Instituto Artek
Ingeniería en Inteligencia Artificial
Programación Orientada a Objetos
Profesor: Irving Rocha Reséndiz

Práctica 5 – Herencia

INA0722
Polo Lucy Luis Alejandro

20 de marzo de 2023
Ciudad de México

Introducción

La programación orientada a objetos es una de las formas más populares de programar en la actualidad, y Java es un lenguaje que se destaca en esta área gracias a sus características OO. Uno de los conceptos fundamentales de la programación orientada a objetos es la herencia, que permite que una clase herede características de otra clase. En Java, la herencia se logra mediante la palabra clave "extends".

Otro concepto importante en la programación orientada a objetos es el polimorfismo, que permite que objetos de diferentes clases puedan ser tratados como si fueran del mismo tipo. En Java, el polimorfismo se logra mediante la creación de interfaces y la implementación de métodos con la misma firma en diferentes clases.

La combinación de herencia y polimorfismo permite la creación de una jerarquía de clases bien organizada y modular, lo que facilita la reutilización de código y la creación de aplicaciones más complejas. La herencia permite la creación de clases más específicas a partir de clases más generales, mientras que el polimorfismo permite que se puedan tratar objetos de diferentes clases de manera uniforme.

En resumen, el uso de herencia y polimorfismo en Java es fundamental para la creación de aplicaciones orientadas a objetos eficientes y escalables. Aprender a utilizar estos conceptos es esencial para cualquier estudiante de informática que desee desarrollar habilidades avanzadas en programación orientada a objetos.

Objetivos

- Objetivo 1: Al finalizar esta práctica, los estudiantes serán capaces de identificar y explicar el concepto de herencia en programación orientada a objetos, utilizando correctamente la palabra clave "extends" en Java.

- Objetivo 2: Al finalizar esta práctica, los estudiantes serán capaces de implementar correctamente la herencia en Java, creando clases hijas que heredan atributos y métodos de una clase padre, y sobrescribiendo métodos según sea necesario.

- Objetivo 3: Al finalizar esta práctica, los estudiantes serán capaces de entender y utilizar correctamente el concepto de polimorfismo en Java, creando interfaces y implementando métodos con la misma firma en diferentes clases, lo que les permitirá tratar objetos de diferentes clases de manera uniforme en sus programas.

Desarrollo

1. Desarrolla un sistema para el control de transporte mundial y espacial.
2. Utilizar conceptos de herencia con un mínimo de 3 niveles y máximo 4 niveles de herencia.
3. En el nivel 3 deberá tener mínimo 4 ramas.
4. En el nivel 4 deberá tener mínimo 3 ramas.
5. Entregar el diagrama UML
6. Pruebas de cada uno de los transportes implementados (mínimo 20 vehículos).

Transporte.java (1er Nivel)

```
package Transporte;

public class Transporte {
    protected String medio;

    public Transporte(String medio) {
        this.medio = medio;

    }
    // Setters
    public void setMedio(String medio) {
        this.medio = medio;
    }


    // Getters
    public String getMedio() {
        return this.medio;
    }

}
```

Aereo.java (2do Nivel)

```
package Transporte.Aereo;
import Transporte.Transporte;

public class Aereo extends Transporte {
    private String conduccion;
    public Aereo(String conduccion,String medio){
        super(medio);
        this.conduccion = conduccion;
    }
}
```

Aerostatos.java (3er Nivel)

```
package Transporte.Aereo;
public class Aerostatos extends Aereo {
    private String tipoCarga;
    private String tipoPropulsion;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public Aerostatos(String tipoCarga,String tipoPropulsion, String
alcance, String autonomia, String velocidad, String conduccion, String
medio) {
        super(conduccion,medio);
        this.tipoCarga = tipoCarga;
        this.tipoPropulsion = tipoPropulsion;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetPropulsion(String impulso){
        this.tipoPropulsion = tipoPropulsion;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }
    public String getPropulsion() {
        return tipoPropulsion;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return  "\n    Uso: " + tipoCarga +
                "\n    Tipo de propulsión: "+tipoPropulsion+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

NoPresurizados.java (3er Nivel)

```
package Transporte.Aereo;
public class NoPresurizados extends Aereo {
    private String tipoCarga;
    private String tipoPropulsion;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public NoPresurizados(String tipoCarga,String tipoPropulsion, String
alcance, String autonomia, String velocidad, String conduccion, String
medio) {
        super(conduccion,medio);
        this.tipoCarga = tipoCarga;
        this.tipoPropulsion = tipoPropulsion;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetPropulsion(String impulso){
        this.tipoPropulsion = tipoPropulsion;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }
    public String getPropulsion() {
        return tipoPropulsion;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return  "\n    Uso: " + tipoCarga +
                "\n    Tipo de propulsión: "+tipoPropulsion+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

NoTripulados.java (3er Nivel)

```
package Transporte.Aereo;
public class NoTripulados extends Aereo {
    private String tipoCarga;
    private String tipoPropulsion;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public NoTripulados(String tipoCarga,String tipoPropulsion, String
alcance, String autonomia, String velocidad, String conduccion, String
medio) {
        super(conduccion,medio);
        this.tipoCarga = tipoCarga;
        this.tipoPropulsion = tipoPropulsion;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetPropulsion(String impulso){
        this.tipoPropulsion = tipoPropulsion;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }
    public String getPropulsion() {
        return tipoPropulsion;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return  "\n    Uso: " + tipoCarga +
                "\n    Tipo de propulsión: "+tipoPropulsion+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Presurizados.java (3er Nivel)

```
package Transporte.Aereo;
public class Presurizados extends Aereo {
    private String tipoCarga;
    private String tipoPropulsion;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public Presurizados(String tipoCarga,String tipoPropulsion, String
alcance, String autonomia, String velocidad, String conduccion, String
medio) {
        super(conduccion,medio);
        this.tipoCarga = tipoCarga;
        this.tipoPropulsion = tipoPropulsion;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetPropulsion(String impulso){
        this.tipoPropulsion = tipoPropulsion;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }
    public String getPropulsion() {
        return tipoPropulsion;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return  "\n    Uso: " + tipoCarga +
                "\n    Tipo de propulsión: "+tipoPropulsion+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Maritimo.java (2do Nivel)

```
package Transporte.Maritimo;
import Transporte.Transporte;

public class Maritimo extends Transporte {
    private String longitudCasco;

    public Maritimo(String longitudCasco,String medio) {
        super(medio);
        this.longitudCasco = longitudCasco;
    }
}
```

Barcos.java (3er Nivel)

```
package Transporte.Maritimo;
public class Barcos extends Maritimo {
        private String tipoCarga;
        private String alcance;
        private String impulso;
        private String autonomia;
        private String velocidad;

        public Barcos(String tipoCarga, String alcance,String impulso,
String autonomia, String velocidad, String longitudCasco, String medio) {
            super(longitudCasco,medio);
            this.tipoCarga = tipoCarga;
            this.alcance = alcance;
            this.impulso = impulso;
            this.autonomia = autonomia;
            this.velocidad = velocidad;
        }

        public void SetCarga(String tipoCarga){
            this.tipoCarga = tipoCarga;
        }
        public void SetAlcance(String alcance){this.alcance = alcance;}
        public void SetImpulso(String impulso){
            this.impulso = impulso;
        }
        public void SetAuto(String autonomia){this.autonomia =
autonomia;}
        public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
        public String getTipoCarga() {
            return tipoCarga;
        }

        public String getAlcance() {
            return alcance;
        }
        public String getImpulso() {
            return impulso;
        }

        public String getAuto() {
            return autonomia;
        }
```

```
        public String getVelocidad() {
            return velocidad;
        }

    @Override
    public String toString() {
        return "\n    Tipo de carga: " + tipoCarga +
                "\n    Alcance: "+alcance+
                "\n    Impulso: "+impulso+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Buques.java (3er Nivel)

```
package Transporte.Maritimo;
public class Buques extends Maritimo {
    private String tipoCarga;
    private String alcance;
    private String impulso;
    private String autonomia;
    private String velocidad;

    public Buques(String tipoCarga, String alcance,String impulso, String
autonomia, String velocidad, String longitudCasco, String medio) {
        super(longitudCasco,medio);
        this.tipoCarga = tipoCarga;
        this.alcance = alcance;
        this.impulso = impulso;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetImpulso(String impulso){
        this.impulso = impulso;
    }
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }

    public String getAlcance() {
        return alcance;
    }
    public String getImpulso() {
        return impulso;
    }

    public String getAuto() {
        return autonomia;
    }
```

```
    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Tipo de carga: " + tipoCarga +
                "\n    Alcance: "+alcance+
                "\n    Impulso: "+impulso+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Embarcaciones.java (3er Nivel)

```
package Transporte.Maritimo;
public class Embarcaciones extends Maritimo {
    private String tipoCarga;
    private String alcance;
    private String impulso;
    private String autonomia;
    private String velocidad;

    public Embarcaciones(String tipoCarga, String alcance,String impulso,
String autonomia, String velocidad, String longitudCasco, String medio) {
        super(longitudCasco,medio);
        this.tipoCarga = tipoCarga;
        this.alcance = alcance;
        this.impulso = impulso;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetImpulso(String impulso){
        this.impulso = impulso;
    }
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }

    public String getAlcance() {
        return alcance;
    }
    public String getImpulso() {
        return impulso;
    }

    public String getAuto() {
        return autonomia;
    }
```

```
    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Tipo de carga: " + tipoCarga +
                "\n    Alcance: "+alcance+
                "\n    Impulso: "+impulso+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Submarinos.java (3er Nivel)

```
package Transporte.Maritimo;
public class Submarinos extends Maritimo{
    private String tipoCarga;
    private String alcance;
    private String impulso;
    private String autonomia;
    private String velocidad;

    public Submarinos(String tipoCarga, String alcance,String impulso,
String autonomia, String velocidad, String longitudCasco, String medio) {
        super(longitudCasco,medio);
        this.tipoCarga = tipoCarga;
        this.alcance = alcance;
        this.impulso = impulso;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetImpulso(String impulso){
        this.impulso = impulso;
    }
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() {
        return tipoCarga;
    }

    public String getAlcance() {
        return alcance;
    }
    public String getImpulso() {
        return impulso;
    }

    public String getAuto() {
        return autonomia;
    }
```

```
    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Tipo de carga: " + tipoCarga +
                "\n    Alcance: "+alcance+
                "\n    Impulso: "+impulso+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Terrestre.java (2do Nivel)

```
package Transporte.Terrestre;
import Transporte.Transporte;

public class Terrestre extends Transporte {
    private String regulado;

    public Terrestre(String clasificacion, String regulado) {
        super(clasificacion);
        this.regulado = regulado;
    }
}
```

Electricos.java (3er Nivel)

```
ackage Transporte.Terrestre;
public class Electricos extends Terrestre {
    private String tipoCarga;
    private String NumEjes;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public Electricos(String tipoCarga,String NumEjes, String alcance,
String autonomia, String velocidad, String regulado, String medio) {
        super(regulado,medio);
        this.tipoCarga = tipoCarga;
        this.NumEjes = NumEjes;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetNumEjesing(){this.NumEjes = NumEjes;}
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() { return tipoCarga;
```

```
    }
    public String getNumEjes(){ return NumEjes;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Modelo: " + tipoCarga +
                "\n    Número de Ejes: "+NumEjes+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Ferroviario.java (3er Nivel)

```
package Transporte.Terrestre;
public class Ferroviario extends Terrestre {
    private String tipoCarga;
    private String NumEjes;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public Ferroviario(String tipoCarga,String NumEjes, String alcance,
String autonomia, String velocidad, String regulado, String medio) {
        super(regulado,medio);
        this.tipoCarga = tipoCarga;
        this.NumEjes = NumEjes;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetNumEjesing(){this.NumEjes = NumEjes;}
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() { return tipoCarga;
    }
    public String getNumEjes(){ return NumEjes;
    }
    public String getAlcance() {
        return alcance;
    }
```

```
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Modelo: " + tipoCarga +
                "\n    Número de Ejes: "+NumEjes+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Motorizados.java (3er Nivel)

```
package Transporte.Terrestre;
public class Motorizados extends Terrestre {
    private String tipoCarga;
    private String NumEjes;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public Motorizados(String tipoCarga,String NumEjes, String alcance,
String autonomia, String velocidad, String regulado, String medio) {
        super(regulado,medio);
        this.tipoCarga = tipoCarga;
        this.NumEjes = NumEjes;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetNumEjesing(){this.NumEjes = NumEjes;}
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() { return tipoCarga;
    }
    public String getNumEjes(){ return NumEjes;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
```

```
    }

    @Override
    public String toString() {
        return "\n    Tipo: " + tipoCarga +
                "\n    Número de Ejes: "+NumEjes+
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

NoMotorizados.java (3er Nivel)

```
package Transporte.Terrestre;
public class NoMotorizados extends Terrestre {
    private String tipoCarga;
    private String NumEjes;
    private String alcance;
    private String autonomia;
    private String velocidad;

    public NoMotorizados(String tipoCarga,String NumEjes, String alcance,
String autonomia, String velocidad, String regulado, String medio) {
        super(regulado,medio);
        this.tipoCarga = tipoCarga;
        this.NumEjes = NumEjes;
        this.alcance = alcance;
        this.autonomia = autonomia;
        this.velocidad = velocidad;
    }

    public void SetCarga(String tipoCarga){
        this.tipoCarga = tipoCarga;
    }
    public void SetNumEjesing(){this.NumEjes = NumEjes;}
    public void SetAlcance(String alcance){this.alcance = alcance;}
    public void SetAuto(String autonomia){this.autonomia = autonomia;}
    public void SetVelocidad(String velocidad){this.velocidad =
velocidad;}
    public String getTipoCarga() { return tipoCarga;
    }
    public String getNumEjes(){ return NumEjes;
    }
    public String getAlcance() {
        return alcance;
    }
    public String getAuto() {
        return autonomia;
    }

    public String getVelocidad() {
        return velocidad;
    }

    @Override
    public String toString() {
        return "\n    Tipo: " + tipoCarga +
                "\n    Número de Ejes: "+NumEjes+
```

```
                "\n    Alcance: "+alcance+
                "\n    Autonomía: "+autonomia+
                "\n    Velocidad: "+velocidad;
    }
}
```

Main.java

```
import Transporte.Aereo.Aerostatos;
import Transporte.Aereo.NoPresurizados;
import Transporte.Aereo.NoTripulados;
import Transporte.Aereo.Presurizados;
import Transporte.Maritimo.Barcos;
import Transporte.Maritimo.Buques;
import Transporte.Maritimo.Embarcaciones;
import Transporte.Maritimo.Submarinos;
import Transporte.Terrestre.*;

import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        int count = 1;
        //Transporte Aereo
        List<Aerostatos> aerostatos = new ArrayList<>();
        List<NoPresurizados> noPresurizados = new ArrayList<>();
        List<Presurizados> presurizados = new ArrayList<>();
        List<NoTripulados> noTripulados = new ArrayList<>();
        //Transporte Marítimo
        List<Buques> buques = new ArrayList<>();
        List<Embarcaciones> embarcaciones = new ArrayList<>();
        List<Barcos> barcos = new ArrayList<>();
        List<Submarinos> submarinos = new ArrayList<>();
        //Transporte Terrestre
        List<Electricos> electricos = new ArrayList<>();
        List<Ferroviario> ferroviarios = new ArrayList<>();
        List<NoMotorizados> noMotorizados = new ArrayList<>();
        List<Motorizados> motorizados = new ArrayList<>();

    //Transporte Aereo
        aerostatos.add(new Aerostatos("Helio", "Civil", "1500 km", "4
horas", "90 km/h", "Piloto", "Aire"));
        aerostatos.add(new Aerostatos("Gas", "Militar", "3000 km", "8
horas", "120 km/h", "Piloto", "Aire"));
        aerostatos.add(new Aerostatos("Helio", "Turístico", "500 km", "2
horas", "80 km/h", "Piloto", "Aire"));

        noPresurizados.add(new NoPresurizados("Pasajeros", "Turbina",
"5000 km", "10 horas", "800 km/h", "Piloto", "Aire"));
        noPresurizados.add(new NoPresurizados("Carga", "Turbina", "3000
km", "8 horas", "700 km/h", "Piloto", "Aire"));
        noPresurizados.add(new NoPresurizados("Pasajeros", "Hélice",
"2000 km", "4 horas", "500 km/h", "Piloto", "Aire"));

        presurizados.add(new Presurizados("Pasajeros", "Turbina", "10000
km", "12 horas", "900 km/h", "Piloto", "Aire"));
        presurizados.add(new Presurizados("Carga", "Turbina", "8000 km",
"10 horas", "800 km/h", "Piloto", "Aire"));
        presurizados.add(new Presurizados("Carga", "Hélice", "5000 km",
```

```
"6 horas", "600 km/h", "Piloto", "Aire"));

        noTripulados.add(new NoTripulados("Carga", "Eléctrico", "100 km",
"1 hora", "50 km/h", "Autónomo", "Aire"));
        noTripulados.add(new NoTripulados("Reconocimiento", "Gasolina",
"500 km", "4 horas", "100 km/h", "Autónomo", "Aire"));
        noTripulados.add(new NoTripulados("Exploración", "Eléctrico", "50
km", "1 hora", "30 km/h", "Autónomo", "Aire"));

    //Transporte Marítimo
        buques.add(new Buques("Carga", "Diesel", "15000 km", "30 días",
"50 km/h", "Capitán", "Mar"));
        buques.add(new Buques("Pasajeros", "Diesel", "10000 km", "20
días", "60 km/h", "Capitán", "Mar"));
        buques.add(new Buques("Carga", "Gasolina", "8000 km", "15 días",
"40 km/h", "Capitán", "Mar"));

        embarcaciones.add(new Embarcaciones("Pasajeros", "Diesel", "500
km", "1 día", "30 km/h", "Capitán", "Mar"));
        embarcaciones.add(new Embarcaciones("Pesca", "Gasolina", "200
km", "8 horas", "20 km/h", "Capitán", "Mar"));
        embarcaciones.add(new Embarcaciones("Turístico", "Gasolina", "100
km", "4 horas", "25 km/h", "Capitán", "Mar"));

        barcos.add(new Barcos("Carga seca", "500 km", "Diesel", "1 día",
"30 km/h", "50 metros", "Mar"));
        barcos.add(new Barcos("Productos químicos", "800 km", "Gasolina",
"2 días", "25 km/h", "60 metros", "Mar"));
        barcos.add(new Barcos("Pasajeros", "200 km", "Diesel", "8 horas",
"35 km/h", "40 metros", "Mar"));

        submarinos.add(new Submarinos("Armamento", "1000 km", "Diesel",
"20 días", "25 nudos", "70 m", "Agua"));
        submarinos.add(new Submarinos("Suministros", "2000 km",
"Eléctrico", "30 días", "20 nudos", "80 m", "Agua"));
        submarinos.add(new Submarinos("Investigación", "500 km",
"Nuclear", "60 días", "30 nudos", "90 m", "Agua"));

        //Transporte Terrestre
        electricos.add(new Electricos("Tesla Model S", "Eléctrico", "350
km", "3.2 s", "250 km/h", "Carretera", "Eléctrico"));
        electricos.add(new Electricos("Nissan Leaf", "Eléctrico", "240
km", "8.4 s", "144 km/h", "Ciudad", "Eléctrico"));
        electricos.add(new Electricos("BMW i3", "Eléctrico", "260 km",
"7.3 s", "150 km/h", "Ciudad", "Eléctrico"));

        ferroviarios.add(new Ferroviario("Tren AVE", "Eléctrico", "700
km", "4 h", "320 km/h", "Alta velocidad", "Tren"));
        ferroviarios.add(new Ferroviario("Tren de Cercanías",
"Eléctrico", "40 km", "1 h", "80 km/h", "Cercanías", "Tren"));
        ferroviarios.add(new Ferroviario("Tren de Carga", "Diésel", "Sin
límite", "Sin límite", "60 km/h", "Carga", "Tren"));

        noMotorizados.add(new NoMotorizados("Bicicleta", "Muscular", "Sin
límite", "Sin límite", "40 km/h", "Ciudad", "Bicicleta"));
        noMotorizados.add(new NoMotorizados("Patineta", "Muscular", "Sin
límite", "Sin límite", "30 km/h", "Ciudad", "Patineta"));
        noMotorizados.add(new NoMotorizados("Patineta eléctrica",
"Eléctrico", "30 km", "2 h", "35 km/h", "Ciudad", "Patineta"));

        motorizados.add(new Motorizados("Automóvil", "Gasolina", "900
```

```
km", "5 s", "350 km/h", "Carretera", "Automóvil"));
        motorizados.add(new Motorizados("Camión", "Diésel", "1200 km",
"15 s", "120 km/h", "Carretera", "Camión"));
        motorizados.add(new Motorizados("Moto", "Gasolina", "400 km",
"2.5 s", "300 km/h", "Carretera", "Moto"));

        //Transporte Terrestre
        System.out.println("\nTransporte Terrestre");
        System.out.println("    Eléctricos:");
        for (Electricos electrico : electricos) {
            System.out.println("- " + electrico.toString());
        }

        System.out.println("\n    Ferroviario:");
        for (Ferroviario ferroviario : ferroviarios) {
            System.out.println("- " + ferroviario.toString());
        }

        System.out.println("\n    No Motorizado:");
        for (NoMotorizados noMotorizado : noMotorizados) {
            System.out.println("- " + noMotorizado.toString());
        }

        System.out.println("\n    Motorizados:");
        for (Motorizados motorizado : motorizados) {
            System.out.println("- " + motorizado.toString());
        }

        //Transporte Maritimo
        System.out.println("\nTransporte Marítimo");
        System.out.println("\n    Buques:");
        for (Buques buque : buques) {
            System.out.println("- " + buque.toString());
        }

        System.out.println("\n    Embarcaciones:");
        for (Embarcaciones embarcacion : embarcaciones) {
            System.out.println("- " + embarcacion.toString());
        }

        System.out.println("\n    Barcos:");
        for (Barcos barco : barcos) {
            System.out.println("- " + barco.toString());
        }

        System.out.println("\n    Submarinos:");
        for (Submarinos submarino : submarinos) {
            System.out.println("- " + submarino.toString());
        }

        //Transporte Aereo
        System.out.println("\nTransporte Aereo");
        System.out.println("    Aerostatos:");
        for (Aerostatos aerostato : aerostatos) {
            System.out.println("- " + aerostato.toString());
        }

        System.out.println("\n    No presurizados:");
        for (NoPresurizados noPresurizado : noPresurizados) {
            System.out.println("- " + noPresurizado.toString());
        }
```

```
        System.out.println("\n    Presurizados:");
        for (Presurizados presurizado : presurizados) {
            System.out.println("- " + presurizado.toString());
        }

        System.out.println("\n   No Tripulado:");
        for (NoTripulados noTripulado : noTripulados) {
            System.out.println("- " + noTripulado.toString());
        }
    }
}
```

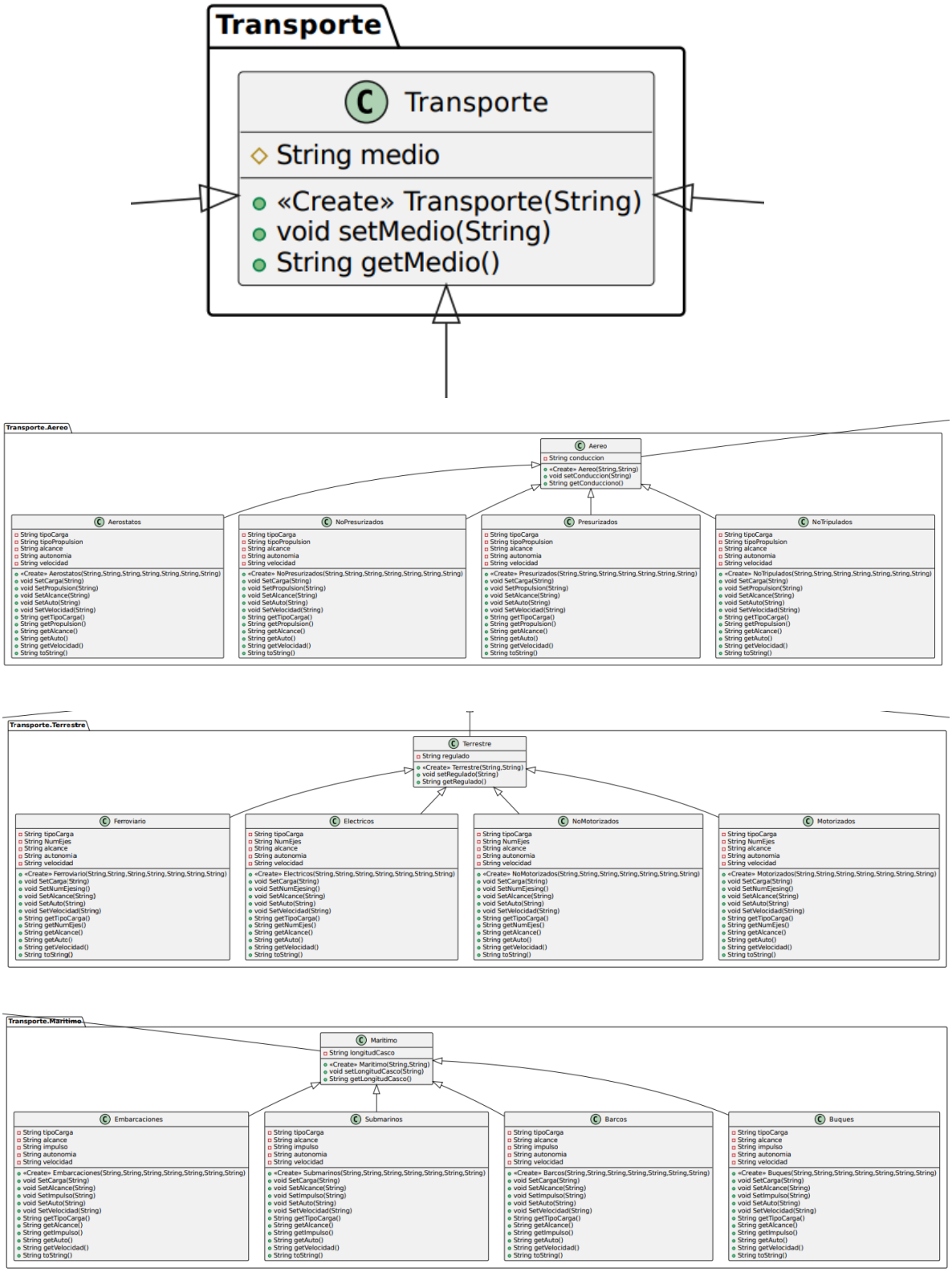
Pruebas

(Adjunto un pdf con los resultados impresos y completos)

File - Main

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\lib\idea_rt.jar=58445:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\LAPL\IdeaProjects\Practica_5\out\production\Practica_5 Main

Transporte Terrestre
    Eléctricos:
-
    Modelo: Tesla Model S
    Número de Ejes: Eléctrico
    Alcance: 350 km
    Autonomía: 3.2 s
    Velocidad: 250 km/h
-
    Modelo: Nissan Leaf
    Número de Ejes: Eléctrico
    Alcance: 240 km
    Autonomía: 8.4 s
    Velocidad: 144 km/h
-
    Modelo: BMW i3
    Número de Ejes: Eléctrico
    Alcance: 260 km
    Autonomía: 7.3 s
    Velocidad: 150 km/h

    Ferroviario:
-
    Modelo: Tren AVE
    Número de Ejes: Eléctrico
    Alcance: 700 km
    Autonomía: 4 h
    Velocidad: 320 km/h
-
    Modelo: Tren de Cercanías
    Número de Ejes: Eléctrico
    Alcance: 40 km
    Autonomía: 1 h
    Velocidad: 80 km/h
-
    Modelo: Tren de Carga
    Número de Ejes: Diésel
    Alcance: Sin límite
    Autonomía: Sin límite
    Velocidad: 60 km/h

    No Motorizado:
-
    Tipo: Bicicleta
    Número de Ejes: Muscular
    Alcance: Sin límite
    Autonomía: Sin límite
    Velocidad: 40 km/h
-
    Tipo: Patineta
    Número de Ejes: Muscular
    Alcance: Sin límite
    Autonomía: Sin límite
    Velocidad: 30 km/h
-
    Tipo: Patineta eléctrica
    Número de Ejes: Eléctrico
    Alcance: 30 km
    Autonomía: 2 h
    Velocidad: 35 km/h

    Motorizados:
-
    Tipo: Automóvil
    Número de Ejes: Gasolina
    Alcance: 900 km
```

Diagrama UML (Adjunto la imagen en mi entrega, no alcanza a verse)



Conclusiones

En conclusión, el uso de herencia y polimorfismo en Java es esencial para la creación de aplicaciones orientadas a objetos eficientes y escalables. La herencia permite crear clases más específicas a partir de clases más generales, lo que facilita la reutilización de código y la creación de aplicaciones más complejas. Por su parte, el polimorfismo permite tratar objetos de diferentes clases de manera uniforme, lo que resulta especialmente útil cuando se trabaja con grandes cantidades de datos.

La implementación correcta de la herencia y el polimorfismo requiere de un conocimiento sólido de los conceptos fundamentales de la programación orientada a objetos y de las características específicas de Java. Para ello, es importante que los estudiantes practiquen la creación de clases hijas, la sobrescritura de métodos, la implementación de interfaces y la creación de objetos polimórficos.

En definitiva, esta práctica puede resultar de gran utilidad para que los estudiantes adquieran habilidades avanzadas en programación orientada a objetos en Java, lo que les permitirá desarrollar aplicaciones más eficientes, escalables y fáciles de mantener. Con el conocimiento adquirido, los estudiantes estarán mejor preparados para enfrentar los desafíos de la programación moderna y podrán destacar en un mercado cada vez más competitivo.

Bibliografía

Oracle. (2021). Inheritance. Java Tutorials for Beginners and Professionals. https://www.javatpoint.com/inheritance-in-java

Baeldung. (2021). Polymorphism in Java. Baeldung. https://www.baeldung.com/java-polymorphism

GeeksforGeeks. (2021). Inheritance in Java. GeeksforGeeks. https://www.geeksforgeeks.org/inheritance-in-java/