarReservasCreadas.

3)

```
for(Promocion p: promos){
    ImageView imgUbi = new ImageView();
    System.out.println("Mostrando promoción en "+p.getPais());
    cargarImagen(imgUbi,p);
    try{
        Thread.sleep(numAleatorio*1000);
    }catch(InterruptedException e){

    }
}
System.out.println("Termina hilo "+Thread.currentThread());
```

### Explicación

Se manejan excepciones en el bloque try-catch al intentar cargar una imagen desde un archivo (FileInputStream input = new FileInputStream). También se manejan excepciones al cargar la imagen del detalle de la promoción (FileInputStream in = new FileInputStream (Main.pathImages+"ubicacion.png")).

## Manejo de Excepciones propias

1)

```
 *
 * @author LENOVO
 */
public class DatosNoIngresadosException extends RuntimeException{
    public DatosNoIngresadosException(String msg){
        super(msg);
    }
}
```

**Explicación**

Al crear una clase solo para una excepción, se considera una excepción propia.

2)

```
public void mostrarDetallesVuelo(Vuelo v, Button btn){
    btn.setOnAction(new EventHandler<ActionEvent>(){
        @Override
        public void handle(ActionEvent e){
            Stage s = new Stage();
            FXMLLoader fl = new FXMLLoader(Main.class.getResource("DetallesPromocion.fxml"));
            Parent rootDetallesVuelo = null;
            try{
                rootDetallesVuelo = fl.load();
            }catch(IOException i){
                i.printStackTrace();
            }
```

**Explicación**

Se manejan excepciones en los bloques try-catch exactamente al intentar pausar el hilo mediante Thread.sleep en el método cargarResumen.

3)

```
public void cargarResumen(){
    Thread t = new Thread(new Runnable(){
        @Override
        public void run(){
            System.out.println("Empezando hilo "+Thread.currentThread());
            try{
                Thread.sleep(1000);
            }catch(InterruptedException e){

            }
            cargarIda();
            try{
                Thread.sleep(1000);
            }catch(InterruptedException e){

            }
```

**Explicación**

Se manejan excepciones en los bloques try-catch al intentar pausar el hilo mediante Thread.sleep en el método cargarResumen.

## Lectura de Archivos

1)

```
public static ArrayList<Cliente> leerClientes(){
    File f = null;
    FileReader fr = null;
    BufferedReader br = null;
    ArrayList<Cliente> clientes = new ArrayList<>();
    Cliente c;
    try{
        f = new File(Main.pathFiles+"clientes.txt");
        fr = new FileReader(f,StandardCharsets.UTF_8);
        br = new BufferedReader(fr);
        br.readLine();
        String linea;
        while((linea=br.readLine())!=null){
```

**Explicación**

Se realiza una lectura de archivo para separar los datos de “clientes.txt”

2)

```
public void cargarDestino() throws IOException{
    ArrayList<Destino> destinos = Destino.leerDestinos();
    cbDestinos.getItems().setAll(destinos);
}
```

**Explicación**

Se lee un archivo de destinos en el método cargarDestino a través de la llamada al método estático leerDestinos de la clase Destino. El código asociado a esta lectura no está proporcionado.

3)

```
public static ArrayList<Tarifa> leerTarifas(){
    ArrayList<Tarifa> tarifas = new ArrayList<>();
    File archivo = null;
    FileReader fr = null;
    BufferedReader br = null;
    try{
        archivo = new File(Main.pathFiles+"tarifas.txt");
        fr = new FileReader(archivo,StandardCharsets.UTF_8);
        br = new BufferedReader(fr);
        br.readLine();
        String linea;
        while((linea=br.readLine())!=null){
```

**Explicación**

Se realiza lectura de archivos para cargar las tarifas mediante el método leerTarifas de la clase Tarifa.

## Escritura de Archivos

1)

```
private void escribirReserva(String nombreArchivo, Reserva reserva) {
    try (FileWriter fw = new FileWriter(nombreArchivo, true);
         BufferedWriter bw = new BufferedWriter(fw)) {

        String linea = construirLineaReserva(reserva);
        bw.write(linea);
        bw.newLine();

        System.out.println("Reserva guardada en el archivo.");

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

**Explicación**

Se está realizando una escritura de archivo para reservar los clientes que van agendando los viajes.

## Serialización

1)

```
  private void serializarReserva(Reserva reserva) {
    try (ObjectOutputStream oos = new ObjectOutputStream(
          new FileOutputStream(reserva.getCodigo() + ".bin"))) {

        oos.writeObject(reserva);
        System.out.println("Reserva serializada correctamente.");

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

**Explicación**

La serialización en este caso se utiliza para guardar el estado de la reserva en un formato binario que se puede recuperar posteriormente

## Interfaz Comparable

1)

```
public class Vuelo implements Comparable<Vuelo>{
    private String numVuelo;
    private String origen;
    private String destino;
    private int duracion;
    private String horaSalida;
```

**Explicación**

En esta clase se implementó una interfaz comparable ya que, se realiza para ordenar los datos por tipos.

## Interfaz Pagable

```
 */
interface Pagable {
    Pago generarTransaccion();
}
```

**Explicación**

La clase Reserva implementa la interfaz Pagable. Esto significa que la clase Reserva debe proporcionar una implementación para el método generarTransaccion() definido en la interfaz.

## Controladores de eventos

1)

```
public void cargarListaReservas(){

    a.add("Vacio");
    a.add("Vacio 2");
    listReservas.getItems().setAll(a);
    Timeline timeline;
    timeline = new Timeline(new KeyFrame(
            Duration.seconds(5),
            event -> actualizarLista()));
    timeline.setCycleCount(Timeline.INDEFINITE);
    timeline.play();
}
```

**Explicación**

Se utiliza un Timeline y un KeyFrame como controlador de eventos para actualizar la lista de reservas periódicamente.

2)

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    try{
        cargarOrigenes();
        cargarDestino();
        cargarSpinner();
        cambiarIdioma();
    }catch(IOException | RuntimeException e){
```

**Explicación**

Se utiliza el método buscar como controlador de eventos para el botón btnBuscar. También se utiliza el método initialize para configurar la interfaz inicial.

## Programación dinámica de GUI

### 1)

```
public void cargarOrigenes(){
    ArrayList<String> origenes = new ArrayList<>();
    origenes.add("Guayaquil");
    origenes.add("Quito");
    origenes.add("Cuenca");
    cbOrigenes.getItems().setAll(origenes);
}
```

**Explicación**

Hay programación dinámica de la GUI al configurar elementos como ComboBox, DatePicker, y Spinner en los métodos cargarOrigenes, cargarDestino, y cargarSpinner.

### 2)

```
@Override
public void run(){
    System.out.println("Empieza hilo "+Thread.currentThread());
    for(Promocion p: promos){
        ImageView imgUbi = new ImageView();
        System.out.println("Mostrando promoción en "+p.getPais());
        cargarImagen(imgUbi,p);
        try{
            Thread.sleep(numAleatorio*1000);
        }catch(InterruptedException e){
```

**Explicación**

Hay programación dinámica de la GUI al crear y mostrar dinámicamente objetos ImageView en el mapa, y al abrir una nueva ventana con detalles de promoción.

## Posicionamiento absoluto

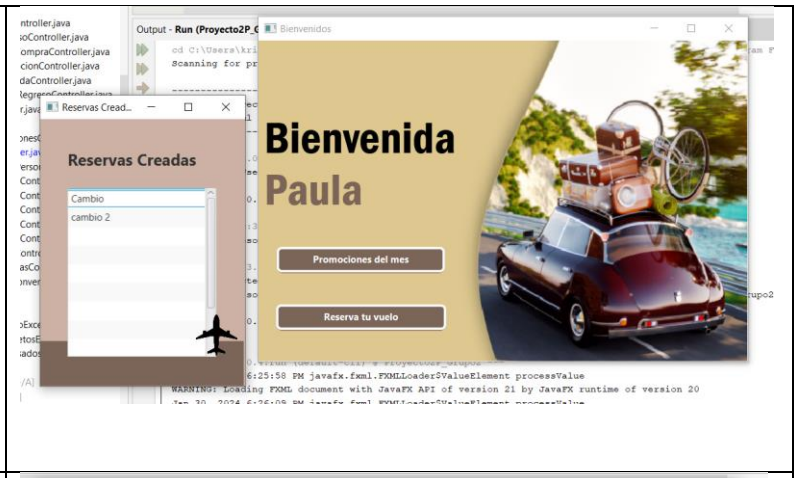
### 1)

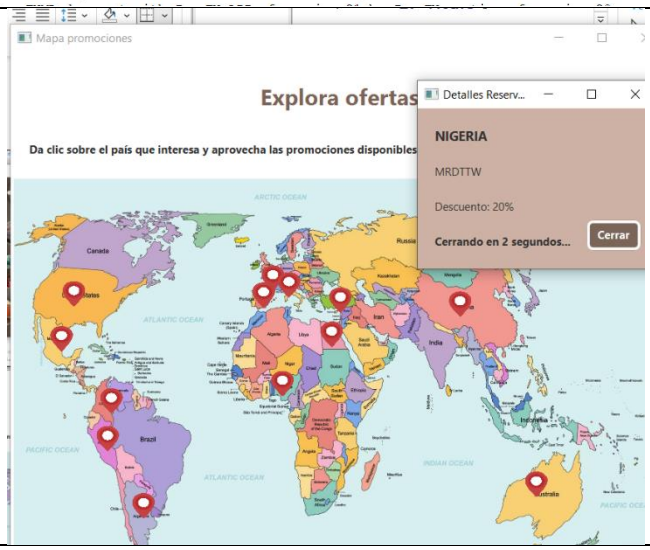
```
img.setLayoutX(p.getX());
img.setLayoutY(p.getY());
```

**Explicación**

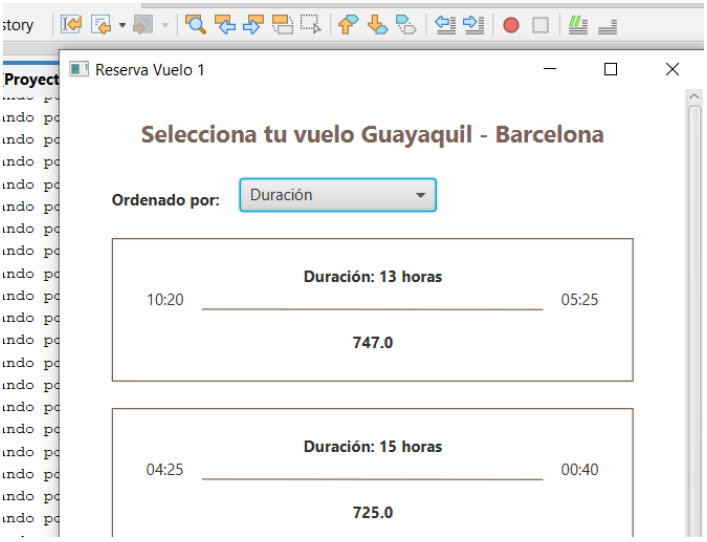
Se identifico un posicionamiento absoluto ya que se utilizó para las posiciones de las ubicaciones en el mapa de promociones.

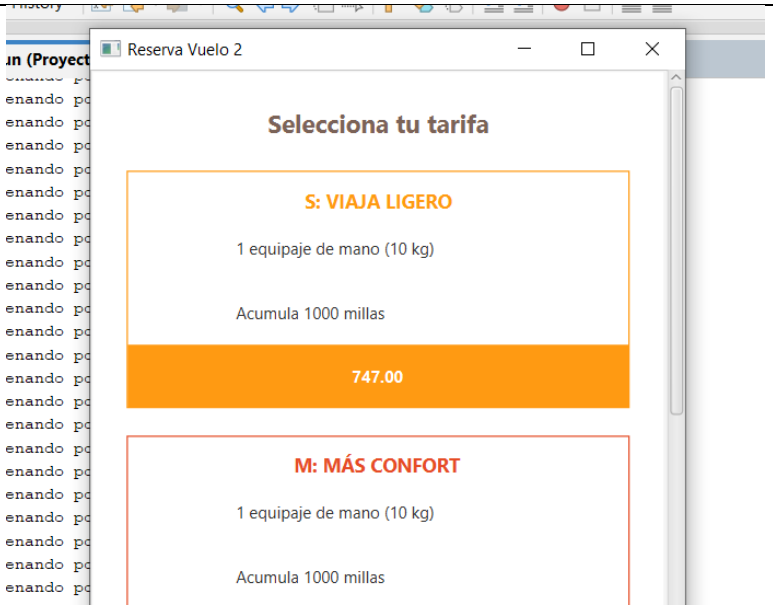
# 5. Programa en ejecución

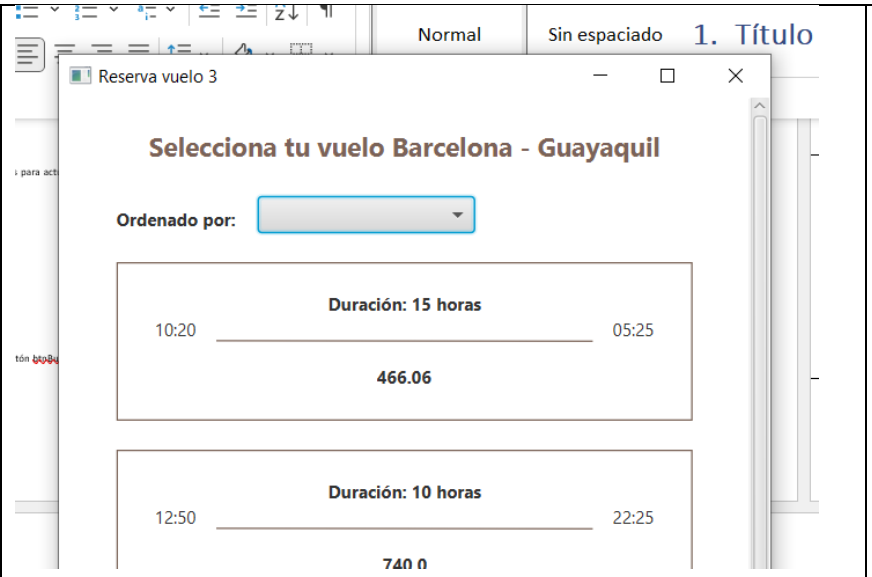



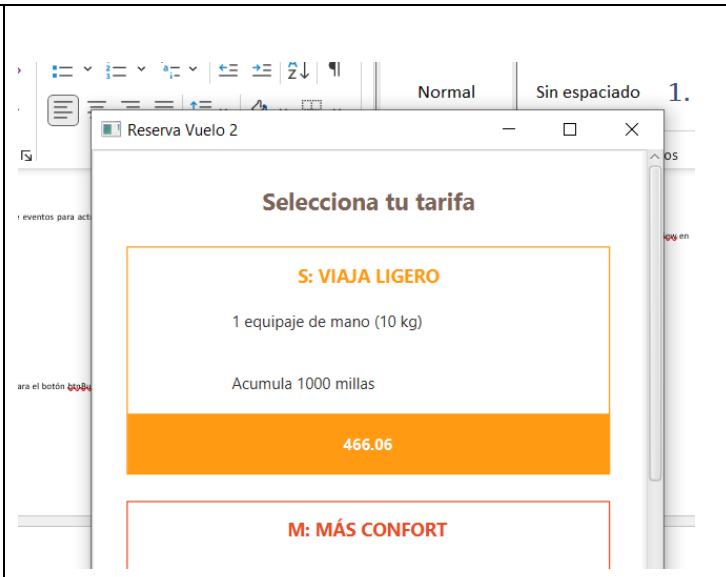



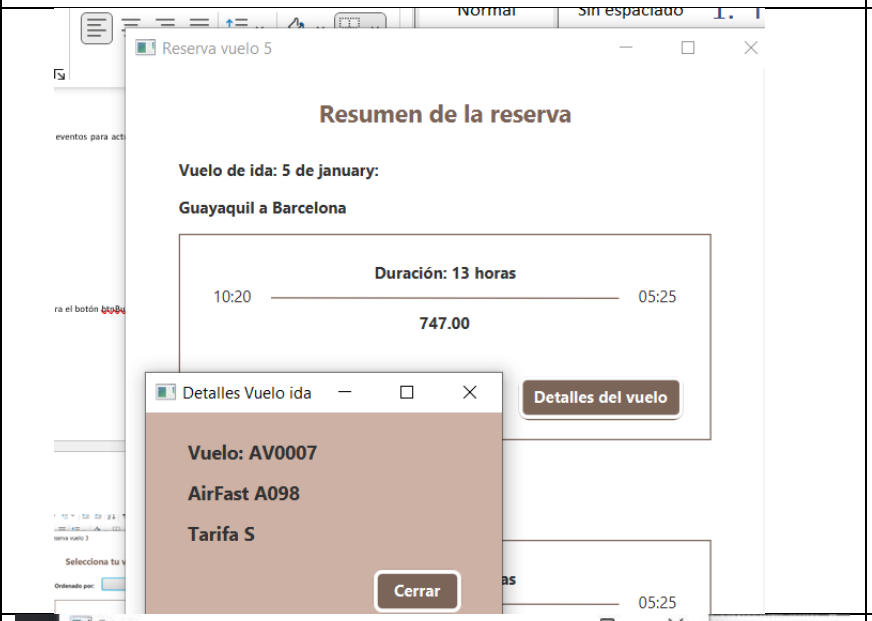


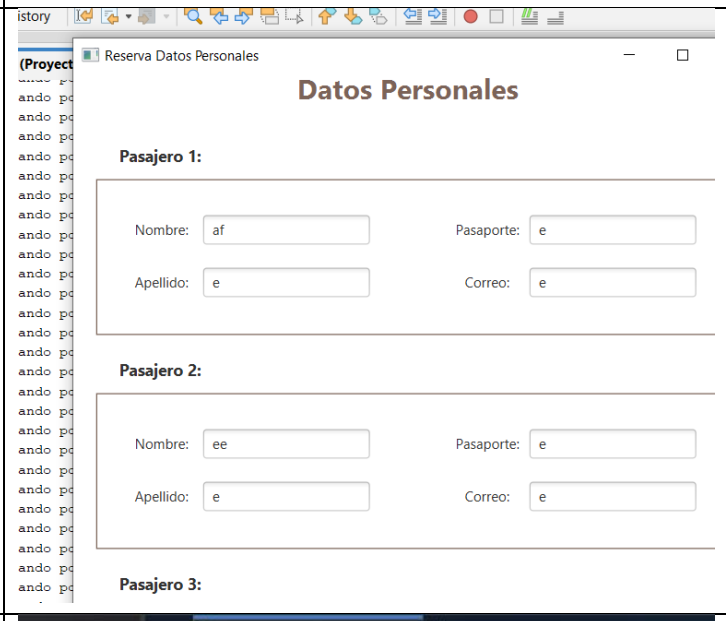


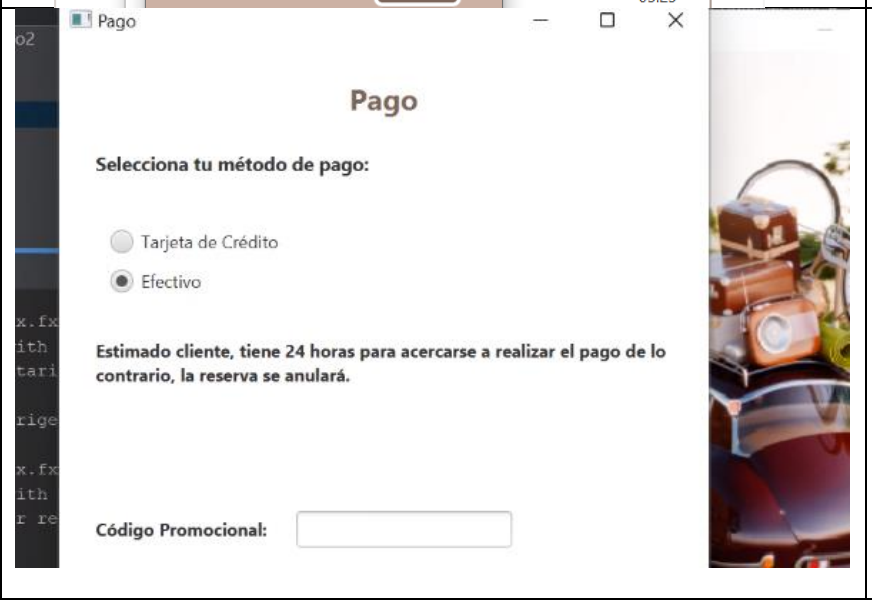



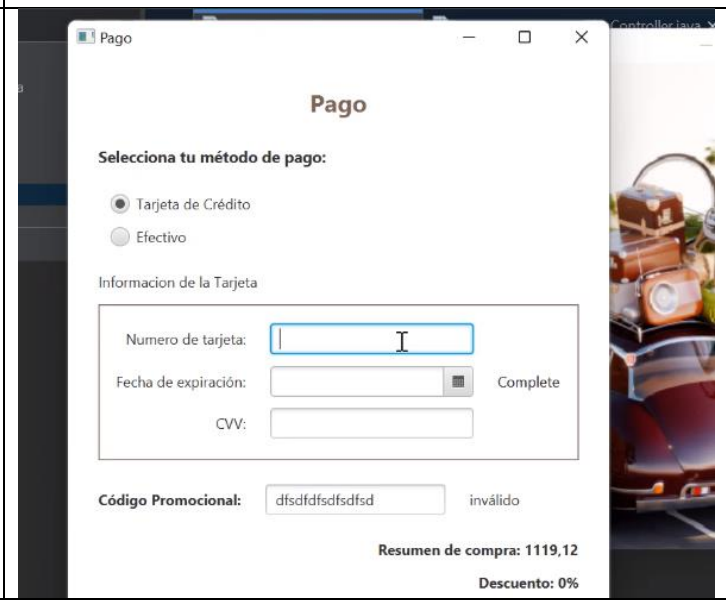

Reserva vuelo 3
Selecciona tu vuelo Barcelona - Guayaquil
Ordenado por:
Duración: 15 horas
10:20
05:25
466.06
Duración: 10 horas
12:50
22:25
740.0

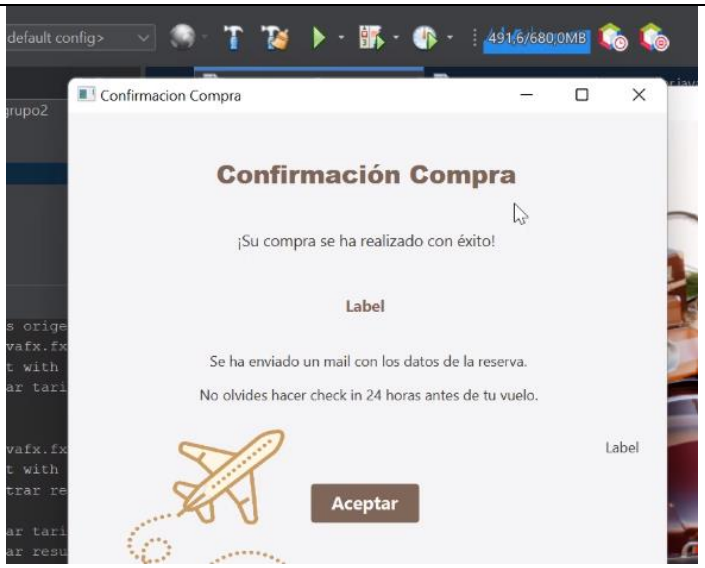

Reserva Vuelo 2
Selecciona tu tarifa
S: VIAJA LIGERO
1 equipaje de mano (10 kg)
Acumula 1000 millas
466.06
M: MÁS CONFORT


Reserva vuelo 5
Resumen de la reserva
Vuelo de ida: 5 de january:
Guayaquil a Barcelona
Duración: 13 horas
10:20
05:25
747.00
Detalles del vuelo
Detalles Vuelo ida
Vuelo: AV0007
AirFast A098
Tarifa S
Cerrar


Reserva Datos Personales
Datos Personales
Pasajero 1:
Nombre: af
Pasaporte: e
Apellido: e
Correo: e
Pasajero 2:
Nombre: ee
Pasaporte: e
Apellido: e
Correo: e
Pasajero 3:


Pago
Pago
Selecciona tu método de pago:
Tarjeta de Crédito
Efectivo
Estimado cliente, tiene 24 horas para acercarse a realizar el pago de lo contrario, la reserva se anulará.
Código Promocional:


Pago
Pago
Selecciona tu método de pago:
Tarjeta de Crédito
Efectivo
Informacion de la Tarjeta
Numero de tarjeta:
Fecha de expiración:
Complete
CVV:
Código Promocional:
dfsdfdfsdfsdfsd
inválido
Resumen de compra: 1119,12
Descuento: 0%


## 6. JAVADOC

**Agregar la documentación JAVADOC como una carpeta adicional a este reporte**. Los métodos deben estar siempre comentados con el formato explicado en clase.