UNIVERSIDAD AUTONOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FISICO MATEMATICAS

"*Practica 6"*

UNIDAD DE APRENDIZAJE: Programación Orientada a Objetos

MAESTRO: JORGE ALBERTO ISLAS PINEDA

CARRERA: LICENCIATUA EN CIENCIAS COMPUTACIONALES

25/09/25

Output:

```
run:
Tipo de b: H�brido
Acelerando... Nueva velocidad: 55.5 km/h

--- hibrido ---
Descripci�n: Veh�culo: Toyota Prius - Velocidad: 55.5 km/h - Tipo: H�brido - Combustible: 60.0% - Bater�a: 70%
Tipo: H�brido
Acelerando... Nueva velocidad: 65.5 km/h
Acelerando 25.0 km/h... Nueva velocidad: 90.5 km/h
Frenando 10.0 km/h... Nueva velocidad: 80.5 km/h
Es de combusti�n - Tipo: H�brido
H�brido repostado. Nivel de combustible: 100.0%
Es el�ctrico - Bater�a: 70%
H�brido cargado. Nivel de bater�a: 100%

--- hibrido ---
Descripci�n: Veh�culo: Ford Fusion Hybrid - Velocidad: 40.0 km/h - Tipo: H�brido - Combustible: 60.0% - Bater�a: 70%
Tipo: H�brido
Acelerando... Nueva velocidad: 50.0 km/h
Acelerando 25.0 km/h... Nueva velocidad: 75.0 km/h
Frenando 10.0 km/h... Nueva velocidad: 65.0 km/h
Es de combusti�n - Tipo: H�brido
H�brido repostado. Nivel de combustible: 100.0%
Es el�ctrico - Bater�a: 70%
H�brido cargado. Nivel de bater�a: 100%

--- auto ---
Descripci�n: Veh�culo: Nissan Sentra - Velocidad: 0.0 km/h - Tipo: Auto - Combustible: Gasolina
Tipo: Auto
Acelerando... Nueva velocidad: 10.0 km/h
Acelerando 25.0 km/h... Nueva velocidad: 35.0 km/h
Frenando 10.0 km/h... Nueva velocidad: 25.0 km/h
Es de combusti�n - Tipo: Gasolina
Auto repostado. Nivel de combustible: 100.0%

--- moto ---
Descripci�n: Veh�culo: Yamaha MT-07 - Velocidad: 0.0 km/h - Tipo: Moto - Combustible: Gasolina
Tipo: Moto
Acelerando... Nueva velocidad: 10.0 km/h
Acelerando 25.0 km/h... Nueva velocidad: 35.0 km/h
Frenando 10.0 km/h... Nueva velocidad: 25.0 km/h
Es de combusti�n - Tipo: Gasolina
Moto repostada. Nivel de combustible: 100.0%

--- electrico ---
Descripci�n: Veh�culo: Chevrolet Bolt EV - Velocidad: 0.0 km/h - Tipo: Auto El�ctrico - Bater�a: 80%
Tipo: Auto El�ctrico
Acelerando... Nueva velocidad: 10.0 km/h
Acelerando 25.0 km/h... Nueva velocidad: 35.0 km/h
Frenando 10.0 km/h... Nueva velocidad: 25.0 km/h
Es el�ctrico - Bater�a: 80%
Auto el�ctrico cargado. Nivel de bater�a: 100%

=== DEMOSTRANDO AutoCloseable (RecursoLog) ===
Recurso Sistema de Entretenimiento inicializado.
Usando recurso: Sistema de Entretenimiento
Operaciones finalizadas.
Cerrando recurso: Sistema de Entretenimiento - Limpieza completada.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Practica8

```
package Runpractica8;

import java.util.Arrays;
import java.util.List;

/**
 *
 * @author garza
 */
public class Practica8 {

 public static void main(String[] args) {

hibrido b = new hibrido("Toyota", "Prius", 45.50);
System.out.println("Tipo de b: " + b.tipo());
b.acelerar();

vehiculo g = new hibrido("Ford", "Fusion Hybrid", 40.00);

auto auto = new auto("Nissan", "Sentra", 0.0);
moto moto = new moto("Yamaha", "MT-07", 0.0);
electrico electrico = new electrico("Chevrolet", "Bolt EV", 0.0);

List<vehiculo> flota = Arrays.asList(b, g, auto, moto, electrico);

for (vehiculo v : flota) {
    System.out.println("\n--- " + v.getClass().getSimpleName() + " ---");
```

```
    System.out.println("Descripción: " + v.describir());
    System.out.println("Tipo: " + v.tipo());

    // Acelerar usando sobrecarga de métodos
    v.acelerar();      // Sin parámetros
    v.acelerar(25.0);  // Con parámetros
    v.frenar(10.0);

    // Demostrar casting y verificación de interfaces
    if (v instanceof combustion) {
        System.out.println("Es de combustión - Tipo: " + ((combustion) v).tipoCombustible());
        ((combustion) v).repostar();
    }

    if (v instanceof electricidad) {
        System.out.println("Es eléctrico - Batería: " + ((electricidad) v).nivelBateria() + "%");
        ((electricidad) v).cargar();
    }
}

System.out.println("\n=== DEMOSTRANDO AutoCloseable (RecursoLog) ===");

try (recurso recurso = new recurso("Sistema de Entretenimiento")) {
    recurso.usar();
    System.out.println("Operaciones finalizadas.");
}
}
}
```

Auto

```
package Runpractica8;

/**
 *
 * @author garza
 */
public class auto extends vehiculo implements combustion {
 private String tipoCombustible;
 private double nivelCombustible;

 public auto(String marca, String modelo, double velocidad) {
 super(marca, modelo, velocidad);
 this.tipoCombustible = "Gasolina";
 this.nivelCombustible = 50.0; // Tanque al 50%
 }

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

 @Override
 public void repostar() {
 nivelCombustible = 100.0;
 System.out.println("Auto repostado. Nivel de combustible: " + nivelCombustible + "%");
 }
```

```
@Override
public String tipoCombustible() {
return tipoCombustible;
}

@Override
public String describir() {
return super.describir() + " - Tipo: " + tipo() + " - Combustible: " + tipoCombustible;
}
}
```

Combustion

```
package Runpractica8;

/**
 *
 * @author garza
 */
public interface combustion {
 void repostar();
 String tipoCombustible();
}
```

Electricidad

```
package Runpractica8;

/**
 *
```

```
 * @author garza
 */
public interface electricidad {
 void cargar();
 int nivelBateria();
}
```

Electrico

```
package Runpractica8;

/**
 *
 * @author garza
 */
public class electrico extends vehiculo implements electricidad {
 private int nivelBateria;

 public electrico(String marca, String modelo, double velocidad) {
 super(marca, modelo, velocidad);
 this.nivelBateria = 80; // Batería al 80%
 }

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

 @Override
 public void cargar() {
```

```
 nivelBateria = 100;
 System.out.println("Auto eléctrico cargado. Nivel de batería: " + nivelBateria + "%");
 }
 @Override
 public int nivelBateria() {
 return nivelBateria;
 }
 @Override
 public String describir() {
 return super.describir() + " - Tipo: " + tipo() + " - Batería: " + nivelBateria + "%";
 }
}
```

Hibrido

```
package Runpractica8;

/**
 *
 * @author garza
 */
public class hibrido extends vehiculo implements combustion, electricidad {
 private String tipoCombustible;
 private double nivelCombustible;
 private int nivelBateria;
 public hibrido(String marca, String modelo, double velocidad) {
 super(marca, modelo, velocidad);
 this.tipoCombustible = "Híbrido";
 this.nivelCombustible = 60.0; // Tanque al 60%
 this.nivelBateria = 70; // Batería al 70%
```

```
}
@Override
public String tipo() {
return "Híbrido";
}
@Override
public void repostar() {
nivelCombustible = 100.0;
System.out.println("Híbrido repostado. Nivel de combustible: " + nivelCombustible + "%");
}
@Override
public String tipoCombustible() {
return tipoCombustible;
}
@Override
public void cargar() {
nivelBateria = 100;
System.out.println("Híbrido cargado. Nivel de batería: " + nivelBateria + "%");
}
@Override
public int nivelBateria() {
return nivelBateria;
}

@Override
public String describir() {
return super.describir() + " - Tipo: " + tipo() +
" - Combustible: " + nivelCombustible + "%" +
```

```
 " - Batería: " + nivelBateria + "%";
 }
}
```

Moto

```
package Runpractica8;

/**
 *
 * @author garza
 */
public class moto extends vehiculo implements combustion {
 private String tipoCombustible;
 private double nivelCombustible;

 public moto(String marca, String modelo, double velocidad) {
 super(marca, modelo, velocidad);
 this.tipoCombustible = "Gasolina";
 this.nivelCombustible = 40.0; // Tanque al 40%
 }

 @Override
 public String tipo() {
 return "Moto";
 }

 @Override
 public void repostar() {
 nivelCombustible = 100.0;
```

```
System.out.println("Moto repostada. Nivel de combustible: " + nivelCombustible + "%");
}

@Override
public String tipoCombustible() {
return tipoCombustible;
}

@Override
public String describir() {
return super.describir() + " - Tipo: " + tipo() + " - Combustible: " + tipoCombustible;
}
}
```

Recurso

```
package Runpractica8;

/**
 *
 * @author garza
 */
public class recurso implements AutoCloseable {
 private String nombreRecurso;

 public recurso(String nombreRecurso) {
 this.nombreRecurso = nombreRecurso;
 System.out.println("Recurso " + nombreRecurso + " inicializado.");
 }
```

```
 public void usar() {
 System.out.println("Usando recurso: " + nombreRecurso);
 }

 @Override
 public void close() {
 System.out.println("Cerrando recurso: " + nombreRecurso + " - Limpieza completada.");
 }
}
```

Vehículo

```
package Runpractica8;

/**
 *
 * @author garza
 */
public abstract class vehiculo {
 private double velocidad;
 protected String marca;
 protected String modelo;

 public vehiculo(String marca, String modelo, double velocidad) {
 this.marca = marca;
 this.modelo = modelo;
 this.velocidad = velocidad;
 }

 // Método abstracto que debe ser implementado por las clases hijas
```

```
public abstract String tipo();

public String describir() {
return "Vehículo: " + marca + " " + modelo + " - Velocidad: " + velocidad + " km/h";
}

// Sobrecarga de métodos acelerar
public void acelerar() {
velocidad += 10;
System.out.println("Acelerando... Nueva velocidad: " + velocidad + " km/h");
}

public void acelerar(double incremento) {
velocidad += incremento;
System.out.println("Acelerando " + incremento + " km/h... Nueva velocidad: " + velocidad + " km/h");
}

public void frenar(double decremento) {
velocidad = Math.max(0, velocidad - decremento);
System.out.println("Frenando " + decremento + " km/h... Nueva velocidad: " + velocidad + " km/h");
}

// Getters
public double getVelocidad() {
return velocidad;
}
```

```
public String getMarca() {
return marca;
}

public String getModelo() {
return modelo;
}
}
```