



**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

23/AGOSTO/2023

Ejercicio 1: Polimorfismo de las figuras Geométricas

De acuerdo con el ejercicio anterior “Herencia de la tarea 1:Figuras Geométricas”, crea a partir de Este el uso de polimorfismo usando varios objetos para sacar una tabla sobre la descripción de sus Áreas, (utiliza un arreglo de objetos y recórrelo con un For).

Codificar el ejercicio y actualiza tu diagrama en UML

- **CÓDIGO**

```
Public class Main {  
  
    Public static void main(String[] args) {  
  
        FiguraGeometrica[] figuras = new FiguraGeometrica[3];  
  
        Figuras[0] = new Circulo(5);  
  
        Figuras[1] = new Rectangulo(4, 2);  
  
        Figuras[2] = new Triangulo(3, 6);  
  
        For (FiguraGeometrica figura : figuras) {  
  
            System.out.println("La figura es un " + figura.nombre + " y su área es " + figura.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

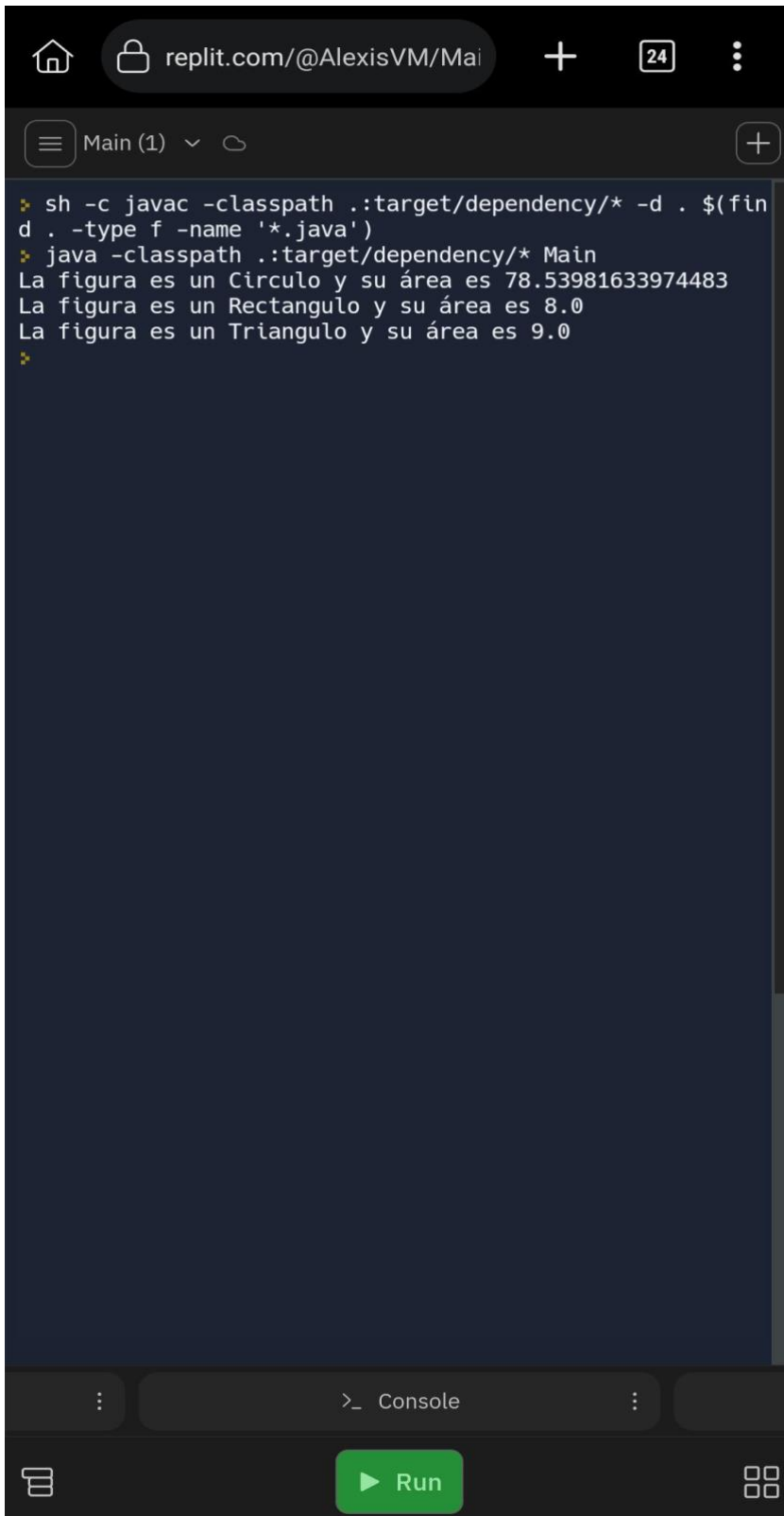
    Public double calcularArea() {

        Return (base * altura) / 2;

```

```
}  
}
```

- SALIDA



The screenshot shows a Replit web interface with a terminal window. The terminal displays the output of a Java compilation and execution process. The URL in the browser is `replit.com/@AlexisVM/Mai`. The terminal shows the following commands and output:

```
❖ sh -c javac -classpath .:target/dependency/* -d . $(find . -type f -name '*.java')  
❖ java -classpath .:target/dependency/* Main  
La figura es un Circulo y su área es 78.53981633974483  
La figura es un Rectangulo y su área es 8.0  
La figura es un Triangulo y su área es 9.0  
❖
```

At the bottom of the interface, there is a tab labeled ">_ Console" and a green "Run" button.

Ejercicio 2: Polimorfismo en personajes en un Videojuego

Aplica el concepto de polimorfismo al ejercicio anterior “Herencia de la tarea 1: Jerarquía de Personajes en un Videojuego”, explica como lo aplicaste y por qué tomaste esa decisión.

Codificar el ejercicio y actualiza tu diagrama en UML, colocar explicación

- **CÓDIGO**

```
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;  
  
        }  
  
        @Override  
  
        public void atacar() {  
  
            System.out.println(nombre + " ataca con su espada!");  
  
        }  
  
        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;
    }

    @Override

    public void atacar() {

        System.out.println(nombre + " ataca con su garra!");
    }

    public void gritar() {

        System.out.println(nombre + " grita!");
    }

}

public static void main(String[] args) {

    Personaje personaje1 = new Jugador("Juan", 1, "Guerrero");

    Personaje personaje2 = new Enemigo("Esqueleto", 1, "Esqueleto");


    personaje1.atacar();

    ((Jugador) personaje1).usarHabilidadEspecial();

    personaje2.atacar();

    ((Enemigo) personaje2).gritar();

}

}

```

In this example, an object of type Character is created and an instance of Player or Enemy is assigned to it. Then the attack() method is called and the corresponding implementation is executed in each case. This is possible thanks to polymorphism

- SALIDA



replit.com/@AlexisVM/Mai



24



Main (1) ▾



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



>_ Console



Run



Ejercicio 3: Polimorfismo en la paletería de Paletas en una Paletería

Po ultimo aplica también el concepto de polimorfismo al ejercicio anterior “Herencia de la tarea 1: Jerarquía de Paletas en una Paletería”, explica como lo aplicaste y por qué tomaste esa decisión.

Codificar el ejercicio y actualiza tu diagrama en UML -y anexa la explicación.

- **CÓDIGO**

```
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 ? "Sí" : "No"));
    }
}

```

POLYMORPHISM WAS ALREADY APPLIED IN THIS CLASS BECAUSE THEY INHERIT ATTRIBUTES FROM THE PALETTE CLASS

- **SALIDA**



replit.com/@AlexisVM/Mai



24



≡ Main (1) ▾ ☁



```
❯ 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: Sí
Sabor: Chocolate
Precio: 15.0
Textura cremosa: Sí
❯
```



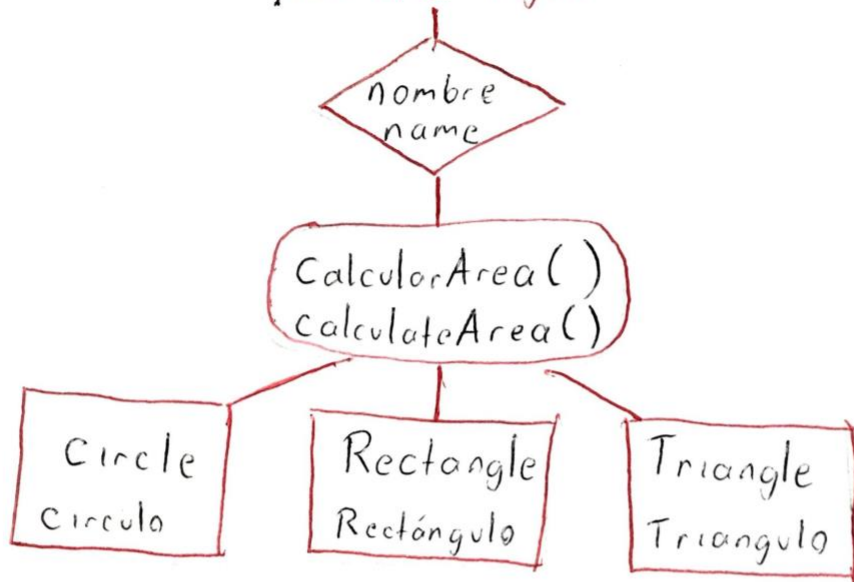
>_ Console



▶ Run



Figura Geométrica Geometric Figure



Personaje character

