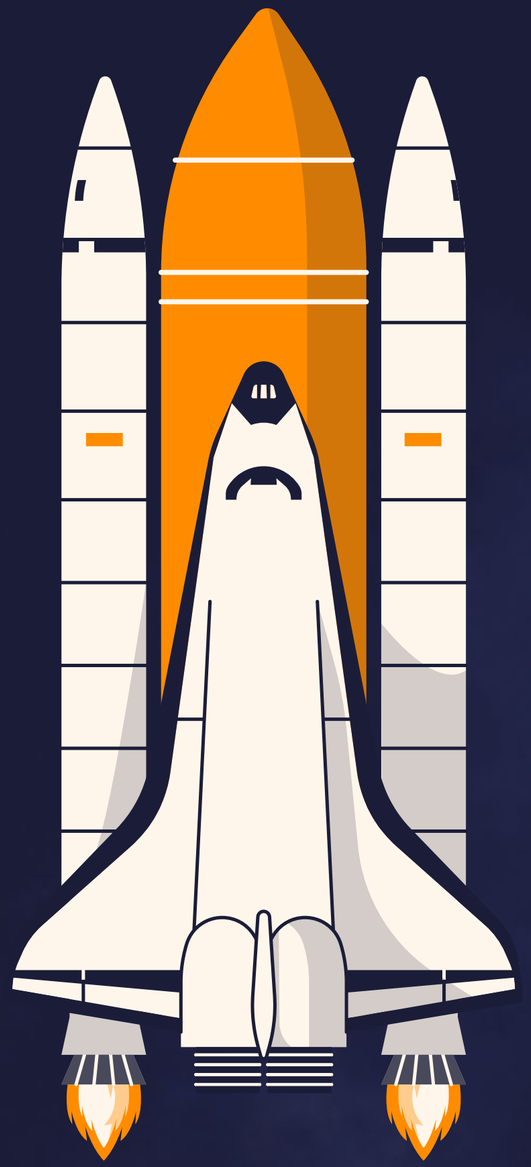


Torre de control

Grupo Alan Turing

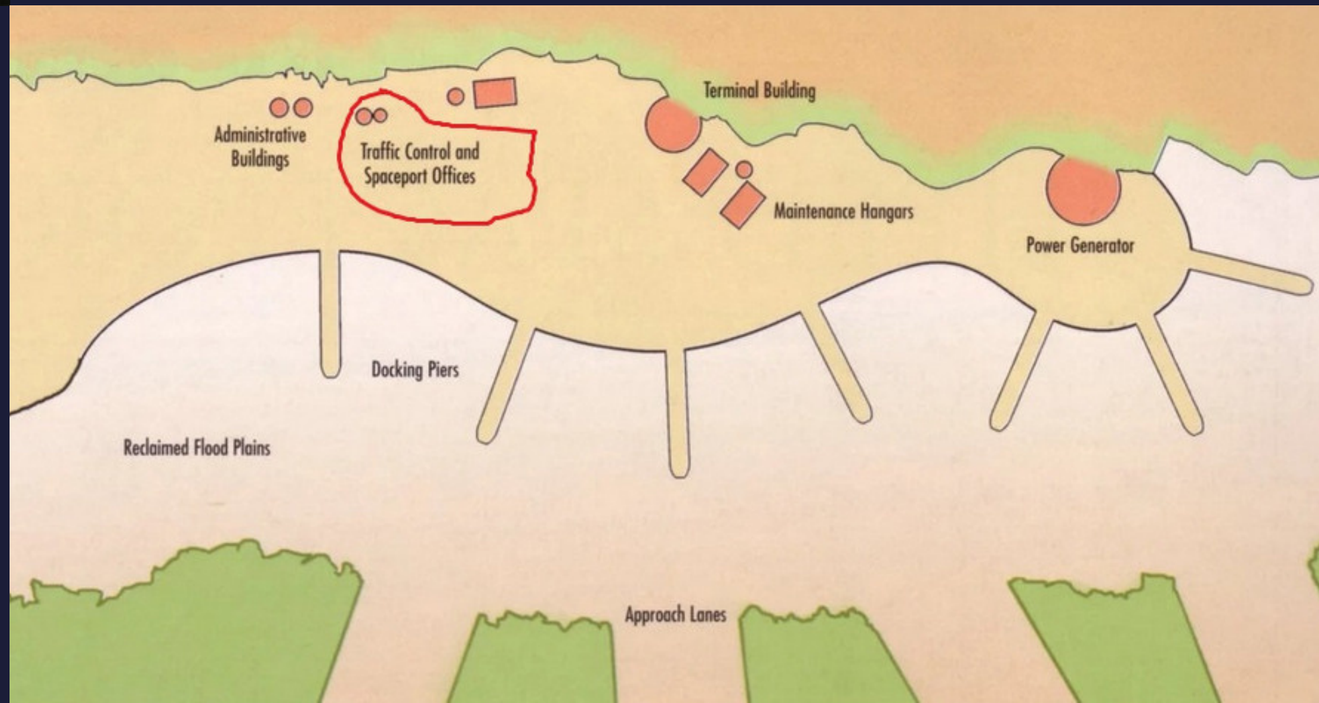
Índice:

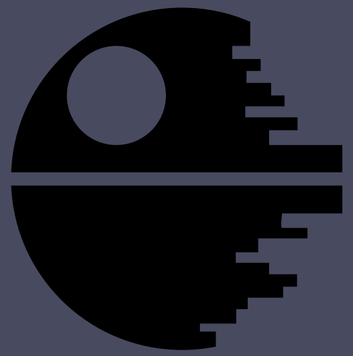
Salir

Empezar

En el planeta Naboo, en el año 32 BBY

Espacio-puerto Theed





Introducción

Empresa: Alan Turing

Cliente: La República Galáctica S.I.N.E.

Producto: 1000000 créditos galácticos

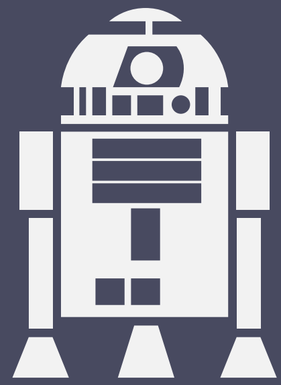
Mantenimiento: 1000 créditos mensuales

Análisis

Ver, gestionar y controlar el flujo de naves naves en el hangar

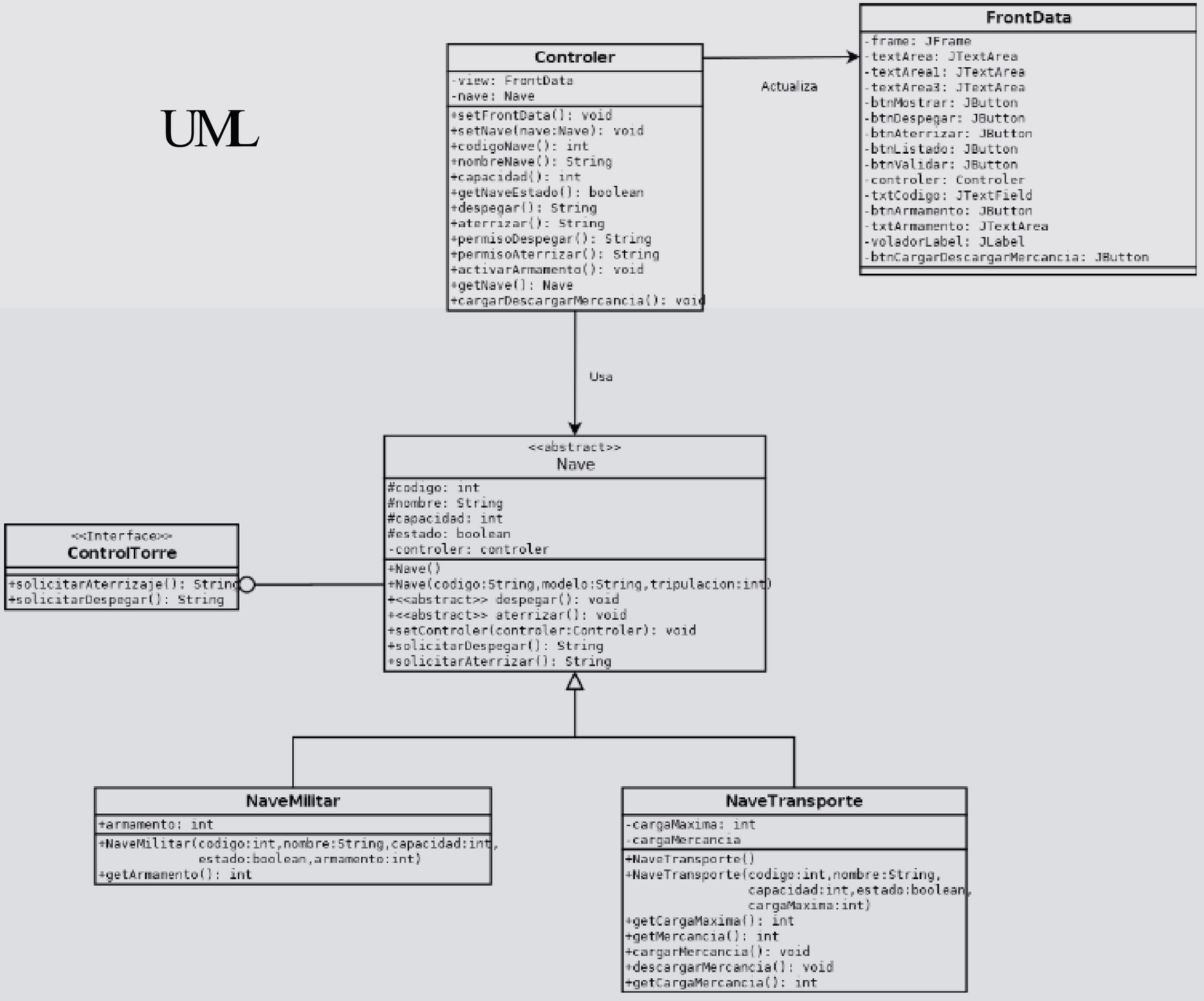
Herencia: tipos de naves

MVC: Modelo Vista
Controlador



Diseño

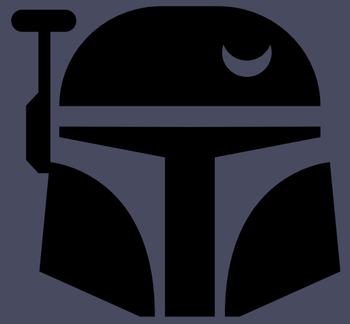
UML


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

El Grupo Alan Turing S.L.N.E propone ampliar la aplicación "Ground Control" para el control de contrabando y la defensa del hangar.

Esta expansión incluirá un sistema de escaneo como barrera en la entrada del hangar, que enviará información sobre cargas ilegales a la torre de control. Se implementará una opción de escaneo en la interfaz gráfica de usuario (GUI), que dirigirá la pantalla a una nueva ventana para revisar posibles puntos de carga ilegal.



Conceptos teórico/práctico

Polimorfismo

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

CASTING

Casting explícito

```
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
```

CARACTERES DE ESCAPE

Salto de línea

```
if (naveTransporte.getCargaMercancia() > 0) {
    info += "\nCantidad de mercancia: " + naveTransporte.getCargaMercancia();
} else {
    info += "\nLa nave está vacía.";
}
txtArmamento.setText(info);
```

SOBRECARGA DE MÉTODOS

```
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;
}
```

REDEFINICIÓN DE MÉTODOS

```
@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;
}
}
```

HERENCIA

```
public class NaveMilitar extends Nave{
    private int armamento;
```

HERENCIA Y CONSTRUCTORES
(VACÍ O Y PARAMETRI ZADO)

```java
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;
    }
```

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();
```

INTERFACES

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


}
```

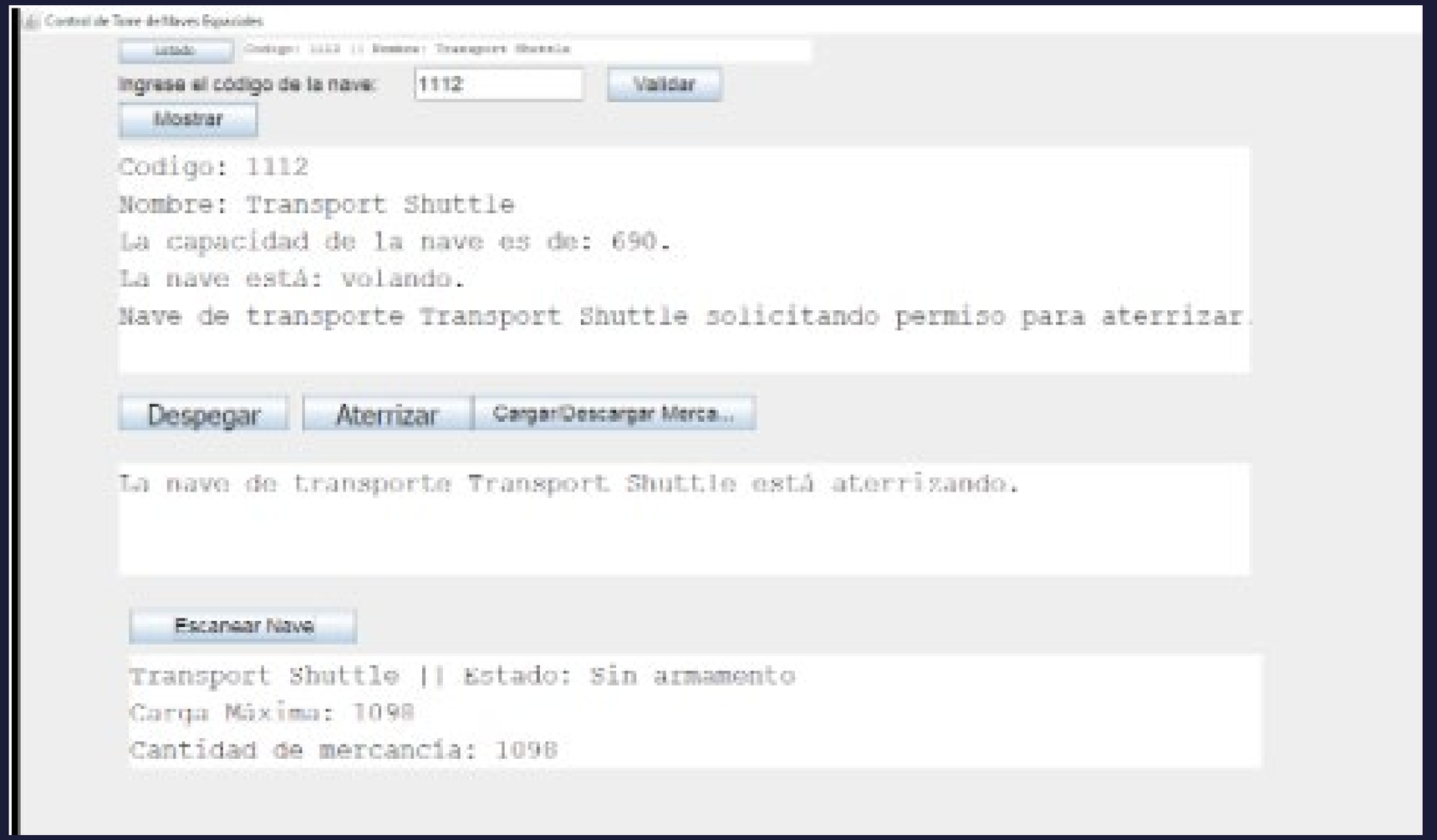
```java
public abstract class Nave implements ControlTorre{
    protected int codigo;
    protected String nombre;
    protected int capacidad;
    protected boolean estado;
    private  Controler controler;

```

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

```
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;

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

Contenedores

```
private JFrame frame;

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

Contenedores de eventos

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

¡Muchas Gracias!

“Este es el camino.”