



**CENTRO UNIVERSITARIO UAEM
ATLACOMULCO**

MATERIA

**PARADIGMAS DE LA
PROGRAMACIÓN**

DOCENTE

JULIO ALBERTO DE LA TEJA LÓPEZ

ALUMNO

ALEXIS VALENCIA MARTÍNEZ

CARRERA

**LICENCIATURA EN INGENIERÍA EN
SISTEMAS COMPUTACIONALES**

ICO-27

FECHA DE ENTREGA

21/AGOSTO/2023

```

Class Main {
    Public static void main(String[] args) {
        Circulo circulo = new Circulo(5);
        System.out.println("El área del círculo es: " + circulo.calcularArea());

        Rectangulo rectangulo = new Rectangulo(4, 2);
        System.out.println("El área del rectángulo es: " + rectangulo.calcularArea());

        Triangulo triangulo = new Triangulo(3, 6);
        System.out.println("El área del triángulo es: " + triangulo.calcularArea());
    }
}

```

```

Abstract class FiguraGeometrica {
    Protected String nombre;

    Public FiguraGeometrica(String nombre) {
        This.nombre = nombre;
    }

    Public abstract double calcularArea();
}

```

```

Class Circulo extends FiguraGeometrica {
    Private double radio;

    Public Circulo(double radio) {
        Super("Circulo");
        This.radio = radio;
    }

    @Override
    Public double calcularArea() {
        Return Math.PI * Math.pow(radio, 2);
    }
}

```

```

Class Rectangulo extends FiguraGeometrica {
    Private double base;
    Private double altura;

    Public Rectangulo(double base, double altura) {
        Super("Rectangulo");
        This.base = base;
        This.altura = altura;
    }

    @Override

```

```

    Public double calcularArea() {
        Return base * altura;
    }
}

```

```

Class Triangulo extends FiguraGeometrica {
    Private double base;
    Private double altura;

```

```

    Public Triangulo(double base, double altura) {
        Super("Triangulo");
        This.base = base;
        This.altura = altura;
    }

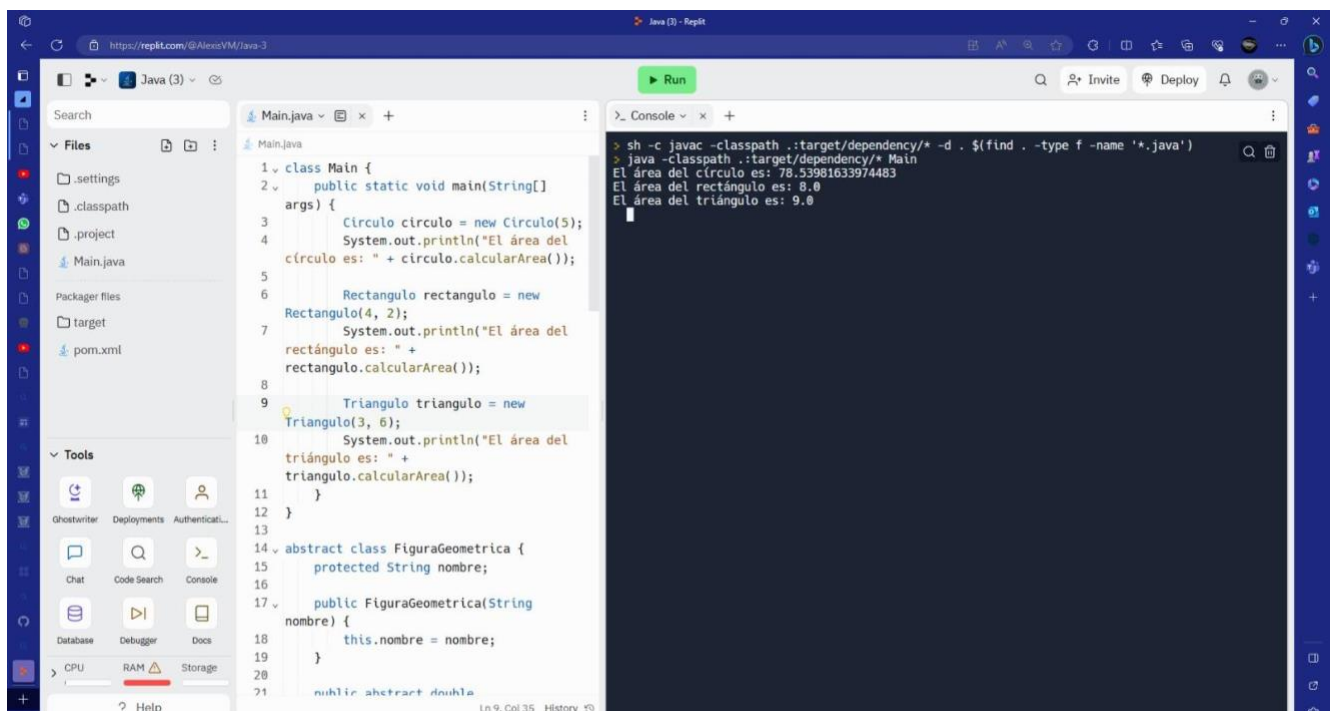
```

@Override

```

    Public double calcularArea() {
        Return (base * altura) / 2;
    }
}

```



```
Public class Main {
    Public static class Personaje {
        String nombre;
        Int nivel;

        Public Personaje(String nombre, int nivel) {
            This.nombre = nombre;
            This.nivel = nivel;
        }

        Public void atacar() {
            System.out.println(nombre + " ataca!");
        }
    }

    Public static class Jugador extends Personaje {
        String clase;

        Public Jugador(String nombre, int nivel, String clase) {
            Super(nombre, nivel);
            This.clase = clase;
        }

        Public void usarHabilidadEspecial() {
            System.out.println(nombre + " usa su habilidad especial!");
        }
    }

    Public static class Enemigo extends Personaje {
        String tipo;

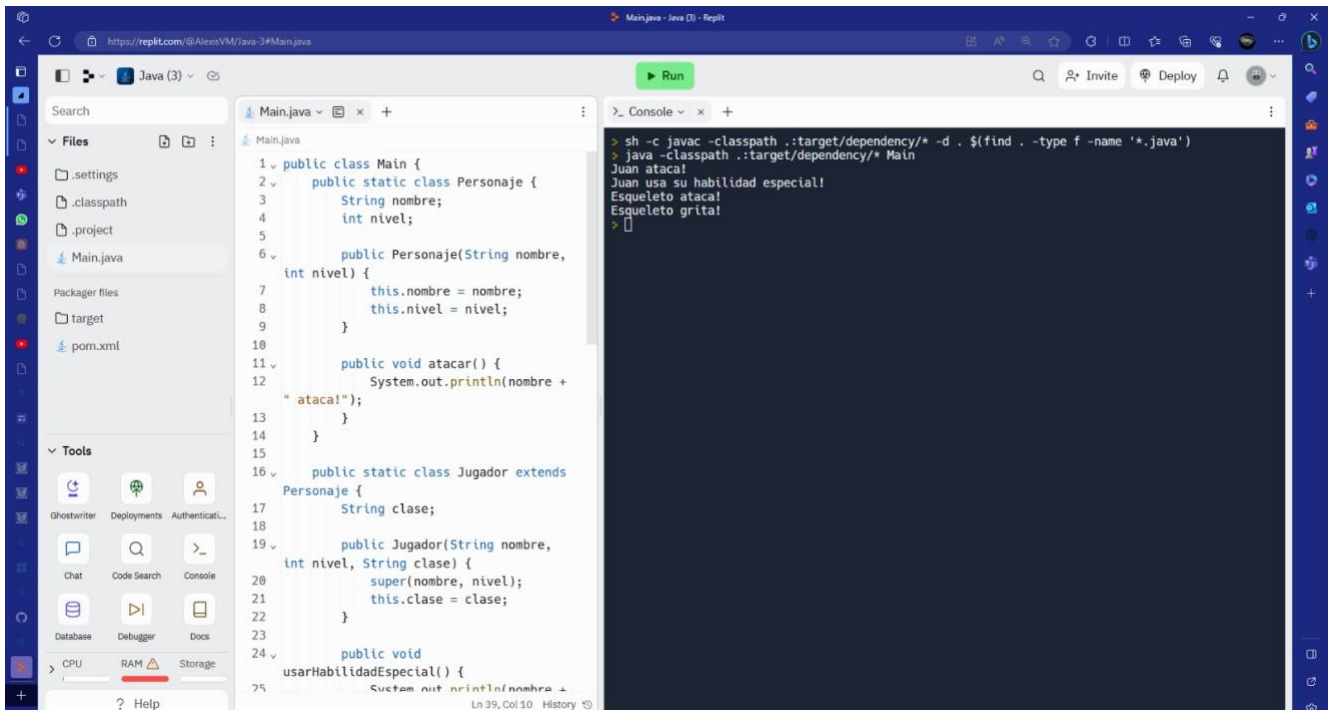
        Public Enemigo(String nombre, int nivel, String tipo) {
            Super(nombre, nivel);
            This.tipo = tipo;
        }

        Public void gritar() {
            System.out.println(nombre + " grita!");
        }
    }

    Public static void main(String[] args) {
        Jugador jugador1 = new Jugador("Juan", 1, "Guerrero");
        Enemigo enemigo1 = new Enemigo("Esqueleto", 1, "Esqueleto");

        Jugador1.atacar();
        Jugador1.usarHabilidadEspecial();
    }
}
```

```
    Enemigo1.atacar();  
    Enemigo1.gritar();  
}  
}
```



The screenshot shows a Replit IDE window titled "Main.java - Java (3) - Replit". The left sidebar displays the file explorer with folders like ".settings", ".classpath", ".project", and files like "Main.java" and "pom.xml". The main editor area shows the following Java code:

```
1 public class Main {  
2     public static class Personaje {  
3         String nombre;  
4         int nivel;  
5  
6         public Personaje(String nombre,  
7             int nivel) {  
8             this.nombre = nombre;  
9             this.nivel = nivel;  
10        }  
11  
12        public void atacar() {  
13            System.out.println(nombre +  
14                " ataca!");  
15        }  
16  
17        public static class Jugador extends  
18            Personaje {  
19            String clase;  
20  
21            public Jugador(String nombre,  
22                int nivel, String clase) {  
23                super(nombre, nivel);  
24                this.clase = clase;  
25            }  
26  
27            public void  
28                usarHabilidadEspecial() {  
29                System.out.println(nombre +  
30                    " usa su habilidad especial!");  
31            }  
32        }  
33    }  
34 }
```

The right sidebar shows the console output after running the code:

```
> sh -c javac -classpath ./target/dependency/* -d . $(find . -type f -name '*.java')  
> java -classpath ./target/dependency/* Main  
Juan ataca!  
Juan usa su habilidad especial!  
Esqueleto ataca!  
Esqueleto grita!  
>
```

```
Class Main {
    Public static void main(String[] args) {
        PaletaAgua paletaAgua = new PaletaAgua("Fresa", 10.0, true);
        paletaAgua.mostrarInformacion();
        paletaAgua.mostrarBaseAgua();

        PaletaCrema paletaCrema = new PaletaCrema("Chocolate", 15.0, true);
        paletaCrema.mostrarInformacion();
        paletaCrema.mostrarTexturaCremosa();
    }
}
```

```
Class Paleta {
    Protected String sabor;
    Protected double precio;

    Public Paleta(String sabor, double precio) {
        This.sabor = sabor;
        This.precio = precio;
    }

    Public void mostrarInformacion() {
        System.out.println("Sabor: " + this.sabor);
        System.out.println("Precio: " + this.precio);
    }
}
```

```
Class PaletaAgua extends Paleta {
    Private boolean baseAgua;

    Public PaletaAgua(String sabor, double precio, boolean baseAgua) {
        Super(sabor, precio);
        This.baseAgua = baseAgua;
    }

    Public void mostrarBaseAgua() {
        System.out.println("Base de agua: " + (this.baseAgua ? "Sí" : "No"));
    }
}
```

```
Class PaletaCrema extends Paleta {
    Private boolean cremosa;

    Public PaletaCrema(String sabor, double precio, boolean cremosa) {
        Super(sabor, precio);
        This.cremosa = cremosa;
    }
}
```

```

Public void mostrarTexturaCremosa() {
    System.out.println("Textura cremosa: " + (this.cremosa ? "Si" : "No"));
}
}

```

The screenshot shows a Replit IDE window titled 'Main.java - Java (3) - Replit'. The left sidebar contains a file explorer with folders like '.settings', '.classpath', '.project', and files like 'Main.java' and 'pom.xml'. The main editor displays the following Java code:

```

30
31 public PaletaAgua(String sabor,
32 double precio, boolean baseAgua) {
33     super(sabor, precio);
34     this.baseAgua = baseAgua;
35 }
36
37 public void mostrarBaseAgua() {
38     System.out.println("Base de
39     agua: " + (this.baseAgua ? "Si" : "No"));
40 }
41
42 class PaletaCrema extends Paleta {
43     private boolean cremosa;
44
45     public PaletaCrema(String sabor,
46 double precio, boolean cremosa) {
47         super(sabor, precio);
48         this.cremosa = cremosa;
49     }
50
51     public void mostrarTexturaCremosa() {
52         System.out.println("Textura
53         cremosa: " + (this.cremosa ? "Si" :
54         "No"));
55     }
56 }

```

The right sidebar shows the console output, which is the result of running the code:

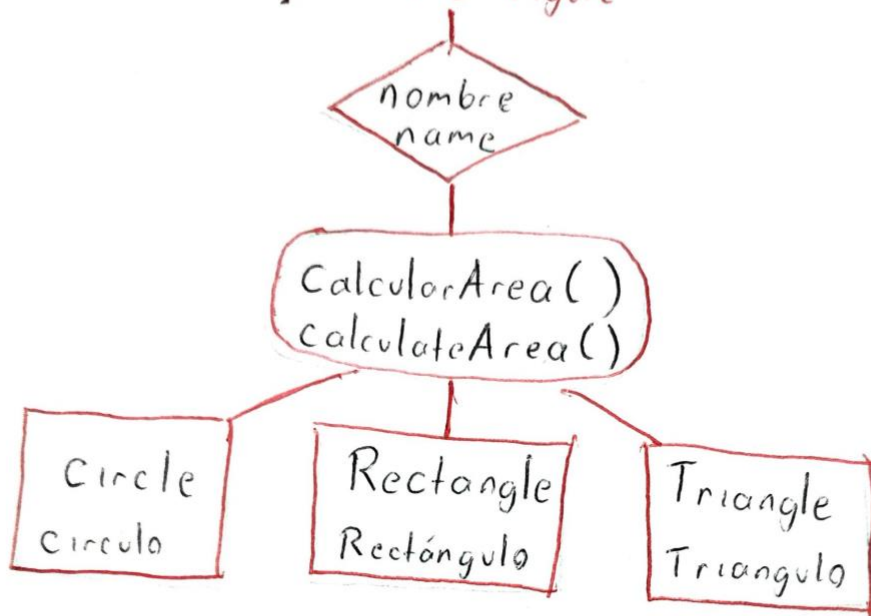
```

> sh -c javac -classpath .:target/dependency/* -d . $(find . -type f -name '*.java')
> java -classpath .:target/dependency/* Main
Sabor: Fresa
Precio: 10.0
Base de agua: Si
Sabor: Chocolate
Precio: 15.0
Textura cremosa: Si
>

```

En conclusión, el concepto de herencia en la programación orientada a objetos es una herramienta poderosa que permite la creación de clases más específicas a partir de clases más generales. Esto facilita la reutilización y organización del código, así como la implementación del polimorfismo. Los ejemplos proporcionados, que involucran diferentes tipos de personajes y paletas, demuestran cómo se puede utilizar la herencia para representar entidades que comparten algunos atributos y métodos con una clase base, pero también tienen sus propias características únicas. Además, la capacidad de traducir el diagrama UML y las instrucciones al inglés muestra la importancia de las habilidades de comunicación multilingüe en el mundo globalizado de hoy. En general, estos ejercicios han proporcionado valiosos conocimientos sobre las aplicaciones prácticas de la herencia y la traducción de idiomas en el desarrollo de software.

Figura Geométrica Geometric Figure



Personaje Character

