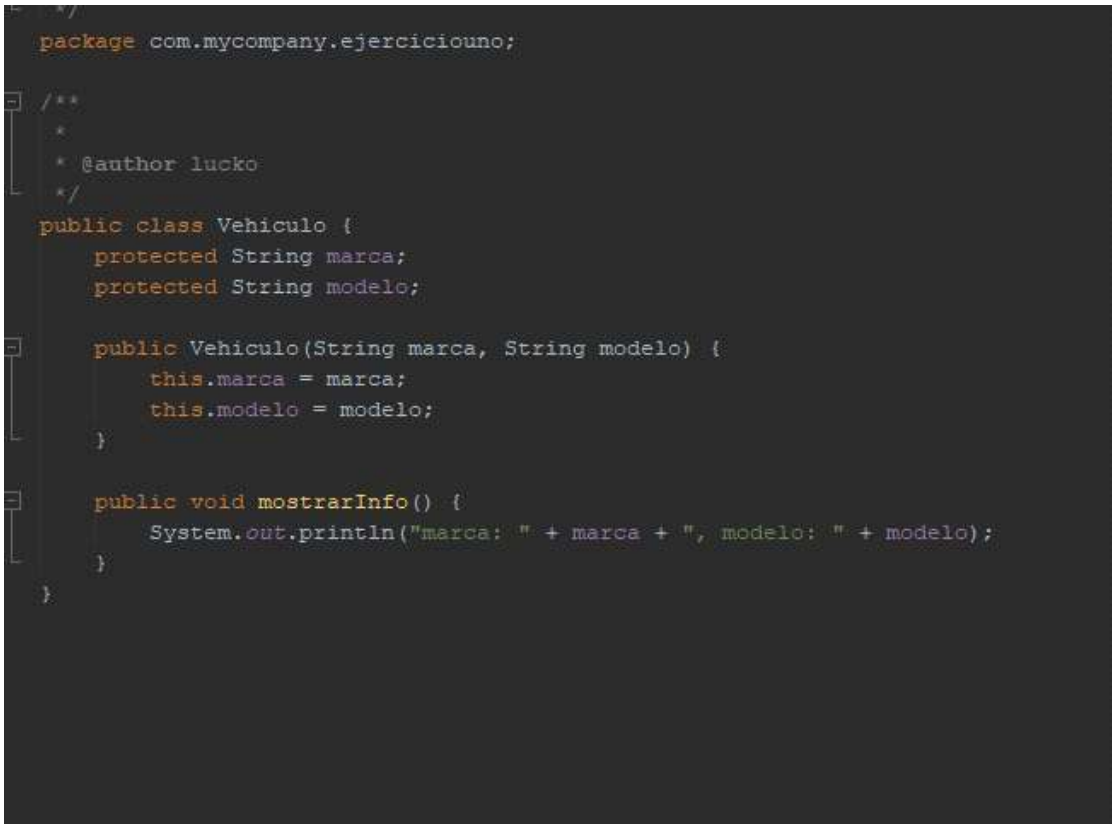
1. Vehículos y herencia básica

● Clase base: Vehículo con atributos marca, modelo y método mostrarInfo()



```
*/
package com.mycompany.ejerciciouno;

/**
 *
 * @author lucko
 */
public class Vehiculo {
    protected String marca;
    protected String modelo;

    public Vehiculo(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }

    public void mostrarInfo() {
        System.out.println("marca: " + marca + ", modelo: " + modelo);
    }
}
```

● Subclase: Auto con atributo adicional cantidadPuertas, sobrescribe mostrarInfo()

```java
*/
package com.mycompany.ejerciciouno;

/**
 *
 * @author lucko
 */
public class Auto extends Vehiculo {
    private int cantidadPuertas;

    public Auto(String marca, String modelo, int cantidadPuertas) {
        super(marca, modelo);
        this.cantidadPuertas = cantidadPuertas;
    }

    @Override
    public void mostrarInfo() {
        super.mostrarInfo();
        System.out.println("puertas: " + cantidadPuertas);
    }
}
```

● Tarea: Instanciar un auto y mostrar su información completa.

```java
*/
public class EjercicioUno {

    public static void main(String[] args) {
        Auto auto = new Auto("toyota", "corolla", 4);
        auto.mostrarInfo();
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ EjercicioUno ---
marca: toyota, modelo: corolla
puertas: 4
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.925 s
Finished at: 2025-10-22T14:04:03-03:00
```

2. Figuras geométricas y métodos abstractos

● Clase abstracta: Figura con método calcularArea() y atributo nombre

```
 * @author lucko
 */
abstract class Figura {
    protected String nombre;

    public Figura(String nombre) {
        this.nombre = nombre;
    }

    public abstract double calcularArea();

    public void mostrarArea() {
        System.out.println(nombre + " area: " + calcularArea());
    }
}
```

● Subclases: Círculo y Rectángulo implementan el cálculo del área

```
 *
 * @author lucko
 */
public class Circulo extends Figura {
    private double radio;

    public Circulo(double radio) {
        super("circulo");
        this.radio = radio;
    }

    @Override
    public double calcularArea() {
        return Math.PI * radio * radio;
    }
}
```

```java
 * @author lucko
 */
public class Rectangulo extends Figura {
    private double base, altura;

    public Rectangulo(double base, double altura) {
        super("rectangulo");
        this.base = base;
        this.altura = altura;
    }

    @Override
    public double calcularArea() {
        return base * altura;
    }
}
```

● Tarea: Crear un array de figuras y mostrar el área de cada una usandopolimorfismo.

```java
 */
public class EjercicioDos {

    public static void main(String[] args) {
        Figura[] figuras = {
            new Circulo(5),
            new Rectangulo(4, 6)
        };

        for (Figura figura : figuras) {
            figura.mostrarArea();
        }
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ EjercicioDos ---
circulo area: 78.53981633974483
rectangulo area: 24.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.962 s
Finished at: 2025-10-22T14:07:20-03:00
```

3. Empleados y polimorfismo

● Clase abstracta: Empleado con método calcularSueldo()

```
 */
abstract class Empleado {
    protected String nombre;

    public Empleado(String nombre) {
        this.nombre = nombre;
    }

    public abstract double calcularSueldo();
}
```

● Subclases: EmpleadoPlanta, EmpleadoTemporal

```
 * @author lucko
 */
class EmpleadoPlanta extends Empleado {
    private double sueldoBase;

    public EmpleadoPlanta(String nombre, double sueldoBase) {
        super(nombre);
        this.sueldoBase = sueldoBase;
    }

    @Override
    public double calcularSueldo() {
        return sueldoBase;
    }
}
```

```java
 */
class EmpleadoTemporal extends Empleado {
    private double pagoPorHora;
    private int horasTrabajadas;

    public EmpleadoTemporal(String nombre, double pagoPorHora, int horasTrabajadas) {
        super(nombre);
        this.pagoPorHora = pagoPorHora;
        this.horasTrabajadas = horasTrabajadas;
    }

    @Override
    public double calcularSueldo() {
        return pagoPorHora * horasTrabajadas;
    }
}
```

● Tarea: Crear lista de empleados, invocar calcularSueldo() polimórficamente, usar instanceof para clasificar

```java
public class EjercicioTres {

    public static void main(String[] args) {
         List<Empleado> empleados = Arrays.asList(
            new EmpleadoPlanta("ana", 2500),
            new EmpleadoTemporal("luis", 20, 80)
        );

        for (Empleado emp : empleados) {
            System.out.println(emp.nombre + " suedo: " + emp.calcularSueldo());
            if (emp instanceof EmpleadoPlanta) {
                System.out.println("empleado de planta");
                System.out.println();
            } else if (emp instanceof EmpleadoTemporal) {
                System.out.println("empleado temporal");
            }
        }
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ Ejerc
ana suedo: 2500.0
empleado de planta

luis suedo: 1600.0
empleado temporal
------------------------------------------
BUILD SUCCESS
------------------------------------------
Total time:  0.954 s
```

4. Animales y comportamiento sobrescrito

● Clase: Animal con método hacerSonido() y describirAnimal()

```java
*/
public class Animal {
    protected String nombre;

    public Animal(String nombre) {
        this.nombre = nombre;
    }

    public void hacerSonido() {
        System.out.println(nombre + " hace sonido");
    }

    public void describirAnimal() {
        System.out.println("soy " + nombre);
    }
}
```

● Subclases: Perro, Gato, Vaca sobrescriben hacerSonido() con @Override

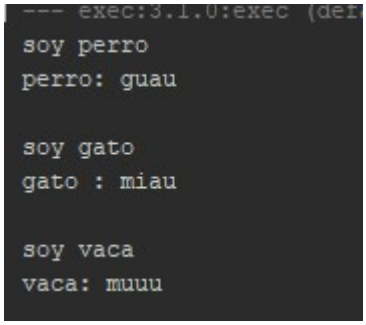
```java
 *
 * @author lucko
 */
class Gato extends Animal {
    public Gato() {
        super("gato ");
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + ": miau");
    }
}
```

```java
 */
class Vaca extends Animal {
    public Vaca() {
        super("vaca");
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + ": muuu");
    }
}
```

```java
 *
 * @author lucko
 */
class Perro extends Animal {
    public Perro() {
        super("perro");
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + ": guau");
    }
}
```

● Tarea: Crear lista de animales y mostrar sus sonidos con polimorfismo

```java
 */
public class EjercicioCuatro {

    public static void main(String[] args) {
        List<Animal> animales = Arrays.asList(
            new Perro(),
            new Gato(),
            new Vaca()
        );

        for (Animal animal : animales) {
            animal.describirAnimal();
            animal.hacerSonido();
            System.out.println();
        }
    }
}
```




LINK AL REPO:

https://github.com/BarreraMariano/UTN-TUPaD-P2/tree/main/7-Herencia%20y%20Polimorfismo