

Introducción:

El objetivo de este informe es mostrar a detalle el desarrollo y funcionamiento de un programa de aerolínea basado en otro ya creado y posteriormente en este ejercicio modificado según nuevos requerimientos funcionales:

Enunciado (programa original):

Se quiere crear un programa que tenga como objetivo el manejo de reservas de un avión. Este avión cuenta con un número fijo de 50 sillars. De ellas, 8 son de clase ejecutiva, mientras que el resto son de clase económica. Cada silla puede ser asignada a un pasajero que cuenta con un nombre y una cédula. Este último dato es la entrada principal para poder consultar una reserva o eliminarla del sistema. Cuando se asigna una silla es necesario conocer las preferencias del usuario. Este puede elegir la posición de la silla, ventana, pasillo o centro, y la clase, ejecutiva o económica. En el caso especial de las sillars ejecutivas, solo es posible elegir las posiciones: ventana o pasillo. Las sillars son asignadas de forma secuencial según su ubicación y su clase. De igual forma, el programa permite buscar la reserva de un pasajero y visualizar los datos de la reserva. El programa debe permitir al usuario:

1. Asignar una silla a un pasajero
2. Eliminar reserva
3. Buscar pasajero
4. Calcular el porcentaje de ocupación del avión

Enunciado (programa modificado):

Se quiere crear un programa que tenga como objetivo el manejo de reservas de un avión. El programa debe estar diseñado para que un funcionario de la aerolínea lo opere, el funcionario tendrá que estar registrado con un nombre de usuario y una contraseña que estarán guardados en un archivo de texto txt, para ingresar al programa, el funcionario deberá digitar el nombre de usuario y contraseña correspondientes. Este avión cuenta con un número fijo de 14 sillas. De ellas, 6 son de clase ejecutiva, mientras que el resto son de clase económica. Cada silla puede ser asignada a un pasajero que cuenta con un nombre y una cédula. Este último dato es la entrada principal para poder consultar una reserva o eliminarla del sistema. Cuando se asigna un asiento se conocen automáticamente las preferencias del usuario ya que este escoge el asiento que quiere ocupar, esto será posible siempre y cuando el asiento este libre, dando una mejor experiencia hacia el usuario de la aerolínea al momento de ser asignado a un asiento. Deben existir dos formas de ser asignado a un asiento, dando clic en el botón del panel que está destinado a esto y digitando el número de asiento al que se desea ser asignado y la segunda es dando clic (en el mapa de asientos) sobre cualquiera de los asientos disponibles e ingresando los datos del pasajero. De igual forma, el programa permite buscar la reserva de un pasajero y visualizar los datos de la reserva. El programa debe permitir al usuario:

1. Asignar una silla a un pasajero
2. Eliminar reserva
3. Buscar pasajero
4. Calcular el porcentaje de ocupación del avión

Mejoras respecto al programa original:

1. Se añade un menú con dos opciones donde el funcionario de la aerolínea (quien maneja el programa) escoge: 1. Si desea registrar un nuevo funcionario con nombre de usuario y contraseña, esta información es guardada en un archivo de texto plano o 2. Si desea ingresar al programa teniendo que digitar correctamente su usuario y contraseña, de lo contrario no será posible hacerlo.

2. La selección del asiento que desea el pasajero no se realiza por criterios del programa, sino según lo que desea el pasajero y dependiendo de los asientos libres que hay, siendo muy intuitivo al presentar dichos asientos en forma de botones en el mapa de asientos.
3. Hay dos formas de escoger y registrar un usuario a un asiento, desde el mapa de asientos o desde el botón en el panel inferior.
4. El programa original tiene una arquitectura modelo – vista, en este se modifica y se pasa a modelo – vista – controlador.

Análisis conceptual:

¿Qué es aerolínea?

Una aerolínea, también conocida como línea aérea o compañía aérea, se encarga del traslado de pasajeros, carga y en ocasiones, animales, utilizando vehículos aéreos como aviones o helicópteros.

El ámbito de las aerolíneas es diverso, con empresas que operan vuelos regulares para transportar pasajeros y carga, así como otras que proporcionan servicios de transporte puntual acordados con sus clientes o grupos específicos, denominadas "chárter".

<https://es.wikipedia.org/wiki/Aerol%C3%ADnea>

¿Qué es pase de abordaje?

El pase de abordar es el comprobante que se otorga para el vuelo, ya sea en formato electrónico (que se presenta en dispositivos móviles como celulares o tabletas) o físico. Es esencial para acceder al área de seguridad y para abordar la aeronave.

<https://www.mundukos.com/aeropuertos/que-es-el-check-in-y-el-pase-de-abordar/>

¿Qué es primera clase?

La zona de primera clase en un avión de reacción suele ubicarse en la parte delantera de la aeronave. En muchos casos, las aerolíneas han optado por eliminar completamente la primera clase en sus servicios internacionales, ofreciendo la clase ejecutiva como el nivel más alto de servicio. Los pasajeros de primera clase suelen tener acceso a áreas restringidas especiales en los aeropuertos mientras esperan sus vuelos.

https://es.wikipedia.org/wiki/Primera_clase#:~:text=Se%20denomina%20primera%20clase%20a,aviones%20o%20autobuses%20entre%20otros.

¿Qué es clase económica?

La clase turista o económica en un avión es la opción de menor costo, con niveles de confort más limitados en comparación con otras clases. Se destaca por la proximidad entre los asientos y una selección reducida de comidas y entretenimiento.

En las propuestas de cambio recientes por parte de las aerolíneas, se contempla la implementación de asientos verticales con el objetivo de reducir los precios de los boletos.

https://es.wikipedia.org/wiki/Cabina_de_avi%C3%B3n#:~:text=La%20clase%20turista%20o%20econ%C3%B3mica,de%20las%20comidas%20y%20entretenimiento.

Requerimientos funcionales:

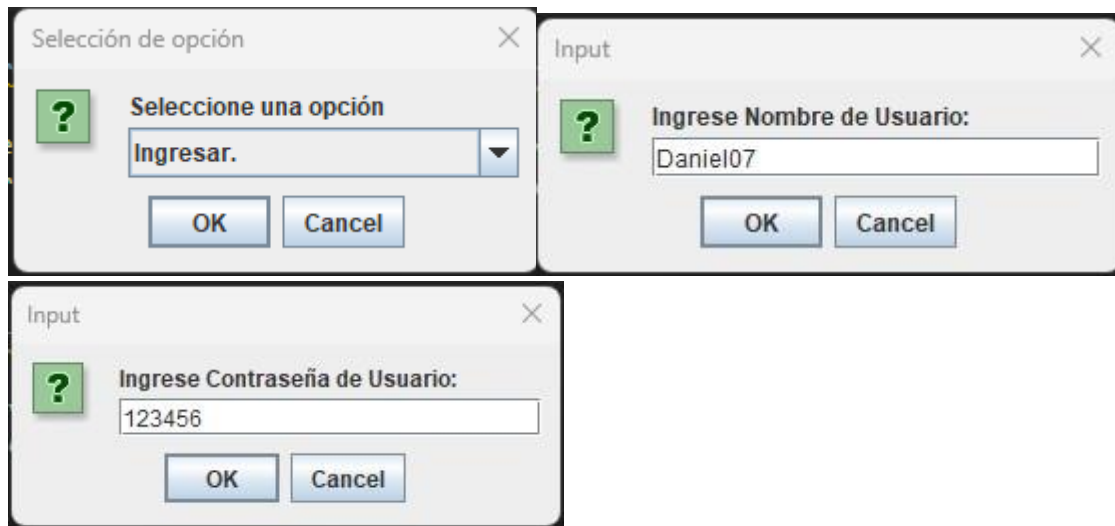
Nombre	R1. Menú de entrada.
Resumen	Debe mostrar un menú con las opciones para ingresar o registrar nuevo funcionario.
Entradas:	
1. Ingresar 2. Registrar	
Resultados:	
1. (Ventana que pide nombre de usuario) - (ventana que pide contraseña) 2. (Ventana que pide nombre de usuario a registrar) - (Ventana que pide contraseña de usuario a ingresar).	

1:






2:




Nombre	R2. Asignar asiento a pasajero.
Resumen	Se asigna un asiento al pasajero según el asiento escogido, ya sea por medio del mapa de asientos o el botón registrar.
Entradas:	
<ul style="list-style-type: none"> - Número de identificación pasajero. - Nombre de pasajero. 	
Resultados:	
<ul style="list-style-type: none"> - Se desactiva el botón de asiento correspondiente al escogido y se muestra un pase de abordaje con la información del pasajero y el asiento asignado junto con la clase a la que pertenece dicho asiento. 	

Caso 1: registro exitoso.

POO AIRLINE



POO AIRLINE



Registro de pasajero

Datos del pasajero

Cédula

1000471135

Nombre

Daniel Barbosa

Aceptar

Cancelar

Registrar Pasajero

Eliminar Pasajero

Buscar Pasajero

Porcentaje Ocupación

Opción 1

Opción 2



POO AIRLINE



Datos del pasajero

Datos del pasajero

Cédula:	1000471135
Nombre:	Daniel Barbosa
Silla:	1
Clase:	Ejecutiva
Ubicación:	Izquierda



Registrar Pasajero

Eliminar Pasajero

Buscar Pasajero

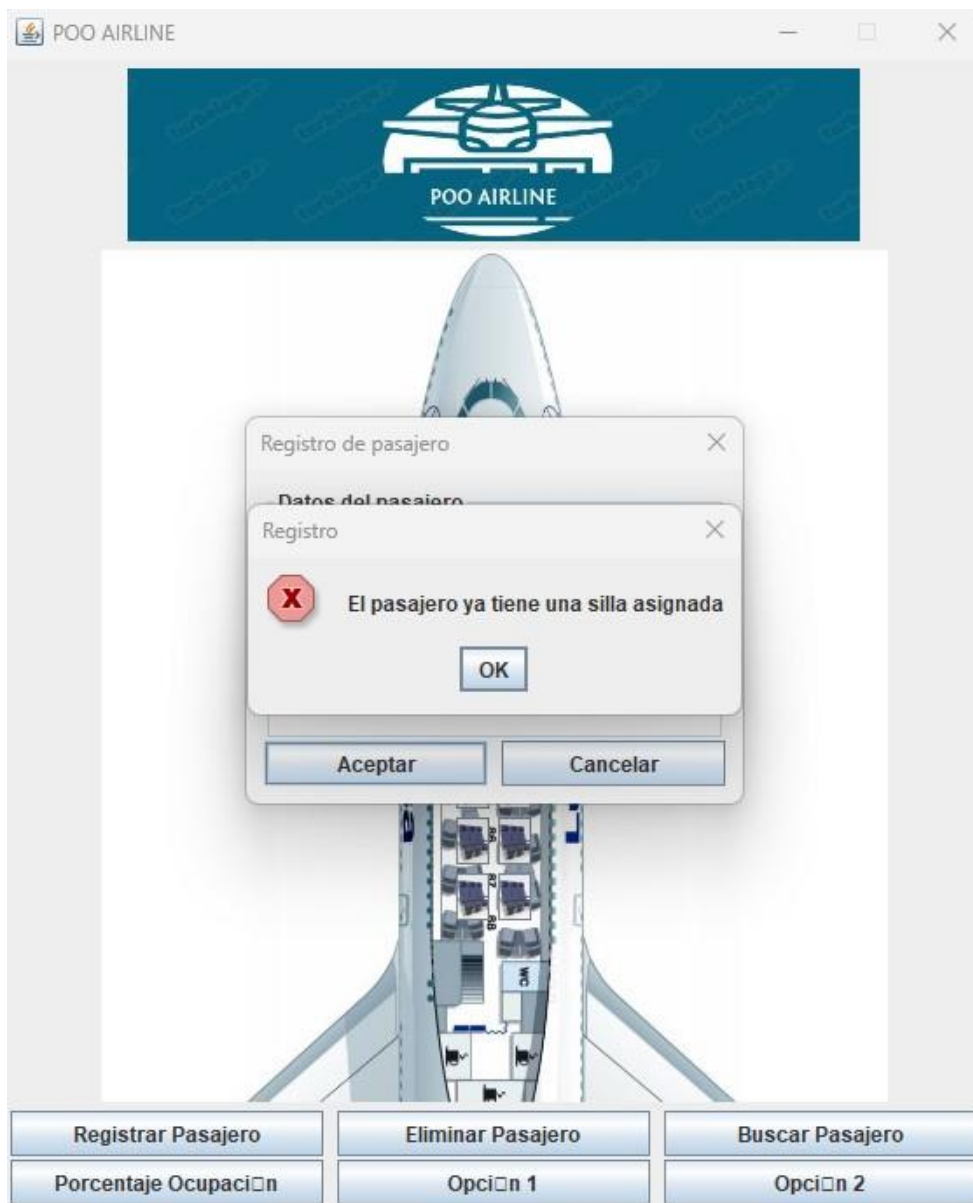
Porcentaje Ocupación

Opción 1

Opción 2

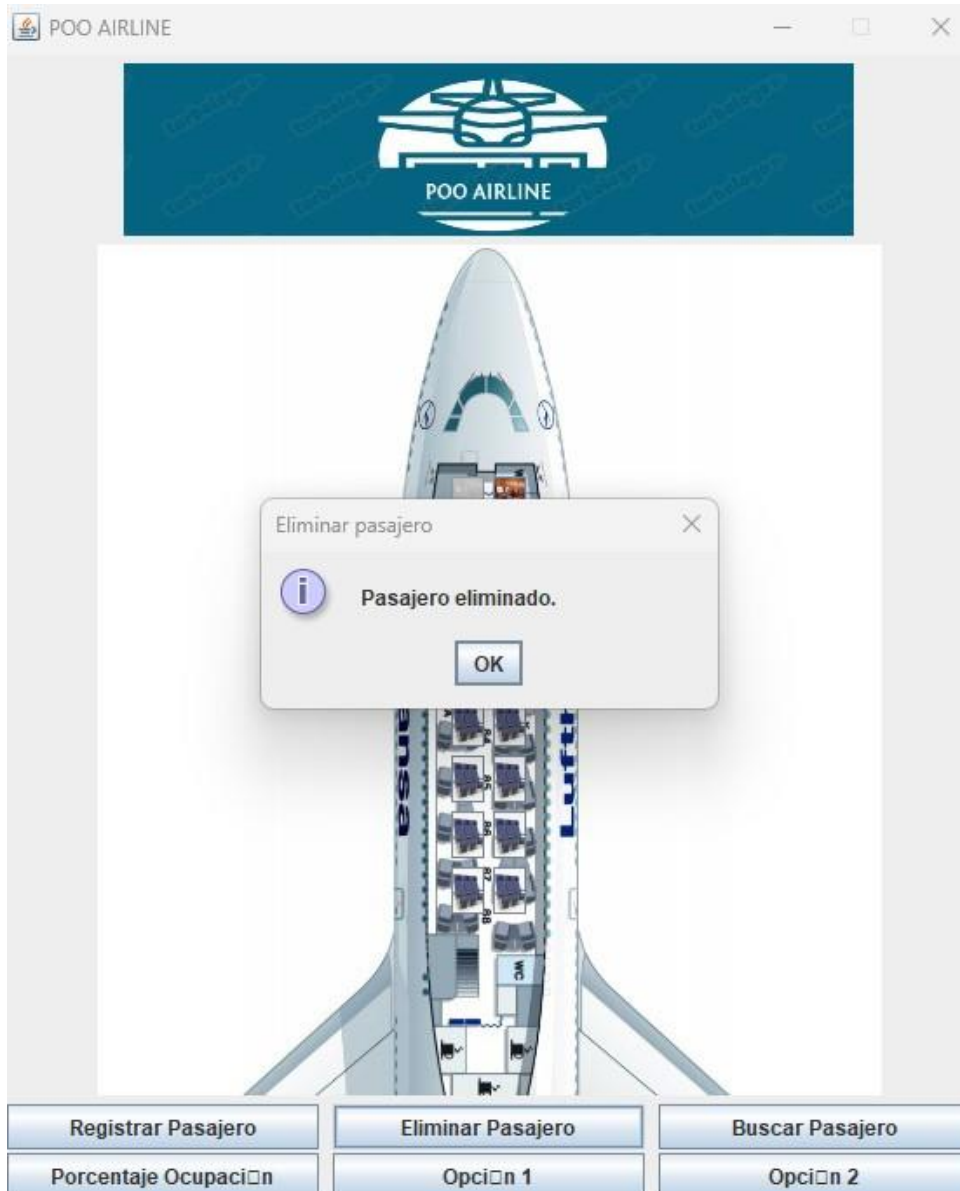


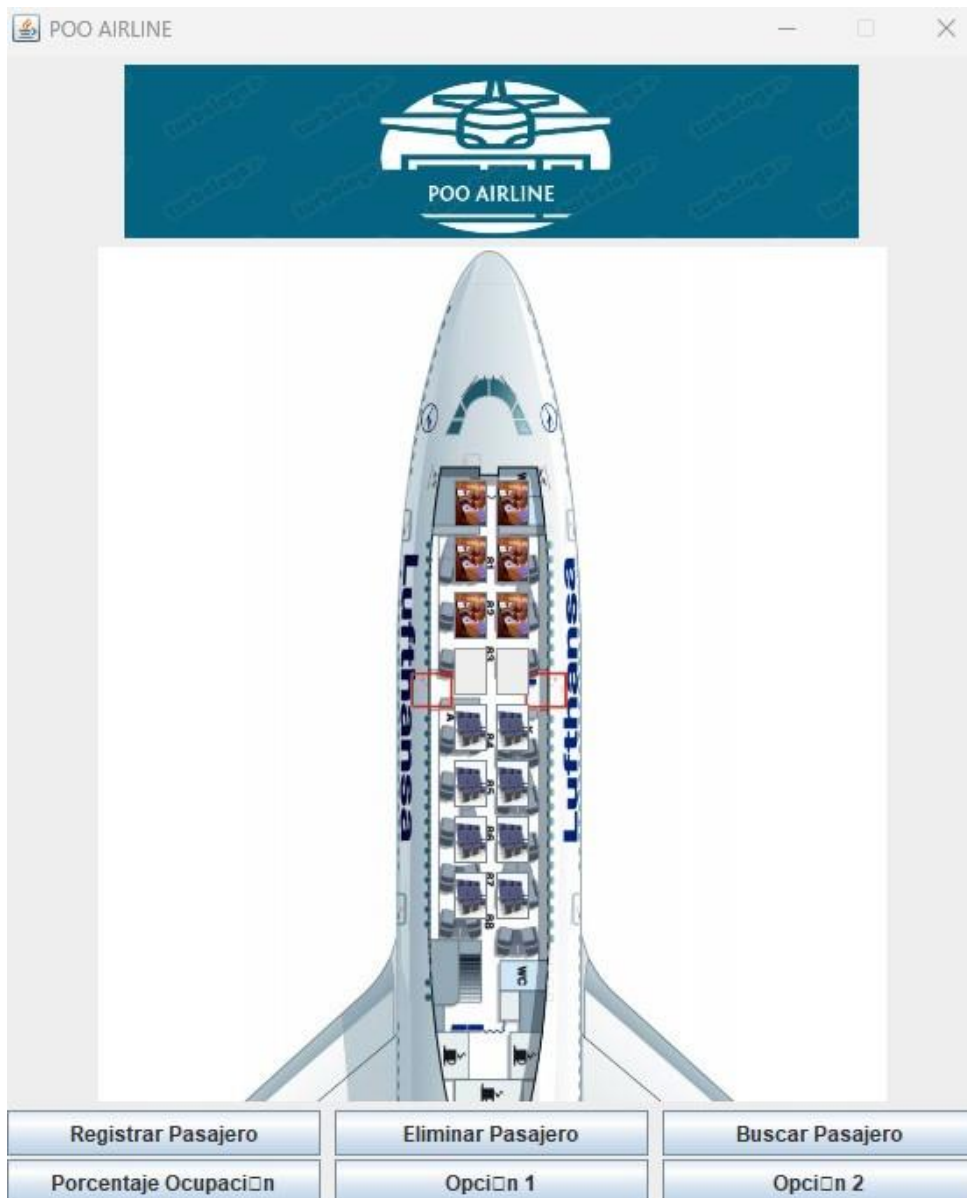
Caso 2: usuario repetido.



Nombre	R3. Eliminar reserva.
Resumen	Elimina el registro de información del pasajero y de su reserva dada su número de identificación.
Entradas:	
Número de identificación del pasajero	
Resultados:	
Se marca como disponible la silla asignada al pasajero y se activa el boton. En caso de que no haya un pasajero con la cédula dada, se muestra un mensaje de error.	









Nombre	R4. Buscar pasajero.
Resumen	Busca un pasajero y su reserva dado su número de identificación.
Entradas:	
- Número de identificación del pasajero	
Resultados:	
- Se muestra un diálogo con la información del pasajero y la reserva. En caso de que no exista, se muestra un mensaje de error	

Caso 1: búsqueda exitosa.


POO AIRLINE



POO AIRLINE



Eliminar pasajero



Ingrese el número de cédula

OK

Cancel

Registrar Pasajero

Eliminar Pasajero


Buscar Pasajero

Porcentaje Ocupación


Opción 1

Opción 2

POO AIRLINE



POO AIRLINE



Datos del pasajero

Datos del pasajero

Cédula: 1000471135

Nombre: Daniel Barbosa

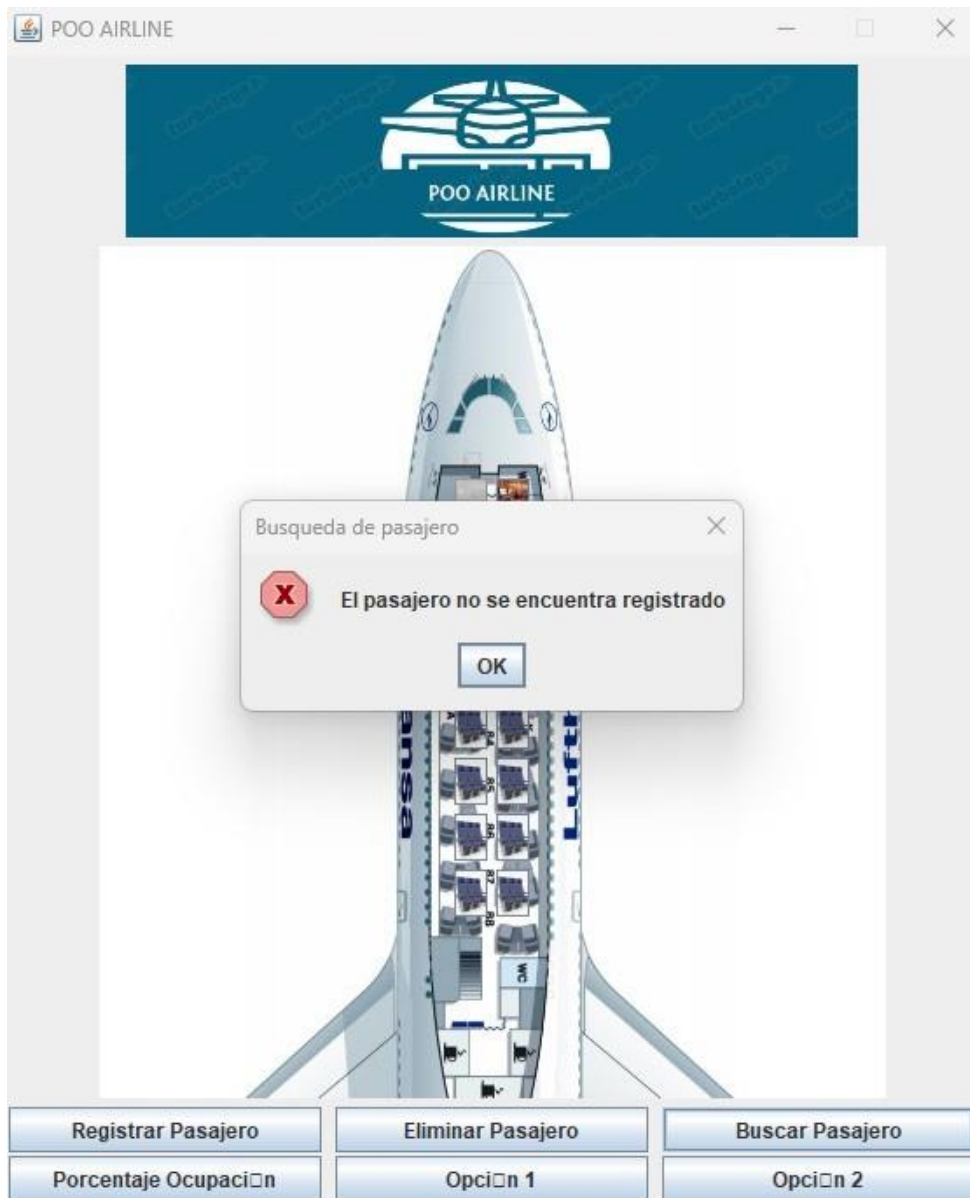
Silla: 1

Clase: Ejecutiva

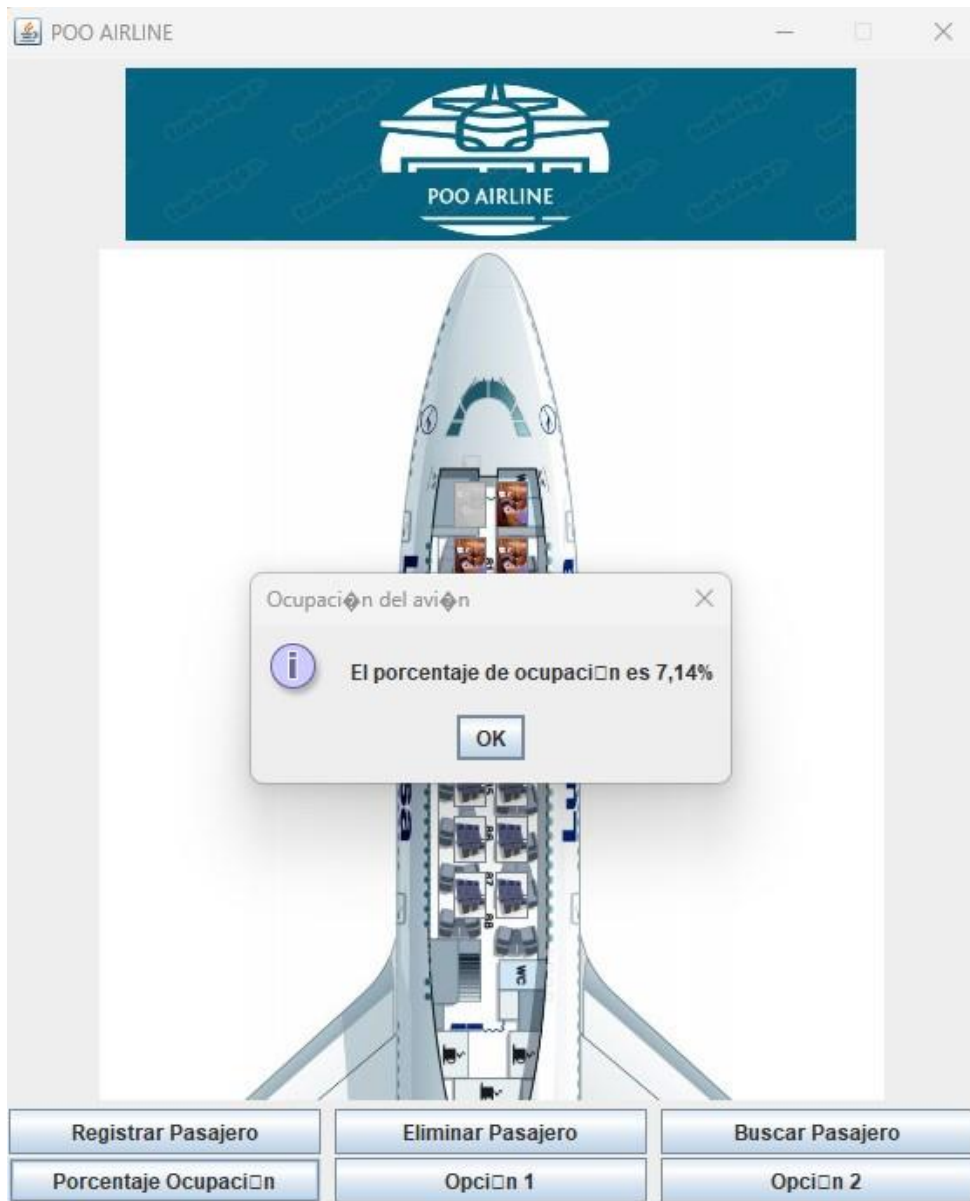
Ubicación: Izquierda

Registrar Pasajero	Eliminar Pasajero	Buscar Pasajero
Porcentaje Ocupación	Opción 1	Opción 2

Caso 2: Pasajero no existe.

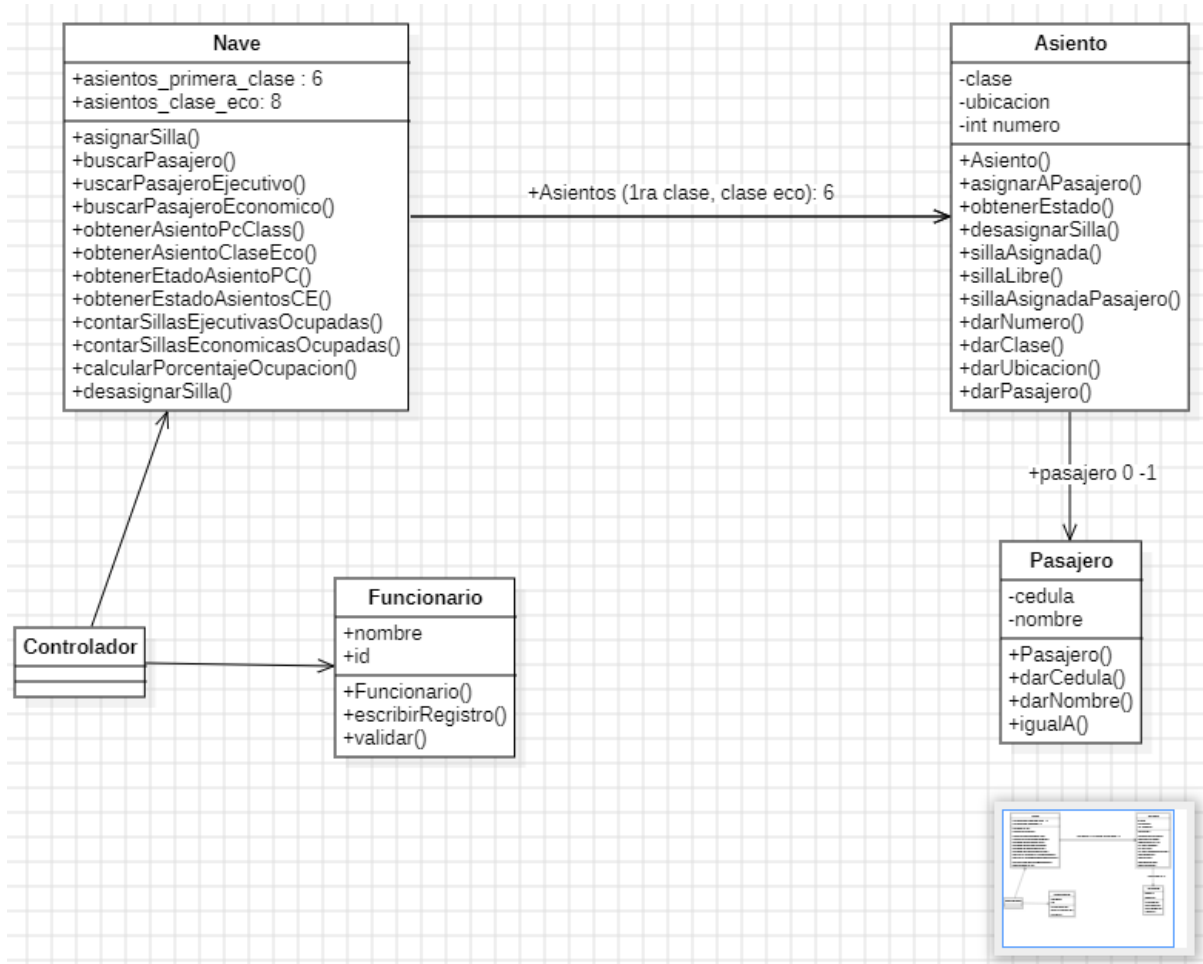


Nombre	R5. Calcular porcentaje de ocupación.
Resumen	Calcula el porcentaje de sillas asignadas con respecto al número total de sillas disponibles en el avión.
Entradas:	
- Ninguna	
Resultados:	
- Muestra un diálogo con el porcentaje de ocupación del avión	

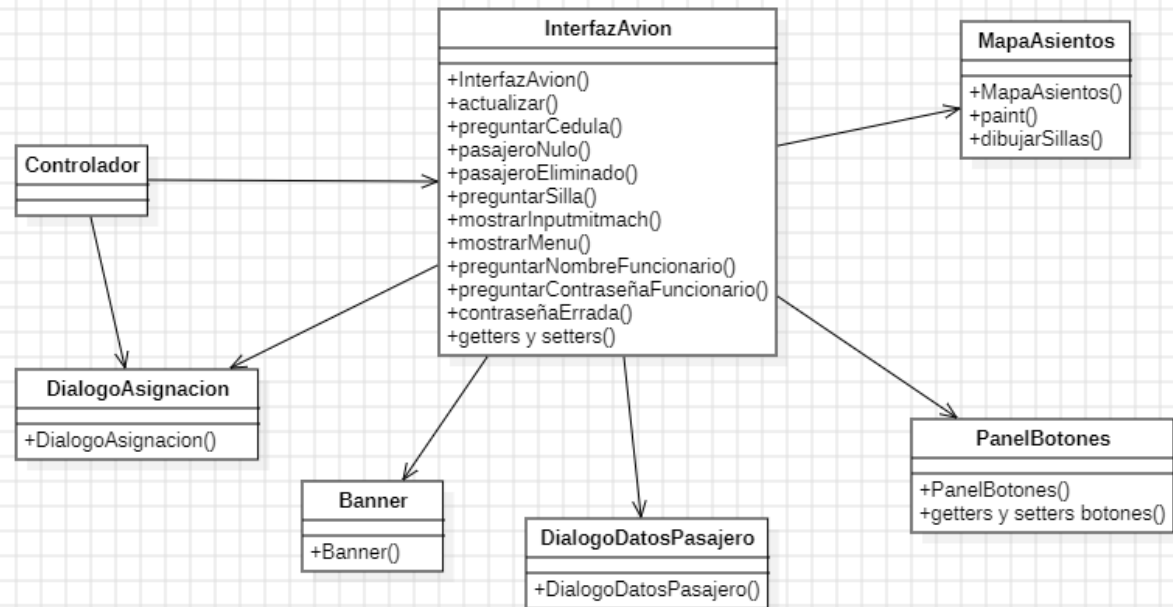


Diseño:

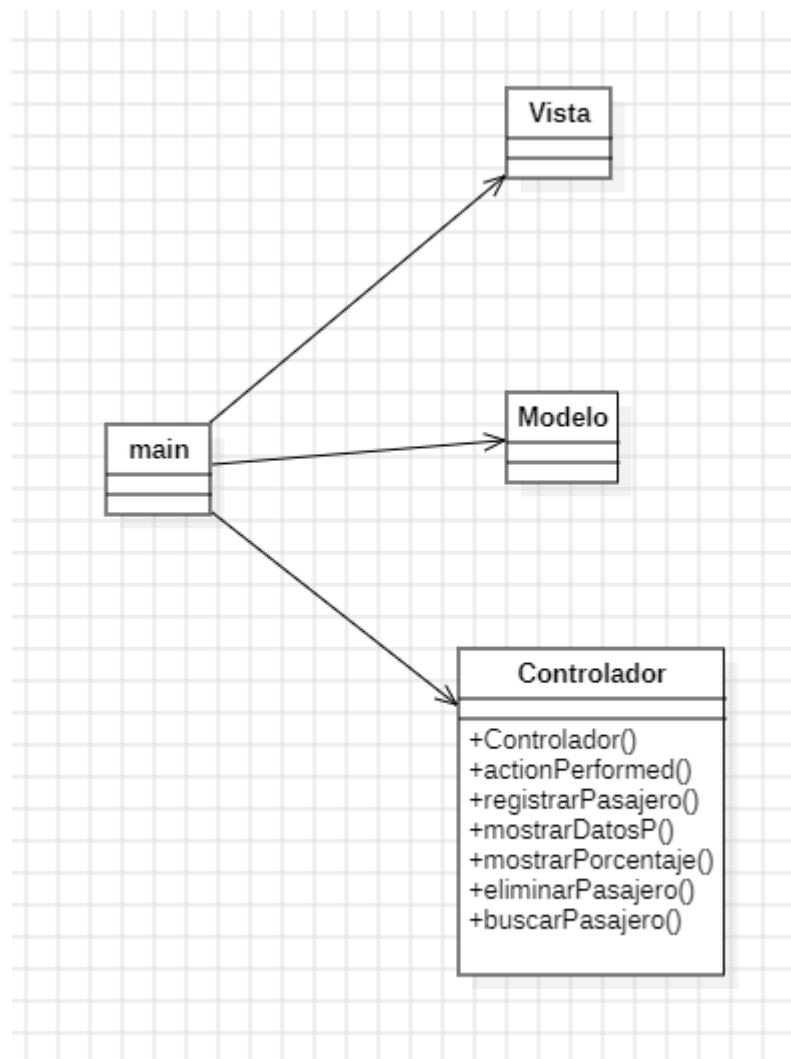
Modelo:



Vista:



Controlador:



Implementacion:

Modelo:

Clase Nave.

```
package modelo;

import modelo.Asiento.Clase;
import modelo.Asiento.Ubicacion;

public class Nave {
    //_____
    //Constantes
    //-----

    public final static int asientos_primera_clase = 6;

    public final static int asientos_clase_eco = 8;

    //-----
    //Atributos
    //-----

    /*
     * asientos primera clase
     */
    private Asiento [] asientosPrimeraClase = new
Asiento[asientos_primera_clase];

    /*
     * asientos clase economica
     */

    private Asiento [] asientosClaseEco = new Asiento[asientos_clase_eco];

    public Nave(){

        //Variable que controla el for
        int i;
```

```

        //Añade los asientos de p clase al arreglo y les da ubicacion

        for (i = 0; i < asientos_primera_clase; i++){
            if (i % 2 == 0){
                asientosPrimeraClase[i] = new Asiento(i+1, Clase.PCLASE,
Ubicacion.DERECHA);
            }else if (i % 2 == 1 ){
                asientosPrimeraClase[i] = new Asiento(i+1, Clase.PCLASE,
Ubicacion.IZQUIERDA);
            }
        }

        //Añade los asientos de clase eco al arreglo y les da ubicacion

        for (i = 0; i < asientos_clase_eco; i++){
            if (i % 2 == 0){
                asientosClaseEco[i] = new Asiento(i+7, Clase.ECOCLASE,
Ubicacion.DERECHA);
            }else if (i % 2 == 1 ){
                asientosClaseEco[i] = new Asiento(i+7, Clase.ECOCLASE,
Ubicacion.IZQUIERDA);
            }
        }

    }

    // -----
    // Métodos
    // -----

    /**
     * Asigna la silla al pasajero en la clase y ubicación especificados.
    <br>
     * <b>post: </b> Si existe una silla con la clase y la ubicación dada,
    el pasajero queda asignado en la primera de ellas según el orden numérico.
     *
     * @param pClase      Clase elegida por el pasajero. Clase pertenece
    {Clase.EJECUTIVA, Clase.ECONOMICA}.
     * @param pUbicacion Ubicación elegida por el pasajero. Si clase =
    Clase.ECONOMICA entonces ubicación pertenece {IZQUIERDA O DERECHA}, <br>
     *                      o si clase = Clase.EJECUTIVA entonces ubicación
    pertenece {VENTANA, PASILLO}.
     * @param pPasajero  Pasajero a asignar. pPasajero != null y no tiene
    silla en el avión.

```

```

    * @return Silla asignada al pasajero o null si no se pudo asignar una
    silla al pasajero en la ubicación y clase especificados.
    */
    @SuppressWarnings("null")
    public void asignarSilla(Pasajero pPasajero, int numSilla) {
        // evalua el numero del asiento y dependiendo de eso escoge en que
        arreglo posicion del arreglo buscarla
        if(numSilla <= 5){
            asientosPrimeraClase[numSilla-1].asignarAPasajero(pPasajero);

        }else {
            asientosClaseEco[numSilla-7].asignarAPasajero(pPasajero);
        }

    }

    /**
     * Busca un pasajero en el avión.
     *
     * @param pPasajero Pasajero a buscar. pPasajero != null.
     * @return Silla en la que se encontró el pasajero. Si no lo encuentra
     retorna null.
     */
    public Asiento buscarPasajero(Pasajero pPasajero) {
        // Busca el pasajero en ejecutiva
        Asiento silla = buscarPasajeroEjecutivo(pPasajero);
        // Si no estaba en ejecutiva
        if (null == silla) {
            // Busca en económica
            silla = buscarPasajeroEconomico(pPasajero);
        }

        return silla;
    }

    /**
     * Busca un pasajero en las sillas ejecutivas. <br>
     * <b>pre: </b> La lista de sillas ejecutivas está inicializada.
     *
     * @param pPasajero Pasajero a buscar. pPasajero != null.
     * @return Silla en la que se encontró el pasajero. Si no lo encuentra
     retorna null.
     */

```



```

    public Asiento buscarPasajeroEjecutivo(Pasajero pPasajero) {
        boolean encontrado = false;
        Asiento silla = null;
        for (int i = 0; i < asientos_primera_clase && !encontrado; i++) {
            silla = asientosPrimeraClase[i];
            if (silla.sillaAsignada() &&
silla.darPasajero().igualA(pPasajero)) {
                encontrado = true;
            }
        }
        if (!encontrado) {
            silla = null;
        }
        return silla;
    }

    /**
     * Busca un pasajero en las sillas económicas. <br>
     * <b>pre: </b> La lista de sillas económicas está inicializada.
     *
     * @param pPasajero Pasajero a buscar. pPasajero != null.
     * @return Silla en la que se encontró el pasajero. Si no lo encuentra
retorna null.
     */
    public Asiento buscarPasajeroEconomico(Pasajero pPasajero) {
        boolean encontrado = false;
        Asiento silla = null;
        for (int i = 0; i < asientos_primera_clase && !encontrado; i++) {
            silla = asientosClaseEco[i];
            if (silla.sillaAsignada() &&
silla.darPasajero().igualA(pPasajero)) {
                encontrado = true;
            }
        }
        if (!encontrado) {
            silla = null;
        }
        return silla;
    }

    /**
     * Devuelve arreglo de asientos primera clase.
     * @return arreglo de asientos p class.
     */
    public Asiento[] obtenerAsientoPclass (){

        return asientosPrimeraClase;
    }

```

```

/**
 * Devuelve arreglo de asientos clase economica.
 * @return arreglo de asientos clase eco.
 */
public Asiento[] obtenerAsientosClaseEco (){

    return asientosClaseEco;
}

public boolean obtenerEstadoAsientoPC (int numSilla){

    boolean estado = asientosPrimeraClase[numSilla-1].obtenerEstado();

    return estado;

}

public boolean obtenerEstadoAsientosCE (int numSilla){

    //NUMERO DE SILLA - 7 PARA QUE DE EL NUMERO EN QUE SE POSICIONA
    DENTRO DEL ARREGLO
    boolean estado = asientosClaseEco[numSilla-7].sillaLibre();

    return estado;
}
/**
 * Retorna el número de sillas ejecutivas ocupadas. <br>
 * <b>pre: </b> La lista de sillas ejecutivas está inicializada.
 * @return Número de silla ejecutivas ocupadas.
 */
public int contarSillasEjecutivasOcupadas( )
{
    int contador = 0;
    for( Asiento sillaEjecutiva : asientosPrimeraClase )
    {
        if( sillaEjecutiva.sillaAsignada( ) )
        {
            contador++;
        }
    }
    return contador;
}

/**

```

```

    * Retorna el número de sillas económicas ocupadas. <br>
    * <b>pre: </b> La lista de sillas económicas está inicializada.
    * @return Número de sillas económicas ocupadas.
    */
    public int contarSillasEconomicasOcupadas( )
    {
        int contador = 0;
        for( Asiento sillaEconomica : asientosClaseEco )
        {
            if( sillaEconomica.sillaAsignada( ) )
            {
                contador++;
            }
        }
        return contador;
    }

    /**
    * Calcula el porcentaje de ocupación del avión.
    * @return Porcentaje total de ocupación.
    */
    public double calcularPorcentajeOcupacion( )
    {
        double porcentaje;
        int totalSillas = asientos_clase_eco + asientos_primera_clase;
        int sillasOcupadas = contarSillasEconomicasOcupadas( ) +
        contarSillasEjecutivasOcupadas( );
        porcentaje = ( double )sillasOcupadas / totalSillas * 100;
        return porcentaje;
    }

    /**
    * Desasigna la silla de un pasajero. <br>
    * <b>post: </b> Si se encuentra una silla con el pasajero, la silla
    quedara con su pasajero igual a null.
    * @param pPasajero Pasajero a retirar. pPasajero != null.
    * @return Retorna true si encontró el pasajero y desasigna la silla,
    false en caso contrario.
    */
    public boolean desasignarSilla( Pasajero pPasajero )
    {
        // Busca el pasajero en el avión
        Asiento silla = buscarPasajero( pPasajero );
        boolean resultado = false;
        // Si lo encuentra desasigna
        if( silla != null )
        {
            silla.desasignarSilla( );
        }
    }

```

```
        resultado = true;
    }
    return resultado;
}

}
```

Clase Asiento.

```
package modelo;

public class Asiento {

    //representa las clases del avion

    public enum Clase{
        PCLASE, ECOCLASE
    }

    //representa las ubicaciones de los asientos

    public enum Ubicacion{
        IZQUIERDA, DERECHA
    }

    //Atributos :

    private int numero;

    private Clase clase;

    private Ubicacion ubicacion;

    private Pasajero pasajero;

    //estado: true = disponible, false = ocupada
```

```

private boolean estado;

/**
 * Crea la silla con su número identificador. <br>
 * <b>post: </b> El objeto silla tiene sus datos básicos número, clase y
ubicación asignados. El pasajero no está asignado y tiene valor null.
 *
 * @param pNumero    Número de silla. pNumero > 0.
 * @param pClase      Clase de silla. pClase pertenece
{EJECUTIVA,ECONOMICA}.
 * @param pUbicacion Ubicación de la silla. pUbicacion pertenece
{VENTANA, CENTRAL, PASILLO}.
 */

public Asiento(int pNumero, Clase pClase, Ubicacion pUbicacion){
    numero = pNumero;
    clase = pClase;
    ubicacion = pUbicacion;
    // Inicialmente no hay ningún pasajero en la silla
    pasajero = null;
    estado = true;

}

//METODOS:

/**
 * Asigna la silla al pasajero recibido como parámetro. <br>
 * <b>post: </b> La silla queda asignada al pasajero recibido como
parámetro.
 *
 * @param pPasajero Pasajero a asignar en la silla. pPasajero !=null.
 */

public void asignarAPasajero(Pasajero pPasajero) {
    pasajero = pPasajero;
    estado = false;

}

public boolean obtenerEstado(){
    return estado;
}

```

```

    }

    /**
     * Desasigna la silla al pasajero. La silla queda nuevamente libre. <br>
     * <b>post: </b> La silla queda sin pasajero asignado.
     */
    public void desasignarSilla() {
        pasajero = null;
        estado = true;
    }

    /**
     * Indica si la silla está asignada.
     *
     * @return Retorna true si la silla esta asignada, false en caso
    contrario.
     */
    public boolean sillaAsignada() {
        boolean asignada = true;
        if (null == pasajero) {
            asignada = false;
        }
        return asignada;
    }
    public boolean sillaLibre() {
        boolean libre = true;
        if (pasajero != null) {
            libre = false;
        }
        return libre;
    }
    /**
     * Indica si la silla está asignada al pasajero recibido como parámetro.
     *
     * @param pPasajero Pasajero a comparar con el de la silla.
     * @return Retorna true si el pasajero ocupa la silla, false si la silla
    está vacía o no coincide con el pasajero recibido como parámetro.
     */
    public boolean sillaAsignadaPasajero(Pasajero pPasajero) {
        boolean asignadaPasajero = false;
        if (null == pasajero) {
            asignadaPasajero = false;
        } else if (pasajero.igualA(pPasajero)) {
            asignadaPasajero = true;
        }

        return asignadaPasajero;
    }

```

```

    }

    /**
     * Retorna el número de la silla.
     *
     * @return Número de la silla.
     */
    public int darNumero() {
        return numero;
    }

    /**
     * Retorna la clase de la silla.
     *
     * @return Clase de la silla.
     */
    public Clase darClase() {
        return clase;
    }

    /**
     * Retorna la ubicación de la silla.
     *
     * @return Ubicación de la silla.
     */
    public Ubicacion darUbicacion() {
        return ubicacion;
    }

    /**
     * Retorna el pasajero asignado a la silla.
     *
     * @return Pasajero asignado a la silla. Si no hay pasajero retorna
    null.
     */
    public Pasajero darPasajero() {
        return pasajero;
    }
}

```

Clase Pasajero.

```

package modelo;

public class Pasajero {

    // -----
    // Atributos
    // -----

    /**
     * Cédula del pasajero.
     */
    private String cedula;

    /**
     * Nombre del pasajero.
     */
    private String nombre;

    // -----
    // Constructores
    // -----

    /**
     * Crea un pasajero con su cédula y nombre. <br>
     * <b>post: </b> El pasajero tiene sus datos básicos cédula y nombre
asignados.
     *
     * @param pCedula Cédula del pasajero. pCedula > 0.
     * @param pNombre Nombre del pasajero. pNombre != null && pNombre !=
""
     */
    public Pasajero(String pCedula, String pNombre) {
        cedula = pCedula;
        nombre = pNombre;
    }

    // -----
    // Métodos
    // -----

    /**
     * Retorna la cédula del pasajero.

```



```

        *
        * @return La cédula del pasajero.
        */
    public String darCedula() {
        return cedula;
    }

    /**
     * Retorna el nombre del pasajero.
     *
     * @return El nombre del pasajero.
     */
    public String darNombre() {
        return nombre;
    }

    /**
     * Indica si el pasajero es igual a otro.
     *
     * @param pOtroPasajero Pasajero a comparar. pOtroPasajero != null.
     * @return Retorna true si es el mismo pasajero, false en caso
    contrario.
     */

    public boolean igualA(Pasajero pOtroPasajero) {
        boolean esIgual = false;
        if (cedula.equals(pOtroPasajero.darCedula())) {
            esIgual = true;
        }

        return esIgual;
    }
}

```

Clase Funcionario.

```

package modelo;

import java.io.*;
import java.util.*;

```

```

public class Funcionario {

// -----
// Atributos
// -----

    /**
     * Nombre del funcionario.
     */
    private String nombre;

    /**
     * Contraseña del funcionario.
     */
    private String id;

// -----
// Constructores
// -----

    /**
     * Crea un pasajero con su cédula y nombre. <br>
     * <b>post: </b> El pasajero tiene sus datos básicos cédula y nombre
asignados.
     *
     * @param pNombre Nombre del funcionario pNombre != null. && pNombre !=
"".
     * @param pContraseña Contraseña del pasajero. pContraseña != null. &&
pContraseña != "".
     */
    public Funcionario(String pNombre, String pContraseña ){

        nombre = pNombre;
        id = pContraseña;

    }

    public void escribirRegistro() {
        FileWriter fw=null;
        PrintWriter pw=null;

```

```

        try{
            fw=new FileWriter("funcionarios.txt",true);
            pw=new PrintWriter(fw);
            pw.println(nombre+";"+id);
        }catch (Exception n){
            n.printStackTrace();
        }
    finally{
        try{
            if (null!=fw)
                fw.close();
        }
        catch (Exception e2){
            e2.printStackTrace();
        }
    }
}

/**
 *valida que los datos de nombre y contraseña sean correctos
 *
 * @return true si son correctos los datos.
 * @return false si no son correctos los datos.
 */
public boolean validarDatos (){

    //leer usuarios y contraseñas del archivo para almacenarlos en un
hash map

    Map<String, String> funcionarios = new HashMap<>();

    try (BufferedReader br = new BufferedReader(new
FileReader("funcionarios.txt"))){
        String line;
        while((line = br.readLine())!= null){
            String[] parts = line.split(";");
            if(parts.length == 2){
                String nombreUsuario = parts[0].trim();
                String contraseña = parts[1].trim();
                funcionarios.put(nombreUsuario, contraseña);
            }
        }
    }catch(IOException e){
        e.printStackTrace();
    }
}

```

```

        if(funcionarios.containsKey(nombre) &&
funcionarios.get(nombre).equals(id)){
            return true;
        }else{
            return false;
        }
    }
}

```

Vista:

Clase InterfazAvion:

```

package vista;

import java.awt.*;
import java.text.*;
import javax.swing.*;

import modelo.Nave;

/**
 * Ventana principal del avión.
 */
@SuppressWarnings("serial")
public class InterfazAvion extends JFrame
{

    private Nave nave;
    private Banner banner;
    private PanelBotones panelBotones;
    private MapaAsientos mapaAsientos;

    //private DialogoDatosPasajero datosPasajero;
}

```

```
public InterfazAvion () {

    setTitle( "POO AIRLINE" );
    setSize( 580, 700 );
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

    //crea la nave
    nave = new Nave();

    // Configura la interfaz
    setLayout( new BorderLayout( ) );

    // Panel del banner
    banner = new Banner( );
    panelBotones = new PanelBotones();
    mapaAsientos = new MapaAsientos( nave );

    add( banner, BorderLayout.NORTH );
    add( mapaAsientos, BorderLayout.CENTER);
    add( panelBotones, BorderLayout.SOUTH );

    setResizable( false);
    setLocationRelativeTo( null );
}

public Banner getBanner() {
    return banner;
}

public void setBanner(Banner banner) {
    this.banner = banner;
}

public PanelBotones getPanelBotones() {
```

```

        return panelBotones;
    }

    public void setPanelBotones(PanelBotones panelBotones) {
        this.panelBotones = panelBotones;
    }

    public MapaAsientos getMapaAsientos() {
        return mapaAsientos;
    }

    public void setMapaAsientos(MapaAsientos mapaAsientos) {
        this.mapaAsientos = mapaAsientos;
    }

    public void actualizar( Nave avion)
    {
        remove( mapaAsientos );

        // Panel del avión
        mapaAsientos = new MapaAsientos( avion );
        add( mapaAsientos, BorderLayout.CENTER );
        validate( );
    }

    public void mostrarPorcentaje (double porcentaje){
        DecimalFormat df = ( DecimalFormat )NumberFormat.getInstance( );
        df.applyPattern( "###.##" );
        JOptionPane.showMessageDialog( this, "El porcentaje de ocupación es " + df.format( porcentaje ) + "%", "Ocupación del avión",
        JOptionPane.INFORMATION_MESSAGE );
    }

    /**
     * pregunta por la cedula del pasajero.
     * @return retorna el numero de cedula.
     */

```

```

public String preguntarCedula (){

    String cedula = JOptionPane.showInputDialog( this, "Ingrese el
número de cédula", "Eliminar pasajero", JOptionPane.QUESTION_MESSAGE );

    return cedula;

}

/**
 * muestra un mensaje en el que informa que el pasajero no esta
registrado en ninguna silla.
 */
public void pasajeroNulo (){

    JOptionPane.showMessageDialog( this, "El pasajero no se encuentra
registrado", "Busqueda de pasajero", JOptionPane.ERROR_MESSAGE );

}

/**
 * Muestra mensaje que cuenta la eliminacion satisfactoria de pasajero.
 */
public void pasajeroEliminado (){

    JOptionPane.showMessageDialog( this, "Pasajero eliminado.",
"Eliminar pasajero", JOptionPane.INFORMATION_MESSAGE );

}

/**
 * pide el numero de silla deseada.
 * @return retorna el numero de silla a la que se asignara el pasajero.
 */
public int preguntarSilla (){

    int numSilla = Integer.parseInt(JOptionPane.showInputDialog("Ingrese
numero de silla deseada: "));

    return numSilla;

}

/**
 * Muestra mensaje de error.
 */
public void mostrarInputmitmach(){
    JOptionPane.showMessageDialog(null, "Ingrese una opcion
valida!!!", "Error", JOptionPane.ERROR_MESSAGE);
}

```

```

    }

    /**
     * Muestra menu para acceder al programa.
     * @return retorna la opcion escogida por el usuario
     */
    public String mostrarMenu(){

        String opc = (JOptionPane.showInputDialog(null,"Seleccione una
opción ","Selección de opción",
                JOptionPane.QUESTION_MESSAGE,null,new
Object[] {"Ingresar. ","Registrar nuevo funcionario."},
                "Seleccionar")).toString();

        return opc;

    }

    /**
     * Pregunta el nombre de usuario que va a tener el funcionario que
controla eel programa.
     * @return nombre del funcionario para ser guardado en el archivo txt.
     */
    public String preguntarNombreFuncionario(){

        String nombre = JOptionPane.showInputDialog(null,"Ingrese Nombre de
Usuario: ");

        return nombre;

    }

    /**
     * Pregunta el nombre de usuario que va a tener el funcionario que
controla eel programa.
     * @return nombre del funcionario para ser guardado en el archivo txt.
     */
    public String preguntarContraseñaFuncionario(){

        String contraseña = JOptionPane.showInputDialog(null,"Ingrese
Contraseña de Usuario: ");

        return contraseña;

    }

    /**
     * Muestra mensaje de error al ingresar contraseña.
     */
    public void contraseñaErrada(){
        JOptionPane.showMessageDialog(null, "El nombre de usuario y/o
contraseña son incorrectos!","Error",JOptionPane.ERROR_MESSAGE);
    }

```



```

    }
    public void mostrarRegistroExitoso () {
        JOptionPane.showMessageDialog(null, "Registro de funcionario
exitoso", "Registro", JOptionPane.INFORMATION_MESSAGE);
    }
}

```

Clase MapaAsientos.

```

package vista;

import modelo.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * Panel para el dibujo del avión.
 */
@SuppressWarnings("serial")
public class MapaAsientos extends JPanel
{

    /**
     * Interfaz principal.
     */
    private InterfazAvion ventana;

    //private Nave nave;
    private Nave avion;
    private ImageIcon imagen;

```

```

        JButton asiento1 = new JButton();
        JButton asiento2 = new JButton();
        JButton asiento3 = new JButton();
        JButton asiento4 = new JButton();
        JButton asiento5 = new JButton();
        JButton asiento6 = new JButton();
        JButton asiento7 = new JButton();
        JButton asiento8 = new JButton();
        JButton asiento9 = new JButton();
        JButton asiento10 = new JButton();
        JButton asiento11 = new JButton();
        JButton asiento12 = new JButton();
        JButton asiento13 = new JButton();
        JButton asiento14 = new JButton();

// Nave nave = new Nave();

//DialogoAsignacion dialogo = new DialogoAsignacion( ventana, nave);

// -----
// Constructores
// -----

/**
 * Crea el panel del avión. <br>
 * <b>post: <b> Se dibujan el avión y todas las sillas.
 * @param pAvion Avión que se va a dibujar. pAvion != null.
 */
public MapaAsientos( Nave pAvion )
{

    super( new BorderLayout() );

    avion = pAvion;

    imagen = new ImageIcon( "../Imagenes/mapa_asientos_def.png" );
    setPreferredSize( new Dimension( imagen.getIconWidth( ),
imagen.getIconHeight( ) ) );
    setOpaque( false );

    dibujarSillas();

```

```

    }

    /**
     * Dibuja la gr fica del avi n.
     * @param pGrafica Gr ficos del avi n. pGrafica != null
     */
    public void paint( Graphics pGrafica )
    {
        pGrafica.drawImage( imagen.getImage( ), 57, 0, null, null );
        super.paint( pGrafica );
    }

    private void dibujarSillas()
    {

        int anchoBoton = 20;
        int altoBoton = 25;

        // Configura propiedades visuales
        setLayout( new GridLayout( 8, 2, 5, 5 ) );
        setBorder( new EmptyBorder( 130, 260, 100, 260 ) );

        ImageIcon asientoCEIcon = new
        ImageIcon("./Imagenes/asiento_clase_economica.jpg");
        ImageIcon asientoPCIcon = new
        ImageIcon("./Imagenes/asiento_primera_clase.jpg");

        asiento1.setBounds(0, 0, anchoBoton, altoBoton);
        asiento1.setEnabled(avion.obtenerEstadoAsientoPC(1));
        asiento1.setIcon(new
        ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento1.getWidth(),
        asiento1.getHeight(), Image.SCALE_SMOOTH)));
        //asiento1.setIcon(new
        ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento1.getWidth(),
        asiento1.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento1);
    }

```

```
asiento2.setEnabled(avion.obtenerEstadoAsientoPC(2));
asiento2.setBounds(0, 0, anchoBoton, altoBoton);
asiento2.setIcon(new
ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento2.getWidth(),
asiento2.getHeight(), Image.SCALE_SMOOTH)));
add(asiento2);
```

```
asiento3.setEnabled(avion.obtenerEstadoAsientoPC(3));
asiento3.setBounds(0, 0, anchoBoton, altoBoton);
asiento3.setIcon(new
ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento3.getWidth(),
asiento3.getHeight(), Image.SCALE_SMOOTH)));
add(asiento3);
```

```
asiento4.setEnabled(avion.obtenerEstadoAsientoPC(4));
asiento4.setBounds(0, 0, anchoBoton, altoBoton);
asiento4.setIcon(new
ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento4.getWidth(),
asiento4.getHeight(), Image.SCALE_SMOOTH)));
add(asiento4);
```

```
asiento5.setEnabled(avion.obtenerEstadoAsientoPC(5));
asiento5.setBounds(0, 0, anchoBoton, altoBoton);
asiento5.setIcon(new
ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento5.getWidth(),
asiento5.getHeight(), Image.SCALE_SMOOTH)));
add(asiento5);
```

```
asiento6.setEnabled(avion.obtenerEstadoAsientoPC(6));
asiento6.setBounds(0, 0, anchoBoton, altoBoton);
asiento6.setIcon(new
ImageIcon(asientoPCIcon.getImage().getScaledInstance(asiento6.getWidth(),
asiento6.getHeight(), Image.SCALE_SMOOTH)));
add(asiento6);
```

```
JButton espacio1 = new JButton();
espacio1.setEnabled(false);
add(espacio1);
```

```
        JButton espacio2 = new JButton();
        espacio2.setEnabled(false);
        add(espacio2);

        asiento7.setEnabled(avion.obtenerEstadoAsientosCE(7));
        asiento7.setBounds(0, 0, anchoBoton, altoBoton);
        asiento7.setIcon(new
        ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento7.getWidth(),
        asiento7.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento7);

        asiento8.setEnabled(avion.obtenerEstadoAsientosCE(8));
        asiento8.setBounds(0, 0, anchoBoton, altoBoton);
        asiento8.setIcon(new
        ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento8.getWidth(),
        asiento8.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento8);

        asiento9.setEnabled(avion.obtenerEstadoAsientosCE(9));
        asiento9.setBounds(0, 0, anchoBoton, altoBoton);
        asiento9.setIcon(new
        ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento9.getWidth(),
        asiento9.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento9);

        asiento10.setEnabled(avion.obtenerEstadoAsientosCE(10));
        asiento10.setBounds(0, 0, anchoBoton, altoBoton);
        asiento10.setIcon(new
        ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento10.getWidth(),
        asiento10.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento10);

        asiento11.setEnabled(avion.obtenerEstadoAsientosCE(11));
        asiento11.setBounds(0, 0, anchoBoton, altoBoton);
        asiento11.setIcon(new
        ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento11.getWidth(),
        asiento11.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento11);
```

```

        asiento12.setEnabled(avion.obtenerEstadoAsientosCE(12));
        asiento12.setBounds(0, 0, anchoBoton, altoBoton);
        asiento12.setIcon(new
ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento12.getWidth(),
asiento12.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento12);

        asiento13.setEnabled(avion.obtenerEstadoAsientosCE(13));
        asiento13.setBounds(0, 0, anchoBoton, altoBoton);
        asiento13.setIcon(new
ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento13.getWidth(),
asiento13.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento13);

        asiento14.setEnabled(avion.obtenerEstadoAsientosCE(14));
        asiento14.setBounds(0, 0, anchoBoton, altoBoton);
        asiento14.setIcon(new
ImageIcon(asientoCEIcon.getImage().getScaledInstance(asiento14.getWidth(),
asiento14.getHeight(), Image.SCALE_SMOOTH)));
        add(asiento14);

    }

    //_____
    //SETTERS Y GETTERS
    //-----

    public InterfazAvion getVentana() {
        return ventana;
    }

    public void setVentana(InterfazAvion ventana) {
        this.ventana = ventana;
    }

```

```
public JButton getAsiento1() {  
    return asiento1;  
}
```

```
public void setAsiento1(JButton asiento1) {  
    this.asiento1 = asiento1;  
}
```

```
public JButton getAsiento2() {  
    return asiento2;  
}
```

```
public void setAsiento2(JButton asiento2) {  
    this.asiento2 = asiento2;  
}
```

```
public JButton getAsiento3() {  
    return asiento3;  
}
```

```
public void setAsiento3(JButton asiento3) {  
    this.asiento3 = asiento3;  
}
```

```
public JButton getAsiento4() {  
    return asiento4;  
}
```

```
public void setAsiento4(JButton asiento4) {  
    this.asiento4 = asiento4;  
}
```

```
public JButton getAsiento5() {  
    return asiento5;  
}
```

```
public void setAsiento5(JButton asiento5) {  
    this.asiento5 = asiento5;  
}
```

```
public JButton getAsiento6() {  
    return asiento6;  
}
```

```
public void setAsiento6(JButton asiento6) {  
    this.asiento6 = asiento6;  
}
```

```
public JButton getAsiento7() {  
    return asiento7;  
}
```

```
public void setAsiento7(JButton asiento7) {  
    this.asiento7 = asiento7;  
}
```



```
public JButton getAsiento8() {  
    return asiento8;  
}
```

```
public void setAsiento8(JButton asiento8) {  
    this.asiento8 = asiento8;  
}
```

```
public JButton getAsiento9() {  
    return asiento9;  
}
```

```
public void setAsiento9(JButton asiento9) {  
    this.asiento9 = asiento9;  
}
```

```
public JButton getAsiento10() {  
    return asiento10;  
}
```

```
public void setAsiento10(JButton asiento10) {  
    this.asiento10 = asiento10;  
}
```

```
public JButton getAsiento11() {  
    return asiento11;  
}
```

```
public void setAsiento11(JButton asiento11) {
```

```
        this.asiento11 = asiento11;
    }

    public JButton getAsiento12() {
        return asiento12;
    }

    public void setAsiento12(JButton asiento12) {
        this.asiento12 = asiento12;
    }

    public JButton getAsiento13() {
        return asiento13;
    }

    public void setAsiento13(JButton asiento13) {
        this.asiento13 = asiento13;
    }

    public JButton getAsiento14() {
        return asiento14;
    }

    public void setAsiento14(JButton asiento14) {
        this.asiento14 = asiento14;
    }

    /*
    public DialogoAsignacion getDialogo() {
        return dialogo;
    }
    */
```

```

    }

    public void setDialogo(DialogoAsignacion dialogo) {
        this.dialogo = dialogo;
    }

    */

}

```

Clase PanelBotones.

```

package vista;

import java.awt.*;

import javax.swing.*;
import javax.swing.border.*;

/**
 * Panel de botones de interacci3n con el programa del avi3n.
 */
@SuppressWarnings("serial")
public class PanelBotones extends JPanel
{
    // -----
    // Constantes
    // -----

    /**
     * Opci3n registrar.
     */
    public final static String REGISTRAR = "REGISTRAR_PASAJERO";

    /**

```

```

    * Opción anular.
    */
    public final static String ANULAR = "ANULAR_PASAJERO";

    /**
     * Opción buscar.
     */
    public final static String BUSCAR = "BUSCAR_PASAJERO";

    /**
     * Opción porcentaje de ocupación.
     */
    public final static String PORCENTAJE = "PORCENTAJE_OCUPACION";

    /**
     * Opción extensión 1.
     */
    private final static String OPCION_1 = "OPCION_1";

    /**
     * Opción extensión 2.
     */
    private final static String OPCION_2 = "OPCION_2";

    // -----
    // Atributos de interfaz
    // -----

    /**
     * Botón Registro de nuevo pasajero.
     */
    private JButton bRegistro;

    /**
     * Botón Anular.
     */
    private JButton bAnular;

    /**
     * Botón de búsqueda.
     */
    private JButton bBuscarPasajero;

    /**
     * Botón porcentaje de ocupación.

```

```

    */
private JButton bPorcOcupacion;

/**
 * Botón de extensión 1.
 */
private JButton botonOpcion1;

/**
 * Botón de extensión 2.
 */
private JButton botonOpcion2;

/**
 * Interfaz principal.
 */
private InterfazAvion ventana;

// -----
// Constructores
// -----

/**
 * Crea el panel de botones. <br>
 * <b>post: </b> Se crean y alistan los botones de la interfaz.
 * @param pVentana Ventana o frame padre. pVentana != null.
 */
public PanelBotones( )
{

    // Configura propiedades visuales
    setLayout( new GridLayout( 2, 3, 8, 2 ) );
    setBorder( new EmptyBorder( 5, 5, 5, 5 ) );

    // Botón registrar
    bRegistro = new JButton( "Registrar Pasajero" );
    bRegistro.setActionCommand( REGISTRAR );

    bRegistro.setPreferredSize( new Dimension( 40, 10 ) );
    add( bRegistro );

    // Botón anular registro
    bAnular = new JButton( "Eliminar Pasajero" );
    bAnular.setActionCommand( ANULAR );

```

```

        add( bAnular );

        // Botón buscar pasajero
        bBuscarPasajero = new JButton( "Buscar Pasajero" );
        //bBuscarPasajero.setActionCommand( BUSCAR );

        add( bBuscarPasajero );

        // Botón porcentaje de ocupación
        bPorcOcupacion = new JButton( "Porcentaje Ocupación" );
        bPorcOcupacion.setActionCommand( PORCENTAJE );

        add( bPorcOcupacion );

        // Botones de opciones adicionales
        botonOpcion1 = new JButton( "Opción 1" );
        botonOpcion1.setActionCommand( OPCION_1 );

        add( botonOpcion1 );
        botonOpcion2 = new JButton( "Opción 2" );
        botonOpcion2.setActionCommand( OPCION_2 );

        add( botonOpcion2 );
    }

    public static String getRegistrar() {
        return REGISTRAR;
    }

    public static String getAnular() {
        return ANULAR;
    }

    public static String getBuscar() {
        return BUSCAR;
    }

    public static String getPorcentaje() {
        return PORCENTAJE;
    }

    public static String getOpcion1() {
        return OPCION_1;
    }

```

```
}

public static String getOpcion2() {
    return OPCION_2;
}

public JButton getbRegistro() {
    return bRegistro;
}

public void setbRegistro(JButton bRegistro) {
    this.bRegistro = bRegistro;
}

public JButton getbAnular() {
    return bAnular;
}

public void setbAnular(JButton bAnular) {
    this.bAnular = bAnular;
}

public JButton getbBuscarPasajero() {
    return bBuscarPasajero;
}

public void setbBuscarPasajero(JButton bBuscarPasajero) {
    this.bBuscarPasajero = bBuscarPasajero;
}

public JButton getbPorcOcupacion() {
    return bPorcOcupacion;
}

public void setbPorcOcupacion(JButton bPorcOcupacion) {
    this.bPorcOcupacion = bPorcOcupacion;
}

public JButton getBotonOpcion1() {
    return botonOpcion1;
}

public void setBotonOpcion1(JButton botonOpcion1) {
    this.botonOpcion1 = botonOpcion1;
}
}
```

```

    public JButton getBotonOpcion2() {
        return botonOpcion2;
    }

    public void setBotonOpcion2(JButton botonOpcion2) {
        this.botonOpcion2 = botonOpcion2;
    }

    public InterfazAvion getVentana() {
        return ventana;
    }

    public void setVentana(InterfazAvion ventana) {
        this.ventana = ventana;
    }
}

```

Clase DialogoDatosPasajero.

```

package vista;

import java.awt.*;

import javax.swing.*;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;

import modelo.*;
import modelo.Asiento.Clase;
import modelo.Asiento.Ubicacion;

/**
 * Formulario para presentar los datos y el registro del pasajero.
 */
@SuppressWarnings("serial")
public class DialogoDatosPasajero extends JDialog{

    // -----

```



```

// Constantes
// -----

/**
 * Opción Aceptar.
 */
public final static String ACEPTAR = "ACEPTAR";

// -----
// Atributos de Interfaz
// -----

/**
 * Interfaz principal.
 */
private InterfazAvion principal;

/**
 * Texto que contiene la cédula.
 */
private JTextField txtCedula;

/**
 * Texto que contiene el nombre.
 */
private JTextField txtNombre;

/**
 * Texto que contiene la clase.
 */
private JTextField txtClase;

/**
 * Texto que contiene la ubicación.
 */
private JTextField txtUbicacion;

/**
 * Texto que contiene la silla.
 */
private JTextField txtSilla;

// -----
// Constructores
// -----

```

```

/**
 * Crea la ventana con los datos del pasajero.
 * @param pPrincipal Ventana o frame padre del diálogo. pPrincipal !=
null.
 * @param pSilla Silla en la que el pasajero está ubicado.
 */
public DialogoDatosPasajero( InterfazAvion pPrincipal, Asiento pSilla )
{
    principal = pPrincipal;

    setTitle( "Datos del pasajero" );
    setSize( 270, 200 );
    setDefaultCloseOperation( DISPOSE_ON_CLOSE );

    Pasajero pasajero = pSilla.darPasajero( );
    setLayout( new BorderLayout( ) );

    // Crea el panel de datos del pasajero
    JPanel panelDatosPasajero = new JPanel( );

    panelDatosPasajero.setBorder( new CompoundBorder( new TitledBorder(
"Datos del pasajero" ), new EmptyBorder( 3, 3, 3, 3 ) ) );
    panelDatosPasajero.setLayout( new GridLayout( 5, 2, 1, 1 ) );

    // Cédula
    JLabel etiquetaCedula = new JLabel( "Cédula: " );
    panelDatosPasajero.add( etiquetaCedula );

    txtCedula = new JTextField( pasajero.darCedula( ) + "" );
    txtCedula.setEditable( false );
    panelDatosPasajero.add( txtCedula );

    // Nombre
    JLabel etiquetaNombre = new JLabel( "Nombre: " );
    panelDatosPasajero.add( etiquetaNombre );

    txtNombre = new JTextField( pasajero.darNombre( ) );
    txtNombre.setColumns( 15 );
    txtNombre.setEditable( false );
    panelDatosPasajero.add( txtNombre );

    // Silla
    JLabel etiquetaSilla = new JLabel( "Silla: " );
    panelDatosPasajero.add( etiquetaSilla );

```

```

txtSilla = new JTextField( pSilla.darNumero( ) + " " );
txtSilla.setEditable( false );
panelDatosPasajero.add( txtSilla );

// Clase
JLabel etiquetaClase = new JLabel( "Clase: " );

panelDatosPasajero.add( etiquetaClase );
String sClase;
if( pSilla.darClase( ) == Clase.ECOCLASE )
{
    sClase = "Económica";
}
else
{
    sClase = "Ejecutiva";
}
txtClase = new JTextField( sClase );
txtClase.setEditable( false );
panelDatosPasajero.add( txtClase );

// Ubicación
JLabel etiquetaUbicacion = new JLabel( "Ubicación: " );
panelDatosPasajero.add( etiquetaUbicacion );

String sUbicacion;
if( pSilla.darUbicacion( ) == Ubicacion.IZQUIERDA )
{
    sUbicacion = "Derecha";
}
else
{
    sUbicacion = "Izquierda";
}

txtUbicacion = new JTextField( sUbicacion );
txtUbicacion.setEditable( false );
panelDatosPasajero.add( txtUbicacion );
add( panelDatosPasajero, BorderLayout.CENTER );

setModal( true );
setLocationRelativeTo( principal );
setResizable( false );
}

```

```

// -----
// Métodos
// -----
}

```

Clase DialogoAsignaon.

```

package vista;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;
import javax.swing.border.EmptyBorder;

import modelo.*;

/**
 * Formulario para la asignación de sillas.
 */
@SuppressWarnings("serial")
public class DialogoAsignacion extends JDialog implements ActionListener {

    // -----
    // Constantes
    // -----

    /**
     * Opción Aceptar.
     */
    public final static String ACEPTAR = "ACEPTAR";

    /**
     * Opción Cancelar.
     */
}

```

```

public final static String CANCELAR = "CANCELAR";

/**
 * Avion.
 */
private Nave avion;

// -----
// Atributos de interfaz
// -----

/**
 * Interfaz principal.
 */
private InterfazAvion principal;

/**
 * Panel con los botones.
 */
private JPanel panelBotones;

/**
 * Panel con los datos.
 */
private JPanel panelDatos;

/**
 * Boton aceptar.
 */
private JButton botonAceptar;

/**
 * Boton cancelar.
 */
private JButton botonCancelar;

/**
 * Texto de ingreso de cedula.
 */
private JTextField txtCedula;

/**
 * Texto de ingreso de cedula.
 */
private JTextField txtNombre;

```

```

    int numeroSilla ;

    /**
     * Crea el formulario de asignación de pasajeros.
     * @param pPrincipal Ventana o frame padre del diálogo. pPrincipal !=
    null.
     * @param pAvion Avión sobre el que se hará la asignación. pAvion !=
    null.
     */
    public DialogoAsignacion( InterfazAvion pPrincipal, Nave pAvion, int
    numero )
    {

        numeroSilla = numero;

        // Guarda la referencia a la ventana padre
        principal = pPrincipal;

        // Guarda la referencia al avión
        avion = pAvion;

        setTitle( "Registro de pasajero" );
        setSize( 300, 230 );
        setDefaultCloseOperation( JDialog.DISPOSE_ON_CLOSE );

        setLayout( new BorderLayout( ) );

        JPanel panelAux = new JPanel( );
        panelAux.setBorder( new EmptyBorder( 10, 10, 10, 10 ) );
        panelAux.setLayout( new BorderLayout( ) );
        add( panelAux, BorderLayout.CENTER );

        // Crea el panel de datos
        inicializarPanelDatos( );
        panelAux.add( panelDatos, BorderLayout.CENTER );

        // Crea el panel de botones
        inicializarPanelBotones( );
        panelAux.add( panelBotones, BorderLayout.SOUTH );

        setModal( true );
        setLocationRelativeTo( principal );
        setResizable( false );
    }
}

```

```

    }

    public void registrar( int nSilla)
    {

        String nombre;
        String cedula;
        Pasajero pasajero;

        nombre = txtNombre.getText( );
        cedula = txtCedula.getText( );

        if( cedula == null || cedula.equals( "" ) )
        {
            JOptionPane.showMessageDialog( this, "La cedula es requerida",
"Registro", JOptionPane.ERROR_MESSAGE );
        }
        else
        {

            if( nombre == null || nombre.equals( "" ) )
            {
                JOptionPane.showMessageDialog( this, "El nombre es
requerido", "Registro", JOptionPane.ERROR_MESSAGE );
            }
            else
            {

                // Crea al pasajero
                pasajero = new Pasajero( cedula, nombre );

                // Verifica que no este ya el pasajero registrado
                Asiento silla = avion.buscarPasajero( pasajero );

                if( silla != null )
                {
                    JOptionPane.showMessageDialog( this, "El pasajero ya
tiene una silla asignada", "Registro", JOptionPane.ERROR_MESSAGE );
                }
                else
                {

                    avion.asignarSilla( pasajero , nSilla );

                    //principal.actualizar( );
                    dispose( );
                }
            }
        }
    }
}

```

```

    }
}

}

}

}

/**
 * Inicializa el panel con los datos del pasajero.
 */
public void inicializarPanelDatos( )
{
    panelDatos = new JPanel( );
    panelDatos.setLayout( new GridLayout( 4, 1, 1, 6 ) );
    panelDatos.setBorder( BorderFactory.createTitledBorder( "Datos del
pasajero" ) );

    // Cédula
    JPanel panelCedula = new JPanel( );
    panelCedula.setLayout( new FlowLayout( FlowLayout.RIGHT, 5, 0 ) );
    JLabel etiquetaCedula = new JLabel( "Cédula " );
    txtCedula = new JTextField( );
    txtCedula.setColumns( 15 );
    panelCedula.add( etiquetaCedula );
    panelCedula.add( txtCedula );
    panelDatos.add( panelCedula );

    // Nombre
    JPanel panelNombre = new JPanel( );
    panelNombre.setLayout( new FlowLayout( FlowLayout.RIGHT, 5, 0 ) );
    JLabel etiquetaNombre = new JLabel( "Nombre " );
    txtNombre = new JTextField( );
    txtNombre.setColumns( 15 );
    panelNombre.add( etiquetaNombre );
    panelNombre.add( txtNombre );
    panelDatos.add( panelNombre );

}

/**

```



```

    * Inicializa el panel con los botones.
    */
public void inicializarPanelBotones( )
{
    panelBotones = new JPanel( );
    panelBotones.setLayout( new GridLayout( 1, 2, 8, 1 ) );

    // Aceptar
    botonAceptar = new JButton( );
    botonAceptar.setText( "Aceptar" );
    //botonAceptar.setActionCommand( ACEPTAR );
    botonAceptar.addActionListener( this );
    panelBotones.add( botonAceptar );

    // Cancelar
    botonCancelar = new JButton( );
    botonCancelar.setText( "Cancelar" );
    //botonCancelar.setActionCommand( CANCELAR );
    botonCancelar.addActionListener( this );
    panelBotones.add( botonCancelar );
}

//_____
//setters y getters de botones cancelar y aceptar
//-----

public JButton getBotonAceptar() {
    return botonAceptar;
}

public void setBotonAceptar(JButton botonAceptar) {
    this.botonAceptar = botonAceptar;
}

public JButton getBotonCancelar() {
    return botonCancelar;
}

public void setBotonCancelar(JButton botonCancelar) {
    this.botonCancelar = botonCancelar;
}

```

```

//ACTION LISTENER

public void actionPerformed(ActionEvent evento) {
    // TODO Auto-generated method stub

    if(evento.getSource() == botonAceptar){
        registrar(numeroSilla);
        principal.actualizar(avion);
        dispose();
        System.out.println("boton aceptar");
    }else if (evento.getSource() == botonCancelar){
        principal.actualizar(avion);
        dispose();
        System.out.println("boton cancelar");
    }

}

}

```

Clase Banner.

```

package vista;

import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 * Panel que contiene el banner con la imagen.
 */
@SuppressWarnings("serial")
public class Banner extends JPanel {

    // -----

```

```

// Constantes
// -----

/**
 * Ruta de la imagen del banner.
 */
public final static String RUTA_IMAGEN = "./Imagenes/banner.jpg";

// -----
// Atributos de la interfaz
// -----

/**
 * Etiqueta de la imagen.
 */
private JLabel labImagen;

// -----
// Constructores
// -----

/**
 * Construye el panel con el banner.
 */
public Banner( )
{
    labImagen = new JLabel( );
    labImagen.setIcon( new ImageIcon( RUTA_IMAGEN ) );
    add( labImagen );
}
}

```

Controlador:

Clase Controlador.

```

package controlador;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import modelo.*;
import vista.*;

```

```

public class Controlador implements ActionListener{

    private Nave modelo;
    private InterfazAvion vista;
    private DialogoAsignacion dAsignacion;
    private DialogoDatosPasajero datosPasajero;
    private Funcionario funcionario;

    int numeroSilla = 0;

    public Controlador(Nave modelo, InterfazAvion vista) {
        this.modelo = modelo;
        this.vista = vista;
        boolean variable = true;
        boolean seguirCodigo = false;

        while(!seguirCodigo){

            String opc = vista.mostrarMenu();
            int intentos = 0;
            boolean accesoConfirmado = false;

            if(opc == "Ingresar."){
                while( !accesoConfirmado || intentos == 2){
                    while(intentos <2){

                        String nombreFuncionario =
vista.preguntarNombreFuncionario();
                        String contraseñaFuncionario =
vista.preguntarContraseñaFuncionario();

                        funcionario = new Funcionario(nombreFuncionario,
contraseñaFuncionario);
                        boolean aprobado = funcionario.validarDatos();

                        if(aprobado == true){
                            vista.setVisible(true);
                            seguirCodigo = true;

```

```

        accesoConfirmado = true;

    }else if(intentos == 2){
        System.out.println("Demasiados intentos");

    }else{
        vista.contraseñaErrada();
        intentos ++;
    }
}

    }else if (opc == "Registrar nuevo funcionario."){
        String nombreFuncionario =
vista.preguntarNombreFuncionario();
        String contraseñaFuncionario =
vista.preguntarContraseñaFuncionario();

        funcionario = new Funcionario(nombreFuncionario,
contraseñaFuncionario);
        funcionario.escribirRegistro();
        vista.mostrarRegistroExitoso();
    }
}

vista.getPanelBotones().getBotonOpcion1().addActionListener( this);
vista.getPanelBotones().getbPorcOcupacion().addActionListener(this);
vista.getPanelBotones().getbAnular().addActionListener(this);

vista.getPanelBotones().getbBuscarPasajero().addActionListener(this);
vista.getPanelBotones().getbRegistro().addActionListener(this);

while(variable == true){

    vista.getPanelBotones().getBotonOpcion1().addActionListener(
this );
    vista.getPanelBotones().getBotonOpcion2().addActionListener(
this );

    vista.getMapaAsientos().getAsiento1().addActionListener( this );
    vista.getMapaAsientos().getAsiento2().addActionListener(this);
    vista.getMapaAsientos().getAsiento3().addActionListener(this);
    vista.getMapaAsientos().getAsiento4().addActionListener(this);
    vista.getMapaAsientos().getAsiento5().addActionListener(this);
    vista.getMapaAsientos().getAsiento6().addActionListener(this);

```

```

        vista.getMapaAsientos().getAsiento7().addActionListener(this);
        vista.getMapaAsientos().getAsiento8().addActionListener(this);
        vista.getMapaAsientos().getAsiento9().addActionListener(this);
        vista.getMapaAsientos().getAsiento10().addActionListener(this);
        vista.getMapaAsientos().getAsiento11().addActionListener(this);
        vista.getMapaAsientos().getAsiento12().addActionListener(this);
        vista.getMapaAsientos().getAsiento13().addActionListener(this);
        vista.getMapaAsientos().getAsiento14().addActionListener(this);
    }

}

// -----
// Métodos
// -----

/**
 * Acciones de respuesta a los eventos de la interfaz.
 * @param e Evento generado por un elemento de la interfaz.
 */

@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub

    if (e.getSource() == vista.getMapaAsientos().getAsiento1()){

        numeroSilla = 1;
        //registra al pasajero pasando numero de silla para tacharla
en el arreglo
        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento2()){

        numeroSilla = 2;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento3()){

```

```
        numeroSilla = 3;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento4()){

        numeroSilla = 4;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento5()){

        numeroSilla = 5;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento6()){

        numeroSilla = 6;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento7()){

        numeroSilla = 7;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento8()){
```

```
        numeroSilla = 8;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento9()){

        numeroSilla = 9;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento10()){

        numeroSilla = 10;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento11()){

        numeroSilla = 11;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);

    }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento12()){

        numeroSilla = 12;

        registrarPasajero(numeroSilla);

        mostrarDatosP(numeroSilla);
```



```

        }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento13()){

            numeroSilla = 13;

            registrarPasajero(numeroSilla);

            mostrarDatosP(numeroSilla);

        }else if(e.getSource() ==
vista.getMapaAsientos().getAsiento14()){

            numeroSilla = 14;

            registrarPasajero(numeroSilla);

            mostrarDatosP(numeroSilla);

        }else if(e.getSource() ==
vista.getPanelBotones().getbPorcOcupacion()){

            mostrarPorcentaje();

        }else if(e.getSource() == vista.getPanelBotones().getbAnular()){
            eliminarPasajero();
        }else if(e.getSource() ==
vista.getPanelBotones().getbBuscarPasajero()){
            buscarPasajero();
        }else if(e.getSource() ==
vista.getPanelBotones().getbRegistro()){

            numeroSilla = 0;
            boolean variable = false;

            try{
                numeroSilla = vista.preguntarSilla();
            }catch(Exception ex){
                vista.mostrarInputmitmach();
                numeroSilla = vista.preguntarSilla();
            }

            while(!variable){
                if(numeroSilla>0 && numeroSilla<15){
                    variable = true;
                }else{

```

```

        vista.mostrarInputmitmach();
        try{
            numeroSilla = vista.preguntarSilla();
        }catch(Exception ex){
            vista.mostrarInputmitmach();
            numeroSilla = vista.preguntarSilla();
        }

    }

}

registrarPasajero(numeroSilla);

mostrarDatosP(numeroSilla);

}

}

public void registrarPasajero (int numeroSilla){

    dAsignacion = new DialogoAsignacion(vista, modelo, numeroSilla );
    dAsignacion.setVisible(true);
    //vista.actualizar();

}

public void mostrarDatosP (int numeroSilla){

    if(numeroSilla <= 5){

        Asiento[] sillas = modelo.obtenerAsientoPclass();
        datosPasajero = new DialogoDatosPasajero(vista,
sillas[numeroSilla-1]);
        datosPasajero.setVisible(true);
    }else {
        Asiento[] sillas = modelo.obtenerAsientosClaseEco();
        datosPasajero = new DialogoDatosPasajero(vista,
sillas[numeroSilla-7]);
        datosPasajero.setVisible(true);
    }
}

```

```

    }

}

/**
 * Muestra el porcentaje de ocupacion del avión.
 */
public void mostrarPorcentaje () {

    double porcentajeOc = modelo.calcularPorcentajeOcupacion();

    vista.mostrarPorcentaje(porcentajeOc);
}

/**
 * Procesa la anulaci3n del registro de un pasajero.
 */
public void eliminarPasajero( )
{
    // Pregunta el n3mero de c3dula

    String cedula = vista.preguntarCedula();

    if( cedula != null && !cedula.isEmpty( ) )
    {
        Pasajero pasajero = new Pasajero( cedula, "no importa" );

        if( !modelo.desasignarSilla( pasajero ) )
        {
            vista.pasajeroNulo();
        }else{
            vista.pasajeroEliminado();
        }
    }

    vista.actualizar(modelo);
}

/**
 * Procesa la b3squeda de un pasajero.
 */
public void buscarPasajero( )
{
    // Pregunta el n3mero de c3dula
    String cedula = vista.preguntarCedula();
    if( cedula != null && !cedula.isEmpty( ) )
    {

```

```

        Pasajero pasajero = new Pasajero( cedula, "no importa" );

        Asiento silla = modelo.buscarPasajero( pasajero );

        if( silla != null )
        {
            DialogoDatosPasajero vDatos = new DialogoDatosPasajero(
vista, silla );
            vDatos.setVisible( true );
        }
        else
        {
            vista.pasajeroNulo();
        }
    }
}
public void prueba (){
    System.out.println("soy prueba y funciona");
}
}

```

Main:

Clase Main:

```

import vista.*;
import modelo.*;
import controlador.*;

public class Main {
    public static void main(String[] args) {

        InterfazAvion vista;
        Nave modelo;
        Controlador controlador;

        modelo = new Nave();
        vista = new InterfazAvion();
    }
}

```

```
        controlador = new Controlador(modelo,vista);  
    }  
  
}
```