GROUND CONTROL

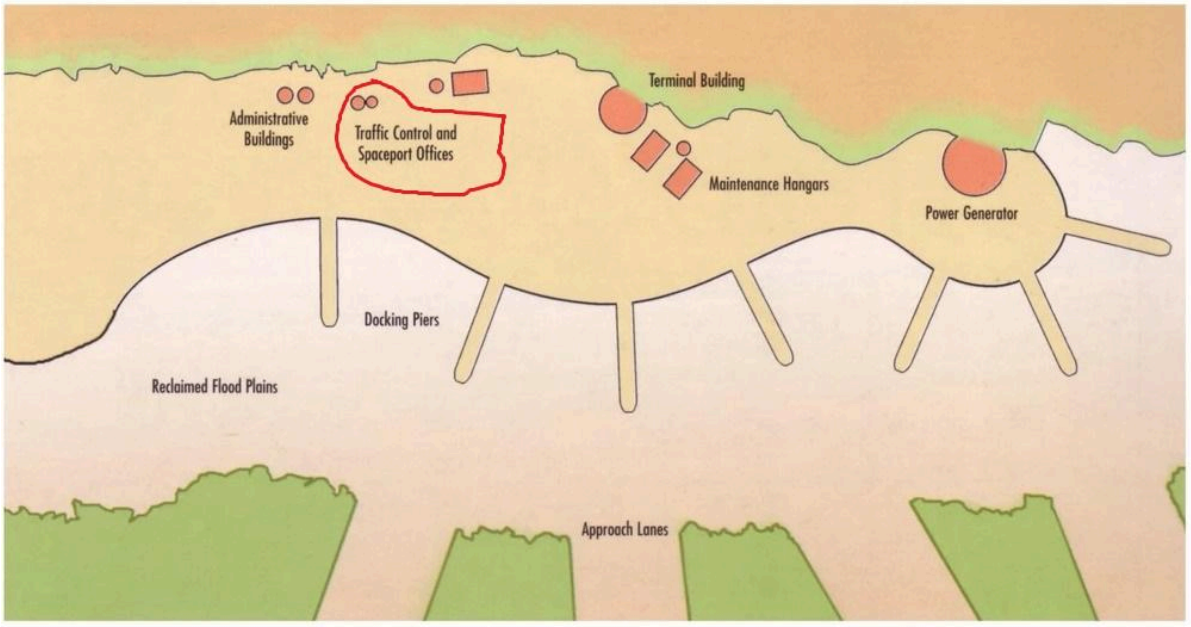
Trabajo realizado por:

- Marta Tirador Gutiérrez
- Víctor García Murillo
- Álvaro Bellón Lanz
- Irene Verdeja Díaz

CONTENIDOS

INTRODUCCIÓN: ORIENTACIÓN ECONÓMICA O DE SERVICIO

El grupo Alan Turing S.L.N.E se dedica al desarrollo de aplicaciones de última generación y pioneras en la innovación de la tecnología de software espacial.
Hemos sido contratados por la República Galáctica en el año 32 BBY para desarrollar una aplicación que servirá al edificio de la terminal de control del nuevo espacio-puerto Theed.



Por la venta del producto de software recibiremos 100,000 créditos galácticos.
Por el mantenimiento de dicho software recibiremos 1000 créditos galácticos mensuales.

ANÁLISIS

Teniendo en cuenta que se prevé que la terminal del espacio-puerto Theed tendrá una concurrencia de naves espaciales tanto militares como de transporte, surge la necesidad de crear clases en herencia para crear objetos de un tipo u otro.

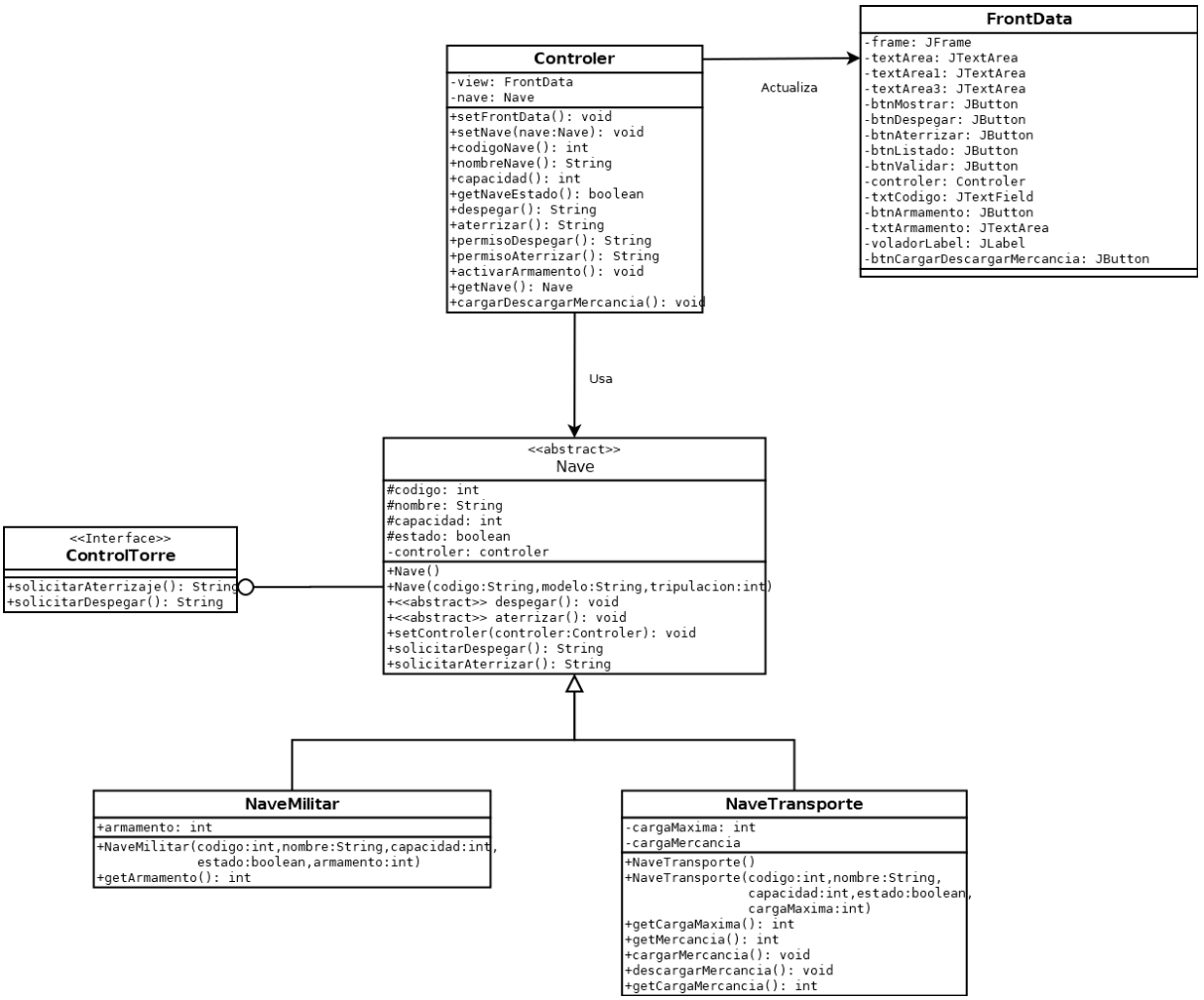
Por otro lado los empleados de la terminal necesitarán una interfaz gráfica de usuario (GUI) donde poder ver, gestionar y controlar el flujo de entradas y salidas de naves del espacio-puerto.

Se requiere una función de escaneo para carga y descarga de mercancía y/o armamento.

Para la GUI se utilizará un patrón de diseño basado en el modelo-vista-controlador.

DISEÑO

UML


Controler
-view: FrontData
-nave: Nave
+setFrontData(): void
+setNave(nave:Nave): void
+codigoNave(): int
+nombreNave(): String
+capacidad(): int
+getNaveEstado(): boolean
+despegar(): String
+aterrizar(): String
+permisoDespegar(): String
+permisoAterrizar(): String
+activarArmamento(): void
+getNave(): Nave
+cargarDescargarMercancia(): void
Actualiza
FrontData
-frame: JFrame
-textArea: JTextArea
-textArea1: JTextArea
-textArea3: JTextArea
-btnMostrar: JButton
-btnDespegar: JButton
-btnAterrizar: JButton
-btnListado: JButton
-btnValidar: JButton
-controler: Controler
-txtCodigo: JTextField
-btnArmamento: JButton
-txtArmamento: JTextArea
-voladorLabel: JLabel
-btnCargarDescargarMercancia: JButton
Usa
<<abstract>>
Nave
#codigo: int
#nombre: String
#capacidad: int
#estado: boolean
-controler: controler
+Nave()
+Nave(codigo:String,modelo:String,tripulacion:int)
+<<abstract>> despegar(): void
+<<abstract>> aterrizar(): void
+setControler(controler:Controler): void
+solicitarDespegar(): String
+solicitarAterrizar(): String
<<Interface>>
ControlTorre
+solicitarAterrizaje(): String
+solicitarDespegar(): String
NaveMilitar
+armamento: int
+NaveMilitar(codigo:int,nombre:String,capacidad:int,
estado:boolean,armamento:int)
+getArmamento(): int
NaveTransporte
-cargaMaxima: int
-cargaMercancia
+NaveTransporte()
+NaveTransporte(codigo:int,nombre:String,
capacidad:int,estado:boolean,
cargaMaxima:int)
+getCargaMaxima(): int
+getMercancia(): int
+cargarMercancia(): void
+descargarMercancia(): void
+getCargaMercancia(): int

AMPLIACIONES

Las ampliaciones propuestas por el Grupo Alan Turing S.L.N.E consisten en una extensión de la aplicación “Ground Control” para que sea posible el control de contrabando y defensa del hangar.

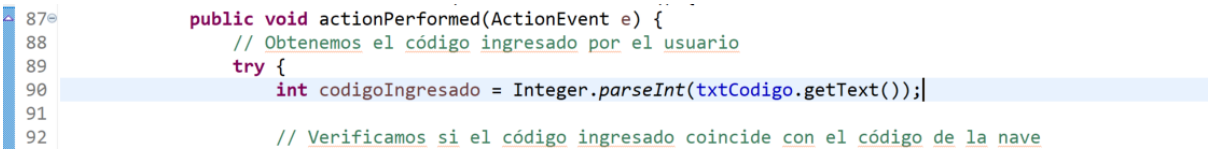
El escaneo se plantea como una barrera a la entrada del hangar que transmitirá a la torre de control la información referente a carga ilegal que pueda transportar la nave.

Para ello, en la GUI se implementaría una opción de escaneo que dirigiría la pantalla a una nueva donde revisar posibles puntos de carga ilegales.

CONCEPTOS TEÓRICO-PRÁCTICOS

CASTING

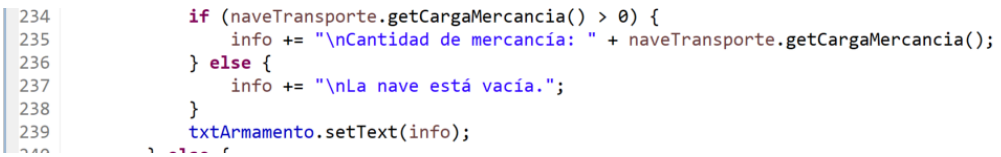
Casting explícito



```
87          public void actionPerformed(ActionEvent e) {
88              // Obtenemos el código ingresado por el usuario
89              try {
90                  int codigoIngresado = Integer.parseInt(txtCodigo.getText());
91
92                  // Verificamos si el código ingresado coincide con el código de la nave
```

CARACTERES DE ESCAPE

Salto de línea



```
234             if (naveTransporte.getCargaMercancia() > 0) {
235                 info += "\nCantidad de mercancía: " + naveTransporte.getCargaMercancia();
236             } else {
237                 info += "\nLa nave está vacía.";
238             }
239             txtArmamento.setText(info);
```

SOBRECARGA DE MÉTODOS

```
10
11    public Nave() {
12          this.codigo=0;
13          this.nombre="";
14          this.capacidad = 0;
15          this.estado=true;
16    }
17    public Nave(int codigo, String nombre,int capacidad, boolean estado) {
18          this.codigo=codigo;
19          this.nombre=nombre;
20          this.capacidad = capacidad;
21          this.estado=estado;
22    }
23
```

REDEFINICIÓN DE MÉTODOS

```
20    @Override
21    public String despegar() {
22        return "La nave militar "+super.nombre+" está despegando.";
23
24    }
25
26    @Override
27    public String aterrizar() {
28        return "La nave militar "+super.nombre+" está aterrizando.";
29    }
30
31    @Override
32    public String solicitarAterrizar() {
33        String mensaje="Nave militar "+super.nombre+ " solicitando permiso para aterrizar...";
34        return mensaje;
35
36    }
37
38    @Override
39    public String solicitarDespegar() {
40       String mensaje="Nave militar "+super.nombre+" solicitando permiso para despegar...";
41        return mensaje;
42    }
43    }
44
```

HERENCIA

```
4  public class NaveMilitar extends Nave{
5      private int armamento;
6
```

HERENCIA Y CONSTRUCTORES (VACÍO Y PARAMETRIZADO)

```
4  public class NaveMilitar extends Nave{
5      private int armamento;
6
7      public NaveMilitar() {
8          super();
9          this.armamento=0;
10     }
11     public NaveMilitar(int codigo, String nombre,int capacidad, boolean estado, int armamento) {
12         super(codigo, nombre,capacidad, estado);
13         this.armamento=armamento;
14     }
15
```

CLASES ABSTRACTAS

```java
public abstract class Nave implements ControlTorre{
    protected int codigo;
    protected String nombre;
    protected int capacidad;
    protected boolean estado;
    private  Controler controler;


    public Nave() {
          this.codigo=0;
          this.nombre="";
          this.capacidad = 0;
          this.estado=true;
    }
    public Nave(int codigo, String nombre,int capacidad, boolean estado) {
          this.codigo=codigo;
          this.nombre=nombre;
          this.capacidad = capacidad;
          this.estado=estado;
    }

    public abstract String despegar();
    public abstract String aterrizar();


```

POLIMORFISMO

```java
        Nave Nave = new NaveTransporte(1112,"Transport Shuttle",690, true,1098);

```

INTERFACES

```java
public interface ControlTorre {
    String solicitarAterrizar();
    String solicitarDespegar();


}

```

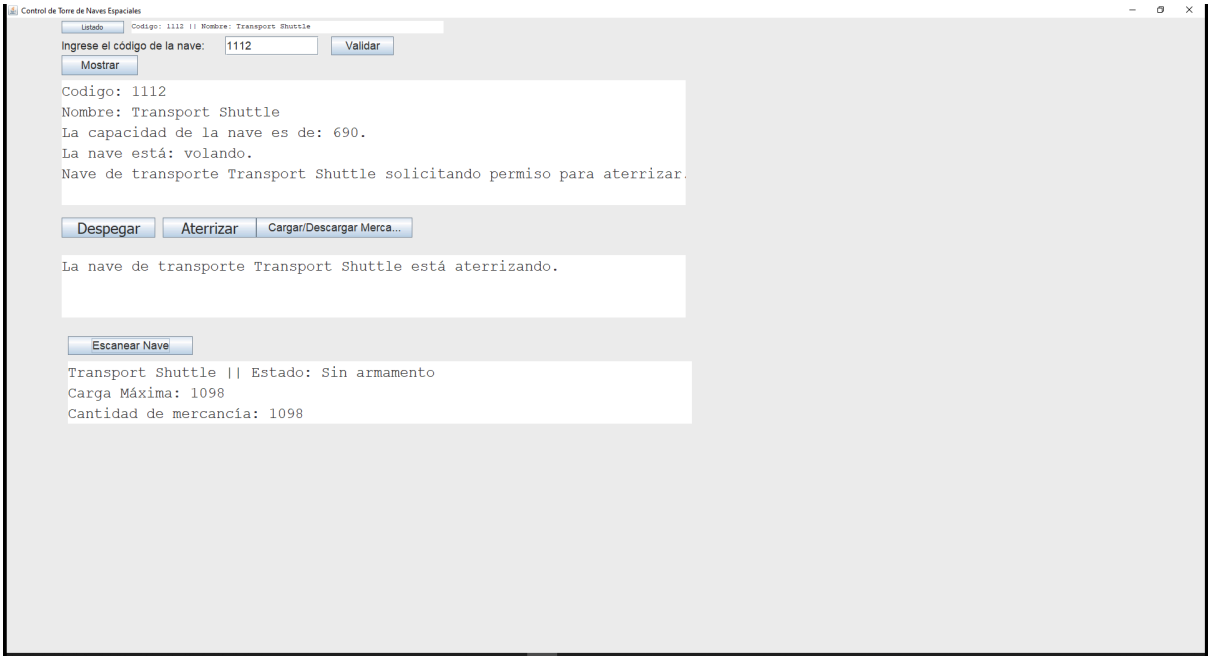
```java
public abstract class Nave implements ControlTorre{
    protected int codigo;
    protected String nombre;
    protected int capacidad;
    protected boolean estado;
    private  Controler controler;

```

EXCEPCIONES

```java
public boolean validarCodigoNave(String codigo) {
    try {
        // Intenta convertir el código a un número
        int codigoNave = Integer.parseInt(codigo);
        // Verifica si el código convertido es positivo
        return codigoNave >= 0;
    } catch (NumberFormatException e) {
        // Si ocurre una NumberFormatException, significa que el código no es un número válido
        return false;
    }
}
```

INTERFACES DE USUARIO



Componentes

```java
private JFrame frame;
private JTextArea textArea;
private JTextArea textArea1;
private JTextArea textArea3;
private JButton btnMostrar;
private JButton btnDespegar;
private JButton btnAterrizar;
private JButton btnListado;
private JButton btnValidar; // Botón de validación
private Controler controler;
private JTextField txtCodigo;
private JButton btnArmamento;
private JTextArea txtArmamento;
private JLabel voladorLabel;

private JButton btnCargarDescargarMercancia;
```

Contenedores

```
private JFrame frame;
```

```
176         JPanel panel = new JPanel();
177         panel.setLayout(null);
178         panel.setBounds(88, 500, 1500, 600);
179         frame.getContentPane().add(panel);
```

Contenedores de eventos

```
194         btnArmamento.addActionListener(new ActionListener() {
195             public void actionPerformed(ActionEvent e) {
196                 activarArmamento();
197             }
198         });
```

ANEXO I

CLASE FLOW

```
public class Flow {

    public static void main(String[] args) {

        FrontData Front = new FrontData();
        Controler Controler = new Controler();
//      Nave Nave = new NaveMilitar(3565,"A Victory I",300, false,2343);
        Nave Nave = new NaveMilitar(4456,"Republic Judiciary CR90 Corvette",400, true, 4556);
        //Nave Nave = new NaveTransporte(2233,"Ortho 2",345, false,9908);
//      Nave Nave = new NaveTransporte(1112,"Transport Shuttle",690, true,1098);

        Controler.SetFrontData(Front);
        Controler.setNave(Nave);

        Front.setControler(Controler);
        Nave.setControler(Controler);
    }

}
```

CLASE NAVE

```java
public abstract class Nave implements ControlTorre{
    protected int codigo;
    protected String nombre;
    protected int capacidad;
    protected boolean estado;
    private  Controler controler;


    public Nave() {
          this.codigo=0;
          this.nombre="";
          this.capacidad = 0;
          this.estado=true;
    }
    public Nave(int codigo, String nombre,int capacidad, boolean estado) {
          this.codigo=codigo;
          this.nombre=nombre;
          this.capacidad = capacidad;
          this.estado=estado;
    }

    public abstract String despegar();
    public abstract String aterrizar();



    public void setControler(Controler controler) {
        this.controler=controler;
    }

    public String solicitarDespegar() {
        return null;
    }
    public String solicitarAterrizar() {
        return null;

    }
}
```

CLASE NAVETRANSPORTE

```java
public class NaveTransporte extends Nave{
    private int cargaMaxima;
    private int cargaMercancia;


    public NaveTransporte() {
         super();
            this.cargaMaxima = 0;
            this.cargaMercancia = 0; // La nave inicia con la carga completa
    }
    public NaveTransporte(int codigo, String nombre, int capacidad, boolean estado, int cargaMaxima) {
        super(codigo, nombre, capacidad, estado);
        this.cargaMaxima = cargaMaxima;
        this.cargaMercancia = cargaMaxima; // La nave inicia con la carga completa
    }

      public int getCargaMaxima() {
            return cargaMaxima;
        }

      public void cargarMercancia() {
            cargaMercancia = cargaMaxima;
        }

        public void descargarMercancia() {
            cargaMercancia = 0;
        }

    @Override
    public String despegar() {
        return "La nave de transporte "+super.nombre+" está despegando.";

    }

    @Override
    public String aterrizar() {
        return "La nave de transporte "+super.nombre+" está aterrizando.";
    }

    @Override
    public String solicitarAterrizar() {
        String mensaje="Nave de transporte "+super.nombre+" solicitando permiso para aterrizar...";
        return mensaje;

    }

    @Override
    public String solicitarDespegar() {
       String mensaje="Nave de transporte "+super.nombre+" solicitando permiso para despegar...";
        return mensaje;
    }

    public int getCargaMercancia() {

        return cargaMercancia;
    }
}
```

CLASE NAVEMILITAR

```
import java.util.Random;

public class NaveMilitar extends Nave{
    private int armamento;

    public NaveMilitar() {
        super();
        this.armamento=0;
    }
    public NaveMilitar(int codigo, String nombre,int capacidad, boolean estado, int armamento) {
        super(codigo, nombre,capacidad, estado);
        this.armamento=armamento;
    }

     public int getArmamento() {
            return armamento;
        }

    @Override
    public String despegar() {
        return "La nave militar "+super.nombre+" está despegando.";

    }

    @Override
    public String aterrizar() {
        return "La nave militar "+super.nombre+" está aterrizando.";
    }

    @Override
    public String solicitarAterrizar() {
        String mensaje="Nave militar "+super.nombre+ " solicitando permiso para aterrizar...";
        return mensaje;

    }

    @Override
    public String solicitarDespegar() {
       String mensaje="Nave militar "+super.nombre+" solicitando permiso para despegar...";
        return mensaje;
    }
    }
```

CLASE CONTROLTORRE

```
public interface ControlTorre {
    String solicitarAterrizar();
    String solicitarDespegar();


}
```

CLASE CONTROLER

```java
import javax.swing.JOptionPane;

public class Controler {
    private FrontData view;
    private Nave nave;

    public void SetFrontData(FrontData view) {
        this.view = view;
    }

    public void setNave(Nave nave) {
        this.nave = nave;
    }

    public int codigoNave() {
        return nave.codigo;
    }

    public String nombreNave() {
        return nave.nombre;
    }

    public int capacidad() {
        return nave.capacidad;
    }

    public boolean getNaveEstado() {
        return nave.estado;
    }

    public String despegar() {
        return nave.despegar();
    }

    public String aterrizar() {
        return nave.aterrizar();
    }

    public String permisoDespegar() {
        return nave.solicitarDespegar();
    }

    public String permisoAterrizar() {
        return nave.solicitarAterrizar();
    }

    private void activarArmamento() {
        if (nave instanceof NaveMilitar) {
            NaveMilitar naveMilitar = (NaveMilitar) nave;
            view.setVoladorInfo("¡Armamento activado! nº serie: " + naveMilitar.getArmamento());
        } else {
            view.setVoladorInfo("Esta nave no tiene armamento");
        }
    }


    public Nave getNave() {
        return nave;
    }

    public void cargarDescargarMercancia() {
        Nave nave = getNave();
        if (nave instanceof NaveTransporte) {
            NaveTransporte naveTransporte = (NaveTransporte) nave;
            int cantidadMercancia = naveTransporte.getCargaMercancia();
            if (cantidadMercancia > 0) {
                // Descargar mercancía
                naveTransporte.descargarMercancia();
            } else {
                // Cargar mercancía
                naveTransporte.cargarMercancia();
            }
            // Actualizar la interfaz con la nueva cantidad de mercancía
            view.actualizarEstadoMercancia(naveTransporte.getCargaMercancia());
        } else {
            JOptionPane.showMessageDialog(null, "Esta operación solo está disponible para naves de transporte.",
                    "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

}
```

CLASE FRONTDATA

```java
import javax.swing.*;
import javax.swing.text.JTextComponent;

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

public class FrontData {

    private JFrame frame;
    private JLabel voladorLabel;
    private JTextArea textArea;
    private JTextArea textArea1;
    private JTextArea textArea3;
    private JButton btnMostrar;
    private JButton btnDespegar;
    private JButton btnAterrizar;
    private JButton btnListado;
    private JButton btnValidar; // Botón de validación
    private Controler controler;
    private JTextField txtCodigo;
    private JButton btnArmamento;
    private JTextArea txtArmamento;


    private JButton btnCargarDescargarMercancia;

    public FrontData() {
        initialize();
    }

    private void initialize() {

        frame = new JFrame();
        frame.setTitle("Control de Torre de Naves Espaciales"); // Configuración del titulo
        frame.setBounds(100, 100, 800, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);
        Font font = new Font("Arial", Font.PLAIN, 18); // Establece la fuente y el tamaño de la fuente

        btnListado = new JButton("Listado");
        btnListado.setFont(new Font("Tahoma", Font.PLAIN, 11));
        btnListado.setEnabled(true);
        btnListado.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int codigo = controler.codigoNave();
                String nombre = controler.nombreNave();
                String mensaje = "Codigo: " + codigo +
                        " || Nombre: " + nombre;
                // Establecer el mensaje en el JTextArea
                textArea3.setText(mensaje);
            }
        });
        btnListado.setBounds(88, 5, 100, 20);
        frame.getContentPane().add(btnListado);

        textArea3 = new JTextArea();
        textArea3.setFont(new Font("Monospaced", Font.PLAIN, 11));
        textArea3.setBounds(200, 5, 500, 20);
        frame.getContentPane().add(textArea3);

        JLabel lblCodigo = new JLabel("Ingrese el código de la nave:");
        lblCodigo.setFont(font); // Aplica la fuente y el tamaño de la fuente
        lblCodigo.setBounds(88, 30, 250, 30);
        frame.getContentPane().add(lblCodigo);

        txtCodigo = new JTextField();
        txtCodigo.setFont(font); // Aplica la fuente y el tamaño de la fuente
        txtCodigo.setBounds(350, 30, 150, 30);
        frame.getContentPane().add(txtCodigo);

        btnValidar = new JButton("Validar"); // Botón de validación
        btnValidar.setFont(font); // Aplica la fuente y el tamaño de la fuente
        btnValidar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                validarCodigo(); // Llama al método para validar el código
            }
        });
        btnValidar.setBounds(520, 30, 100, 30);
        frame.getContentPane().add(btnValidar);
```

```
btnMostrar = new JButton("Mostrar");
btnMostrar.setFont(font); // Aplica la fuente y el tamaño de la fuente
btnMostrar.setEnabled(false); // Deshabilitar el botón inicialmente
btnMostrar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Obtenemos el código ingresado por el usuario
        try {
            int codigoIngresado = Integer.parseInt(txtCodigo.getText());

            // Verificamos si el código ingresado coincide con el código de la nave
            if (codigoIngresado == controler.codigoNave()) {
                // Si coincide, mostramos la información de la nave
                int codigo = controler.codigoNave();
                String nombre = controler.nombreNave();
                int capacidad = controler.capacidad();
                String estado;
                if (controler.getNaveEstado()) {
                    estado = "volando";
                } else {
                    estado = "en tierra";
                }
                String permiso;
                if (estado.equals("volando")) {
                    permiso = controler.permisoAterrizar();
                } else {
                    permiso = controler.permisoDespegar();
                }
                String mensaje = "Codigo: " + codigo +
                        "\nNombre: " + nombre + "\nLa capacidad de la nave es de: " + capacidad +
                        ". \nLa nave está: " + estado + ". \n" + permiso;

                // Establecer el mensaje en el JTextArea
                textAreal.setText(mensaje);

                // Habilitar los botones después de haber ingresado un código correcto
                btnDespegar.setEnabled(true);
                btnAterrizar.setEnabled(true);
                btnCargarDescargarMercancia.setEnabled(true);
                btnArmamento.setEnabled(true);
            } else {
                // Si el código ingresado no coincide, mostramos un mensaje de error
                textAreal.setText("El código ingresado no coincide con el de la nave.");
            }
        } catch (NumberFormatException ex) {
            // Si se produce una excepción al intentar convertir a entero, mostramos un mensaje de error
            textAreal.setText("Por favor, ingrese un valor numérico válido en el campo de código.");
        }
    }
});
btnMostrar.setBounds(88, 60, 122, 32);
frame.getContentPane().add(btnMostrar);

textAreal = new JTextArea();
textAreal.setFont(new Font("Monospaced", Font.PLAIN, 24));
textAreal.setBounds(88, 100, 1000, 200);
frame.getContentPane().add(textAreal);

btnDespegar = new JButton("Despegar");
btnDespegar.setFont(new Font("Tahoma", Font.PLAIN, 24));
btnDespegar.setEnabled(false); // Deshabilitar el botón inicialmente
btnDespegar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (controler.getNaveEstado()) { // Si la nave está volando
            textArea.setText("La nave está volando. Solicita permiso para aterrizar.");
        } else {
            textArea.setText(controler.despegar());
        }
    }
});
btnDespegar.setBounds(88, 320, 150, 32);
frame.getContentPane().add(btnDespegar);

textArea = new JTextArea();
textArea.setFont(new Font("Monospaced", Font.PLAIN, 24));
textArea.setBounds(88, 380, 1000, 100);
frame.getContentPane().add(textArea);
```

```
    btnAterrizar = new JButton("Aterrizar");
    btnAterrizar.setFont(new Font("Tahoma", Font.PLAIN, 24));
    btnAterrizar.setEnabled(false); // Deshabilitar el botón inicialmente
    btnAterrizar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (controler.getNaveEstado()) { // Si la nave está volando
                textArea.setText(controler.aterrizar());
            } else {
                textArea.setText("La nave está en tierra. Solicita permiso para despegar.");
            }
        }
    });
    btnAterrizar.setBounds(250, 320, 150, 32);
    frame.getContentPane().add(btnAterrizar);

    JPanel panel = new JPanel();
    panel.setLayout(null);
    panel.setBounds(88, 500, 1500, 600);
    frame.getContentPane().add(panel);

    btnArmamento = new JButton("Escanear Nave");
    btnArmamento.setFont(font); // Aplica la fuente y el tamaño de la fuente
    btnArmamento.setEnabled(false); // Deshabilitar el botón inicialmente
    btnArmamento.setBounds(10, 10, 200, 30);
    panel.add(btnArmamento);

    txtArmamento = new JTextArea(); // Cambio de JTextField a JTextArea
    txtArmamento.setBackground(Color.WHITE); // Establece el fondo en blanco
    txtArmamento.setFont(new Font("Monospaced", Font.PLAIN, 24)); // Aplica la fuente y el tamaño de la fuente
    txtArmamento.setEditable(false); // Para que no se pueda editar
    txtArmamento.setBounds(10, 50, 1000, 100); // Ajustar los límites del JTextArea
    panel.add(txtArmamento); // Agregar el JTextArea al panel

    btnArmamento.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            activarArmamento();
        }
    });

    // Otros componentes de la interfaz

    btnCargarDescargarMercancia = new JButton("Cargar/Descargar Mercancía");
    btnCargarDescargarMercancia.setFont(font);
    btnCargarDescargarMercancia.setEnabled(false); // Deshabilitar el botón inicialmente
    btnCargarDescargarMercancia.setBounds(400, 320, 250, 32);
    btnCargarDescargarMercancia.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            controler.cargarDescargarMercancia();
        }
    });
    frame.getContentPane().add(btnCargarDescargarMercancia);

    frame.setVisible(true);
}

public void setControler(Controler controler) {
    this.controler = controler;
}

public void setArmamentoOcargaMaxima(String texto) {
    txtArmamento.setText(texto);
}

public void setVoladorInfo(String info) {
    voladorLabel.setText(info);
}
```

```
    private void activarArmamento() {
        if (controler.getNave() instanceof NaveMilitar) {
            NaveMilitar naveMilitar = (NaveMilitar) controler.getNave();
            txtArmamento.setText(naveMilitar.nombre + "\nEstado: ¡Armamento activado!"+"\nArmamento: " + naveMilitar.getArmamento());
        } else if (controler.getNave() instanceof NaveTransporte) {
            NaveTransporte naveTransporte = (NaveTransporte) controler.getNave();
            String info = naveTransporte.nombre + "\nEstado: Sin armamento"+"\nCarga Máxima: " + naveTransporte.getCargaMaxima();
            if (naveTransporte.getCargaMercancia() > 0) {
                info += "\nCantidad de mercancia: " + naveTransporte.getCargaMercancia();
            } else {
                info += "\nLa nave está vacia.";
            }
            txtArmamento.setText(info);
        } else {
            txtArmamento.setText("Esta nave no tiene armamento ni carga máxima.");
        }
    }

    public void actualizarEstadoMercancia(int cantidad) {
        txtArmamento.setText("Cantidad de mercancia: " + cantidad);
    }

    // Método para validar el código
    private void validarCodigo() {
        try {
            int codigoIngresado = Integer.parseInt(txtCodigo.getText());
            if (codigoIngresado == controler.codigoNave()) {
                // Si el código es válido, habilita el botón Mostrar
                btnMostrar.setEnabled(true);
            } else {
                // Si el código no es válido, muestra un mensaje de error
                JOptionPane.showMessageDialog(frame, "Código de nave incorrecto", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            // Si ocurre un error al convertir a entero, muestra un mensaje de error
            JOptionPane.showMessageDialog(frame, "Por favor, ingrese un valor numérico válido en el campo de código", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```