SOBRECARGA DE CONSTRUCTORES

Algunas veces hay una necesidad de inicializar un objeto de diferentes maneras. Esto se puede hacer usando la sobrecarga de constructor. Hacerlo le permite construir objetos de varias maneras.

Tomemos un ejemplo para comprender la necesidad de sobrecargar constructores. Considere el siguiente programa:

```
//Demostración de Sobrecarga de constructores

class MiClase{

    int x;


    MiClase(){

        System.out.println("Dentro de MiClase().");
```

```
        x=0;
    }

    MiClase(int i){
        System.out.println("Dentro de MiClase(int).");
        x=i;
    }

    MiClase(double d){
        System.out.println("Dentro de MiClase(double).");
        x=(int)d;
    }

    MiClase(int i, int j){
        System.out.println("Dentro de MiClase(int, int).");
        x=i*j;
    }
}

class DemoSobrecargaConstructor{
    public static void main(String[] args) {
        MiClase t1=new MiClase();
        MiClase t2=new MiClase(28);
        MiClase t3=new MiClase(15.23);
```

```
        MiClase t4=new MiClase(2,4);


        System.out.println("t1.x: "+ t1.x);
        System.out.println("t2.x: "+ t2.x);
        System.out.println("t3.x: "+ t3.x);
        System.out.println("t4.x: "+ t4.x);
    }
}
```

Salida:

```
Dentro de MiClase().
Dentro de MiClase(int).
Dentro de MiClase(double).
Dentro de MiClase(int, int).
t1.x: 0
t2.x: 28
t3.x: 15
t4.x: 8
```

MiClase() está sobrecargado de cuatro maneras, cada una construyendo un objeto de manera diferente. El constructor apropiado se llama en función de los parámetros especificados cuando se ejecuta . Al sobrecargar el constructor de una clase, le da flexibilidad al usuario de su clase en la forma en que se construyen los objetos.

Objeto inicializa otro objeto

Una de las razones más comunes por las que los constructores están sobrecargados es permitir que un objeto inicialice otro. Por ejemplo, considere

este programa que usa la clase Summation para calcular la suma de un valor entero:

```
//Inicializar un objeto con otro

class Suma{
    int sum;

    //Constructor desde un int
    Suma(int num){
        sum=0;
        for (int i=1;i<=num;i++)
            sum+=i;
    }

    //Constructor desde otro objeto
    Suma (Suma obj){
        sum=obj.sum;
    }
}

class DemoSobrecargaConstructor{
    public static void main(String[] args) {
        Suma s1=new Suma(5);
        Suma s2=new Suma(s1);
```

```
        System.out.println("s1.sum: "+s1.sum);

        System.out.println("s2.sum: "+s2.sum);

    }

}
```

Salida:

```
s1.sum: 15

s2.sum: 15
```

A menudo, como muestra este ejemplo, una ventaja de proporcionar un constructor que usa un objeto para inicializar otro es la eficiencia. En este caso, cuando se construye s2, no es necesario recalcular la suma.

Por supuesto, incluso en los casos en que la eficiencia no es un problema, a menudo es útil proporcionar un constructor que haga una copia de un objeto.

Usando this() en sobrecarga de constructores

La referencia this() se puede utilizar durante la sobrecarga del constructor para llamar implícitamente al constructor predeterminado desde el constructor parametrizado. Tenga en cuenta que debería ser la primera declaración dentro de un constructor.

```
//Programa Java para ilustrar el rol de this()

// en la sobrecarga del constructor


class DemoSCT

{

    double largo, ancho, alto;
```

```
    int numero;


    // Constructor utilizado cuando todas las dimensiones
    // se especifican
    DemoSCT(double w, double h, double d, int num)
    {
        largo = w;
        ancho = h;
        alto = d;
        numero = num;
    }


    // Constructor utilizado cuando no se especificaron
dimensiones
    DemoSCT()
    {
        // Vacio
        largo = ancho = alto = 0;
    }


    // Constructor utilizado cuando solo se especifica
numero
    DemoSCT(int num)
    {
```

```
        // this() se utiliza para llamar al constructor
predeterminado
        // desde el constructor con parámetros
        this();
        numero = num;
    }

    public static void main(String[] args)
    {
        // crear DemoSCT usando solo numero
        DemoSCT DemoSCT1 = new DemoSCT(5);

        // obteniendo el valor inicial de largo en DemoSCT1
        System.out.println(DemoSCT1.largo);
    }
}
```

Salida:

```
0.0
```

Como podemos ver en el programa anterior, llamamos al constructor DemoSCT(int num) durante la creación del objeto utilizando solo el número. Al usar la instrucción this() dentro de ella, el constructor predeterminado DemoSCT() es invocado implícitamente desde allí.

×Nota: La llamada del constructor debe ser la primera declaración en el cuerpo del constructor. Por ejemplo, el siguiente fragmento no es válido y arroja un error de tiempo de compilación.

```
DemoSCT(int num)
  {
     numero = num;
     /* La llamada de constructor debe ser la primera
     instrucción en un constructor */
     this();  /*ERROR*/

  }
```

Puntos importantes que deben tenerse en cuenta al hacer sobrecarga de constructores:

- La llamada del constructor con this(), debe ser la primera declaración del constructor en Java.
- Si hemos definido cualquier constructor parametrizado, el compilador no creará el constructor predeterminado. Y viceversa, si no definimos ningún constructor, el compilador crea el constructor predeterminado (también conocido como constructor sin-argumentos) de forma predeterminada durante la compilación
- La invocación recursiva al constructor no es válida en java.