Universidad Autónoma de Nuevo León.
Facultad de Ciencias Físico Matemáticas.

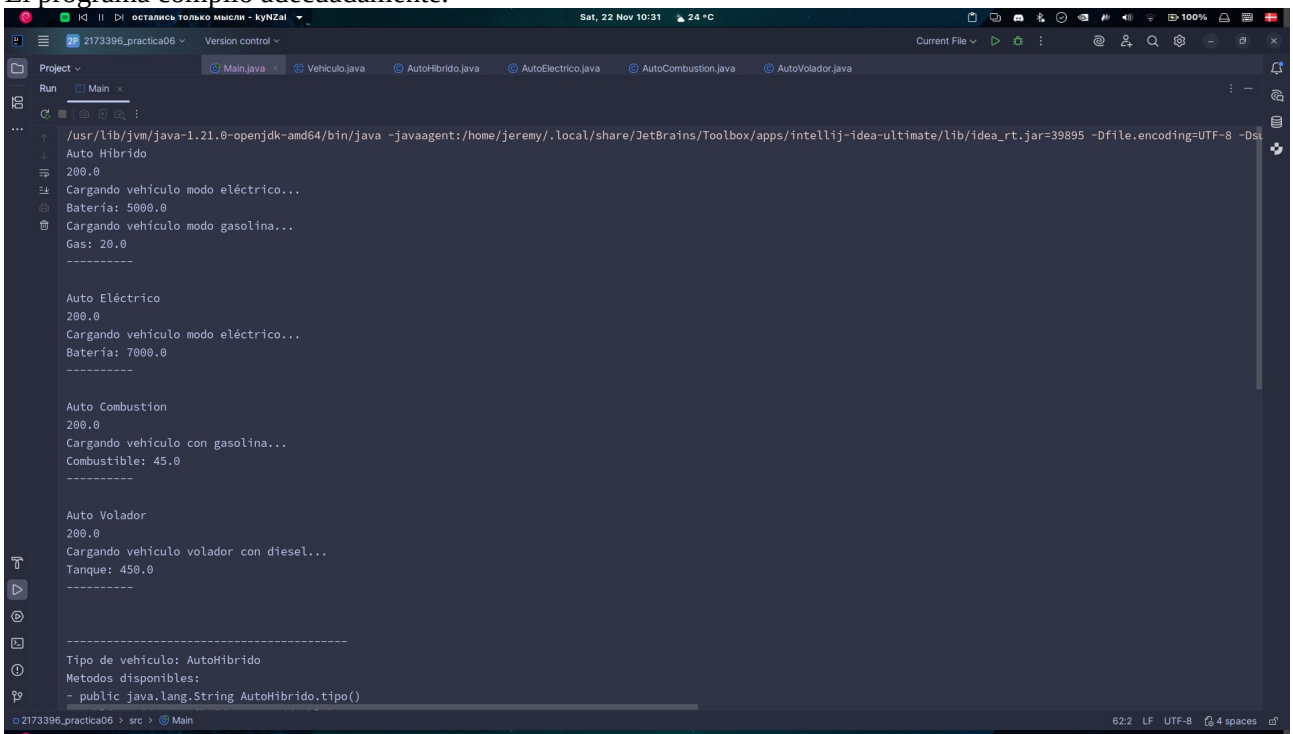
Práctica #06 – Polimorfismo.

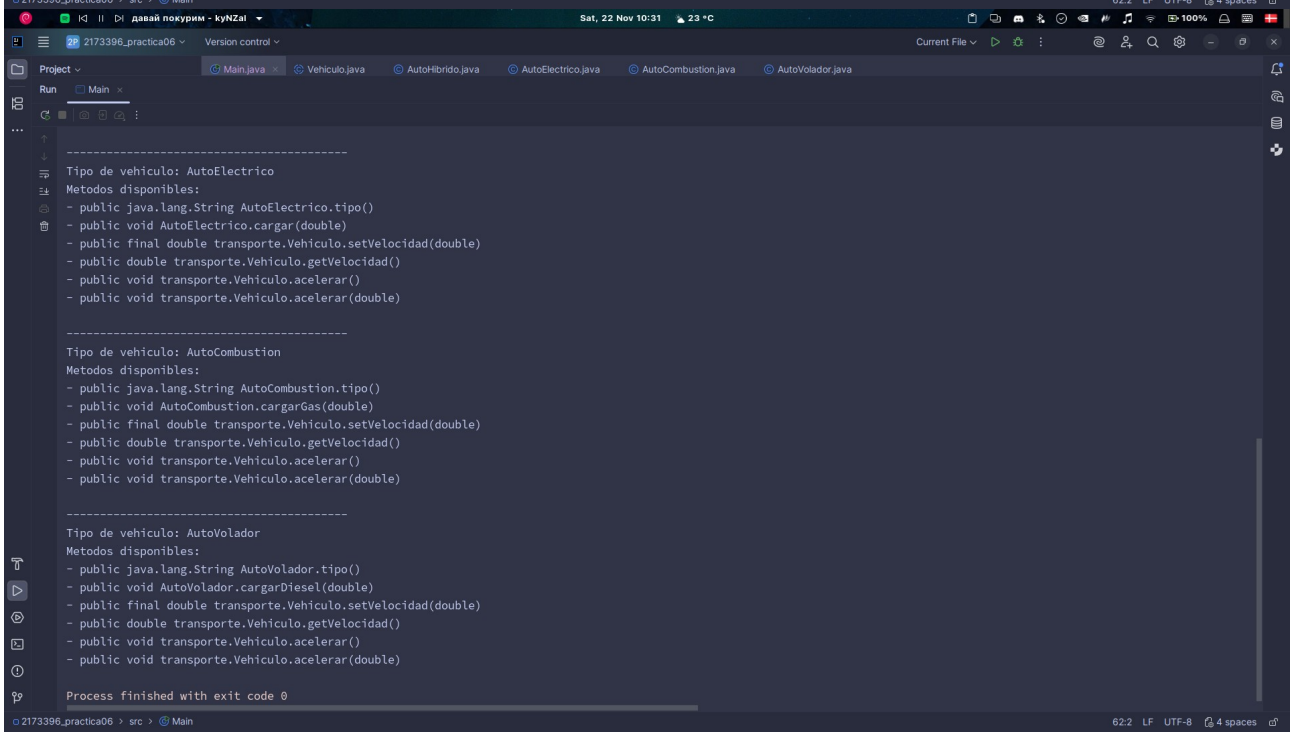
Laboratorio de Programación Orientada a Objetos.

Alumno: Jeremy Uriel Rossell Segura.
Matrícula: 2173396.
Grupo: 036.
Horario: VIERNES 7:00-9:00.
Materia: Laboratorio de Programación Orientada a objetos.
Docente: Jorge Alberto Islas Pineda.

Compilación.

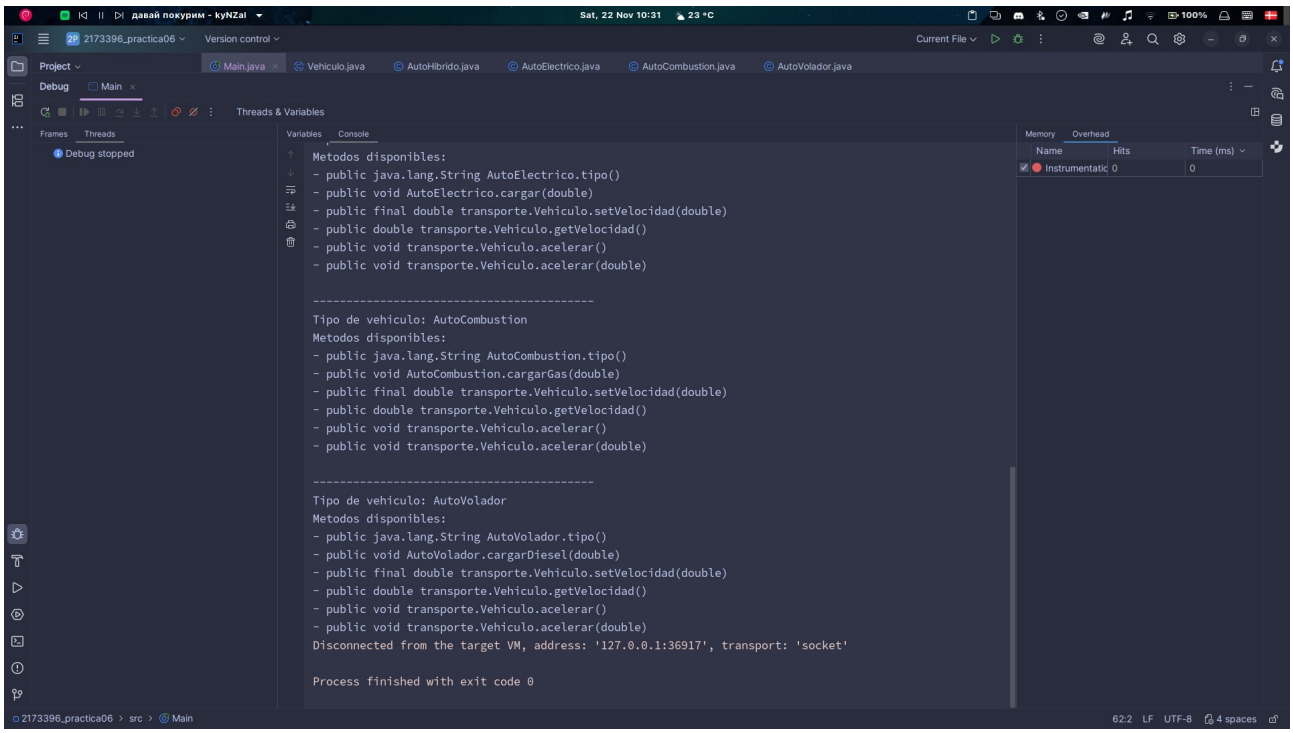
El programa compiló adecuadamente.





Depuración.

El programa se depuró adecuadamente.


давай покурим - kyNZal
Sat, 22 Nov 10:31
23 °C
100%
2173396_practica06
Version control
Current File
Project
Main.java
Vehiculo.java
AutoHibrido.java
AutoElectrico.java
AutoCombustion.java
AutoVolador.java
Debug
Main
Threads & Variables
Frames
Threads
Debug stopped
Variables
Console
Memory
Overhead
Name
Hits
Time (ms)
Instrumentatic
0
0
Metodos disponibles:
- public java.lang.String AutoElectrico.tipo()
- public void AutoElectrico.cargar(double)
- public final double transporte.Vehiculo.setVelocidad(double)
- public double transporte.Vehiculo.getVelocidad()
- public void transporte.Vehiculo.acelerar()
- public void transporte.Vehiculo.acelerar(double)

Tipo de vehiculo: AutoCombustion
Metodos disponibles:
- public java.lang.String AutoCombustion.tipo()
- public void AutoCombustion.cargarGas(double)
- public final double transporte.Vehiculo.setVelocidad(double)
- public double transporte.Vehiculo.getVelocidad()
- public void transporte.Vehiculo.acelerar()
- public void transporte.Vehiculo.acelerar(double)

Tipo de vehiculo: AutoVolador
Metodos disponibles:
- public java.lang.String AutoVolador.tipo()
- public void AutoVolador.cargarDiesel(double)
- public final double transporte.Vehiculo.setVelocidad(double)
- public double transporte.Vehiculo.getVelocidad()
- public void transporte.Vehiculo.acelerar()
- public void transporte.Vehiculo.acelerar(double)
Disconnected from the target VM, address: '127.0.0.1:36917', transport: 'socket'
Process finished with exit code 0
2173396_practica06 › src › Main
62:2
LF
UTF-8
4 spaces

Código.

El código del programa.

Main.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(Toyota.tipo());
        Toyota.setVelocidad(200.00);
        System.out.println(Toyota.getVelocidad());
        Toyota.acelerar();
        Toyota.cargarGas( litrosGas: 45);
        System.out.println("----------\n");

        // Volador
        AutoVolador Boeing = new AutoVolador( marca: "boeing",  modelo: "ct-70",  velocic
        System.out.println(Boeing.tipo());
        Boeing.setVelocidad(200.00);
        System.out.println(Boeing.getVelocidad());
        Boeing.acelerar();
        Boeing.cargarDiesel( litrosDiesel: 450);
        System.out.println("----------\n");

        List<Vehiculo> flota = Arrays.asList(BYD, Tesla, Toyota, Boeing);
        for(Vehiculo v : flota) {
            Class<?> ClaseVehiculo = v.getClass();
            System.out.println("\n----------");
            System.out.println("Tipo de vehiculo: " + ClaseVehiculo.getSimpleName())
            Method[] metodos = ClaseVehiculo.getMethods();
            System.out.println("Metodos disponibles: ");

            for(Method metodo : metodos) {
                if (!metodo.getDeclaringClass().equals(Object.class)) {
                    System.out.println("- " + metodo.toGenericString());
                }
            }
        }

    }
```

Vehiculo.java

```java
package transporte;

public abstract class Vehiculo {  11 usages  4 inheritors
    private double velocidad;  3 usages
    protected String marca;  1 usage
    protected String modelo;  1 usage

    public Vehiculo() {  no usages

    }

    public Vehiculo(String marca, String modelo) {  no usages
        this(marca, modelo,  velocidad: 0.0);
    }

    public Vehiculo(String marca, String modelo, double velocidad) {  5 usages
        this.marca = marca;
        this.modelo = modelo;
        setVelocidad(velocidad);
    }

    public double getVelocidad() {  6 usages
        return this.velocidad;
    }

    public final double setVelocidad(double v) {  7 usages
        if (v >= 0) {
            this.velocidad = v;
        }
        return v;
    }

    public void acelerar() {  4 usages
        setVelocidad(getVelocidad() + 10);
    }
```

AutoCombustion.java

```java
import tipoDeMotor.ICombustion;
import transporte.Vehiculo;

public class AutoCombustion extends Vehiculo implements ICombustion {  2 usages

    public AutoCombustion(String marca, String modelo, double velocidad) { super(mar

    @Override  2 usages
    public void cargarGas(double litrosGas) {
        System.out.println("Cargando vehículo con gasolina...");
        System.out.println("Combustible: " + litrosGas);
    }

    @Override  4 usages
    public String tipo() { return "Auto Combustion"; }
}
```

AutoVolador.java

```java
import tipoDeMotor.IVolador;
import transporte.Vehiculo;

public class AutoVolador extends Vehiculo implements IVolador {

    public AutoVolador(String marca, String modelo, double velocidad) { super(marca

    @Override
    public void cargarDiesel(double litrosDiesel) {
        System.out.println("Cargando vehículo volador con diesel...");
        System.out.println("Tanque: " + litrosDiesel);
    }

    @Override
    public String tipo() { return "Auto Volador"; }
}
```

AutoElectrico.java

```java
import tipoDeMotor.IElectrico;
import transporte.Vehiculo;

public class AutoElectrico extends Vehiculo implements IElectrico {  2 usages

    public AutoElectrico(String marca, String modelo, double velocidad) { super(marc

    @Override  2 usages
    public void cargar(double KWh) {
        System.out.println("Cargando vehículo modo eléctrico...");
        System.out.println("Batería: " + KWh);
    }

    @Override  4 usages
    public String tipo() { return "Auto Eléctrico"; }
}
```

AutoHibrido.java

```java
import tipoDeMotor.ICombustion;
import tipoDeMotor.IElectrico;
import transporte.Vehiculo;

public class AutoHibrido extends Vehiculo implements IElectrico, ICombustion {  2 usa

    public AutoHibrido(String marca, String modelo, double velocidad) { super(marca

    @Override  2 usages
    public void cargar(double KWh) {
        System.out.println("Cargando vehículo modo eléctrico...");
        System.out.println("Batería: " + KWh);
    }

    @Override  2 usages
    public void cargarGas(double litrosGas) {
```

IVolador.java

```java
package tipoDeMotor;

public interface IVolador {  2 usages  1 implementation
    void cargarDiesel(double litrosDiesel);  1 usage
}
```

IElectrico.java

```java
package tipoDeMotor;

public interface IElectrico {  4 usages  2 implementations
    void cargar(double KWh);  2 usages  2 implementations
}
```

ICombustion.java

```java
package tipoDeMotor;

public interface ICombustion {  4 usages  2 implementations
    void cargarGas(double litrosGas);  2 usages  2 implemen
}
```

Archivos.

Los archivos del programa.