1. Polimorfismo

El polimorfismo en Java es un concepto importante en programación orientada a objetos que permite que los objetos de diferentes clases se comporten de manera similar cuando se les trata de la misma manera. Esto significa que puedes tener una clase base con métodos y luego clases derivadas que sobrescriben esos métodos para tener su propia implementación. Cuando llamas a un método sobre un objeto, Java sabe automáticamente qué versión del método debe ejecutar basándose en el tipo de objeto al que estás haciendo referencia en ese momento.

2. Herencia

La herencia posibilita la creación de nuevas clases basadas en la estructura preexistente de otras clases, facilitando la reutilización de código y la organización jerárquica de las entidades. Las clases derivadas heredan tanto atributos como métodos de sus clases progenitoras, facilitando así la ampliación y adaptación del comportamiento de las clases.

3. Sobrecarga de Métodos

La sobrecarga de métodos permite la definición de múltiples funciones con el mismo nombre dentro de una clase, diferenciándolas mediante la variación de sus parámetros. Esta facultad ofrece una manera práctica de configurar interfaces más adaptables y fáciles de comprender, adaptando los métodos a diferentes tipos de datos o circunstancias.

4. Polimorfismo Paramétrico (Generics en Java)

El polimorfismo paramétrico, o generics en Java, posibilita la implementación de clases y métodos que operan sobre tipos genéricos, cuya especificación se realiza posteriormente. Esta técnica brinda un nivel de flexibilidad y seguridad en la manipulación de datos, permitiendo la creación de código capaz de interactuar con una variedad de tipos sin sacrificar la seguridad en la fase de compilación.

5. Polimorfismo de Inclusión

El polimorfismo de inclusión, también denominado subtipado o polimorfismo de subclases, describe la facultad de tratar un objeto perteneciente a una subclase como si fuera un objeto de su clase superior. Esta capacidad permite una aproximación más generalizada y flexible en la escritura de código, donde objetos de distintas clases pueden ser tratados de manera uniforme en contextos específicos

Ejemplos de código:

Herencia:

Creación de una clase base y clases derivadas que demuestran el uso de herencia.

```
// Clase base
class Figura {
   int area() {
      return 0;
   }
}

// Clase derivada
class Rectangulo extends Figura {
   int base, altura;
   Rectangulo(int b, int a) {
      base = b;
      altura = a;
   }
   int area() {
      return base * altura;
   }
}
```

Polimorfismo de Inclusión:

Figura f = new Rectangulo(5, 10); // Un objeto de la clase derivada se trata como un objeto de la clase base

Sobrecarga (Overloading):

class Operaciones {

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

Polimorfismo Paramétrico (Generics):

```
class MiColeccion<T> {
    private T[] arreglo;
    MiColeccion(T[] arreglo) {
        this.arreglo = arreglo;
    }
    T obtenerElemento(int indice) {
        return arreglo[indice];
    }
}
```

Polimorfismo (en general):

```
interface Animal {
    void hacerSonido();
}

class Perro implements Animal {
    public void hacerSonido() {
        System.out.println("Guau");
    }
}

class Leon implements Animal {
```

```
    public void hacerSonido() {
        System.out.println("Rugido");
    }
```

Análisis Comparativo:

- **Diferencias entre Polimorfismo y Sobrecarga de Métodos:**

El polimorfismo permite que objetos de diferentes clases respondan de manera distinta a un mismo mensaje, mientras que la sobrecarga de métodos permite definir múltiples métodos con el mismo nombre pero con diferentes parámetros en una misma clase.

- **Diferenciación entre Sobrecarga y Redefinición de Métodos:**

La sobrecarga de métodos ocurre dentro de una misma clase, donde se definen múltiples métodos con el mismo nombre pero con diferentes parámetros. La redefinición de métodos, o overriding, ocurre en una relación de herencia, donde una subclase redefine un método que ya existe en su superclase.

En mis propias palabras:

- ¿Qué es el término firma?

La "firma" de un método es como su id de presentación, tiene el nombre del método y qué tipo de datos puede aceptar, básicamente es como la manera en que el método se identifica entre otros métodos Por ejemplo, si tienes dos métodos "sumar", uno que suma números enteros y otro que suma números decimales, la firma de cada método sería diferente porque uno acepta números enteros y el otro acepta números decimales.

¿Diferencias entre los términos Overloading y Overriding?-

La sobrecarga significa tener varios métodos con el mismo nombre pero con diferentes formas de usarlos dentro de una misma clase, Por ejemplo, con el mismo metodo anterior, “sumar”, que puede sumar tanto números enteros como decimales

En cambio, la redefinición sucede cuando tienes una clase que hereda de otra y quieres cambiar cómo funciona un método que ya existe en la clase padre, entonces, en la clase hija, puedes escribir el mismo método con el mismo nombre, pero puedes hacer que haga algo diferente.

¿Se pueden sobrecargar métodos estáticos?-

Efectivamente mi querido wattson, en Java, puedes tener varios métodos con el mismo nombre pero que aceptan diferentes tipos de datos como entrada, incluso si son métodos que no necesitan un objeto específico para ser llamados (que serian los métodos estáticos) esto quiere decir que puedes tener diferentes versiones de un mismo método con diferentes maneras de usarlo, incluso si son métodos estáticos. Volvemos al mismo ejemplo del metodo sumar, que puede sumar números enteros en una versión y sumar números decimales en otra versión, y ambos pueden ser métodos estáticos.

¿Es posible sobrecargar la clase main() en Java

Nope, Java espera que este método tenga un formato específico: debe llamarse main, ser público, estático y no devolver ningún valor (osea void).aparte, debe tener un array de cadenas como referencia, que se usa para pasar argumentos de línea de comandos al programa.