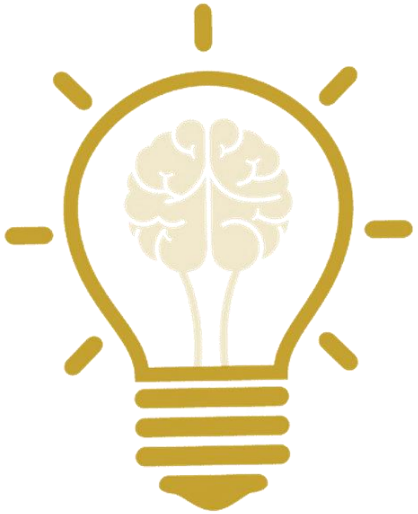

# PROGRAMACIÓN EN JAVA

Materia PJA-001 material creado por el Ing. Jim Requena

# Programación Orientada a Objetos en JAVA
# Tercera Parte
# (Herencia y Polimorfismo)



Materia PJA-001 material creado por el Ing. Jim Requena

# Objetivos de la sesión

- Comprender el concepto de herencia en Java
- Aprender a crear y utilizar clases heredadas
- Entender el polimorfismo y su aplicación
- Practicar con ejercicios de herencia y polimorfismo

# Herencia

- Permite crear nuevas clases basadas en clases existentes
- La nueva clase (subclase) hereda atributos y métodos de la clase base (superclase)
- Promueve la reutilización de código

```
public class Subclase extends Superclase {
    // Nuevos atributos y métodos
}
```

iTDS | Profesionales en Educación

# Ejmplo Herencia

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

        public void comer() {
                System.out.println(nombre + " está comiendo.");
        }
}

public class Perro extends Animal {
        public Perro(String nombre) {
                super(nombre);
}

        public void ladrar() {
                System.out.println(nombre + " está ladrando.");
        }
}
```

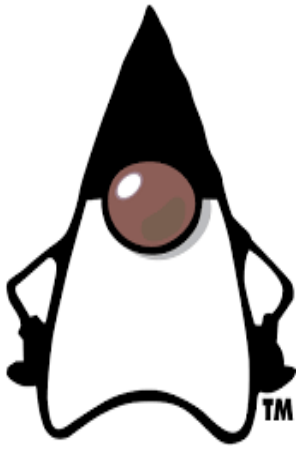
# Palabra Clave super

- Se usa para llamar al constructor de la superclase
- También se usa para acceder a métodos de la superclase

```java
public class Gato extends Animal {
    private int vidasRestantes;

    public Gato(String nombre, int vidas) {
        super(nombre);
        this.vidasRestantes = vidas;
    }

    @Override
    public void comer() {
        super.comer();
        System.out.println("El gato come pescado.");
    }
}
```



# Polimorfismo

- Permite que objetos de diferentes clases sean tratados como objetos de una clase base común

- Dos tipos principales: sobrecarga de métodos y sobreescritura de métodos

iTDS | Profesionales en Educación

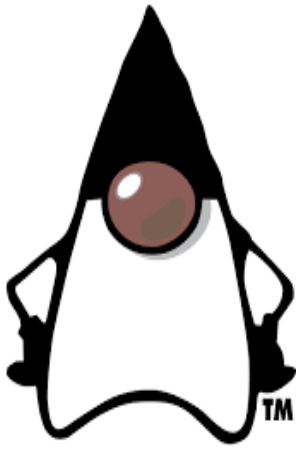
# Sobrecarga de métodos

- Múltiples métodos con el mismo nombre, pero diferentes parámetros

```java
public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }

    public double sumar(double a, double b) {
        return a + b;
    }

    public int sumar(int a, int b, int c) {
        return a + b + c;
    }
}
```



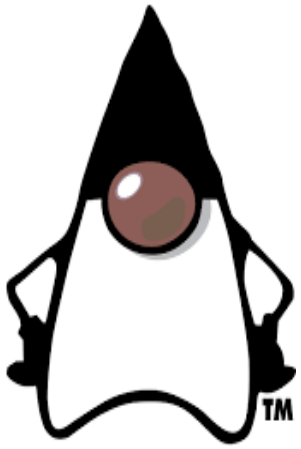
# Sobrecarga de métodos

- Múltiples métodos con el mismo nombre, pero diferentes parámetros

```java
public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }

    public double sumar(double a, double b) {
        return a + b;
    }

    public int sumar(int a, int b, int c) {
        return a + b + c;
    }
}
```



# Polimorfismo en Acción

```java
Animal[] animales = new Animal[3];
animales[0] = new Animal("Animal genérico");
animales[1] = new Perro("Fido");
animales[2] = new Gato("Whiskers", 9);

for (Animal animal : animales) {
    animal.hacerSonido();  // Llama al método apropiado para cada tipo
}
```

# Ejercicio práctico

Crear una jerarquía de clases para un sistema de figuras geométricas:

1. Clase base Figura con método calcularArea()
2. Subclases Circulo, Rectangulo y Triangulo
3. Implementar el cálculo de área para cada figura
4. Crear un array de Figura con diferentes tipos y calcular sus áreas

# Solución Ejercicio Practico 1/2

```java
public abstract class Figura {
    public abstract double calcularArea();
}

public class Circulo extends Figura {
    private double radio;

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

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

public class Rectangulo extends Figura {
    private double base;
    private double altura;

    public Rectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }
```

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

public class Triangulo extends Figura {
    private double base;
    private double altura;

    public Triangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

    @Override
    public double calcularArea() {
        return 0.5 * base * altura;
    }
}

public class Main {
    public static void main(String[] args) {
        Figura[] figuras = new Figura[3];
        figuras[0] = new Circulo(5);
        figuras[1] = new Rectangulo(4, 6);
        figuras[2] = new Triangulo(3, 4);

        for (Figura figura : figuras) {
            System.out.println("Área: " + figura.calcularArea());
        }
    }
}
```

# Solución Ejercicio Practico 2/2

# Tarea

1. Extiende el sistema de figuras geométricas para incluir una nueva figura: Hexagono
2. Implementa un método calcularPerimetro() en la clase Figura y sobreescríbelo en todas las subclases
3. Crea una interfaz Dibujable con un método dibujar() e impleméntala en las clases de figuras
4. Modifica el programa principal para que también muestre el perímetro y "dibuje" cada figura
5. Implementa una clase GrupoFiguras que pueda contener múltiples figuras y calcular el área total

# Recursos adicionales

- Java Documentation on Inheritance: docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
- Java Documentation on Polymorphism: docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

# Gracias!