Área Académica Computación y Electrónica
Licenciatura en Ciencias Computacionales
Programación Orientada a Objetos

Tema: Sobrecarga de Constructores

Elaboración: M. en C. Iliana Castillo Pérez

Fecha de elaboración: febrero/2019

Tema

Sobrecarga de Constructores

Resumen

En este trabajo se presenta como utilizar diversos constructores para construir objetos, pudiendo todos ellos incluirse al mismo tiempo en la misma clase.
Además, llamada a constructores dentro de la misma clase y a constructores de una clase superior desde una clase hija.

Palabras Clave: constructores, inicialización de objetos, creación de objetos.

Tema

Constructor Overloading

Abstract

This paper presents how to use different constructors to build objects, all of which can be included in the same class at the same time.

In addition, we call constructors within the same class and constructors of a higher class from a daughter class.

Keywords: constructors, object initialization, object creation.

Sobrecarga de Constructores

Al momento de instanciar un objeto, el lenguaje puede ser capaz de suministrar varios constructores basándose en los datos del objeto que se está creando.

A colocar varios constructores con diversidad de parámetros en una misma clase, para crear e inicializar objetos de ese tipo de clase, se le llama sobrecarga de constructores.

Ejemplo

En este ejemplo se crean objetos de tipo Empleado con atributos como nombre, salario y fecha de nacimiento. Al implementarlo se puede dar opciones al usuario para construir los objetos de tipo Empleado aún cuando no se conozcan todos los valores de sus atributos:

```
1 public class Empleado {
2   private final double SALARIO_BASE = 15000.00;
3   private String nombre;
4   private double salario;
5   private Fecha fechaNacimiento;
6
7   public Empleado(String nombre, double salario, Fecha fNac){
8     this.nombre = nombre;
9     this.salario = salario;
10    this.fechaNacimiento = fNac;
11  }
```

Ejemplo

```
12  public Empleado(String nombre, double salario) {
13    this(nombre, salario, null);
14  }
15
16  public Empleado(String nombre, Fecha fNac) {
17    this(nombre, SALARIO_BASE, fNac);
18  }
19
20  public Empleado(String nombre) {
21    this(nombre, SALARIO_BASE);
22  }
23 //más código…
24 }
```

En el código del ejemplo anterior, la palabra reservada *this* siempre debe ser la primera línea dentro del constructor. Puede haber más código de inicialización después de la llamada *this*, pero no antes.

Descripción del ejemplo

El código del ejemplo anterior muestra cuatro constructores sobrecargados para la clase Empleado. El primer constructor (líneas 7-11) inicializa todas la variables de instancia. El segundo (líneas 12-14) no contiene la fecha de nacimiento. La referencia this se utiliza como forma de envío de llamada a otro constructor (siempre dentro de la misma clase), en este ejemplo, el primer constructor. El tercer constructor (líneas 15-17) llama al primer constructor pasando la constante de clase SALARIO_BASE. El cuarto constructor (líneas 18-20) llama al segundo constructor pasando SALARIO_BASE, que, a su vez, llama al primer constructor pasando null en lugar de la fecha de nacimiento.

Los constructores NO se heredan

Aunque las subclases heredan todos los métodos y variables de sus superclases, no heredan sus constructores.

Las clases sólo pueden obtener un constructor de dos maneras: debe escribirlo el programador o, si éste no lo escribe, debe usar el constructor predeterminado. Siempre se llama al constructor de nivel superior además de llamar al constructor subordinado.

Llamada a constructores de una clase de nivel superior

Al igual que los métodos, los constructores pueden llamar a los constructores no privados de las superclases.

Puede llamar a un determinado constructor de una superclase como parte de la inicialización de una subclase utilizando la palabra reservada **super** en la *primera línea* del constructor de la subclase.

Para controlar la llamada de ese constructor concreto, es preciso suministrar los argumentos adecuados a **super( )**.

Llamada a constructores de una clase de nivel superior

Cuando no hay ninguna llamada a **super** con argumentos, se llama *por default* al constructor de la superclase que tenga cero argumentos.

La llamada a **super( )** puede incluir cualquier cantidad de argumentos adecuados para los distintos constructores disponibles en la clase superior, pero *debe ser la primera sentencia* del constructor.

Ejemplo

```
1 public class Gerente extends Empleado {
2   private String departamento;
3
4   public Gerente(String nombre, double salario, String depto){
5     super(nombre, salario);
6     departamento = depto;
7   }
8
9    public Gerente(String nombre, String departamento) {
10     super(nombre);
11     this.departamento = departamento;
12   }

13   public Gerente(String departamento) {
14     this.departamento = departamento;
15   }
16 }
```

Quick Recap

- Every class has a constructor whether it's a normal class or an abstract class.
- Constructors are not methods and they don't have any return type.
- Constructor name should match with class name .
- Constructor can use any access specifier, they can be declared as private also. Private constructors are possible in java but there scope is within the class only.
- **Like constructors method can also have name same as class name, but still they have return type, though which we can identify them that they are methods not constructors.**
- If you don't implement any constructor within the class, compiler will do it for.
- **this() and super() should be the first statement in the constructor code.** If you don't mention them, compiler does it for you accordingly.

Quick Recap

- Constructor overloading is possible but overriding is not possible. Which means we can have overloaded constructor in our class but we can't override a constructor.
- Constructors can not be inherited.
- If Super class doesn't have a no-arg (default) constructor then compiler would not insert a default constructor in child class as it does in normal scenario.
- Interfaces **do not have constructors**.
- Abstract class can have constructor and it gets invoked when a class, which implements interface, is instantiated. (i.e. object creation of concrete class).
- A constructor can also invoke another constructor of the same class – By using this(). If you want to invoke a parameterized constructor then do it like this: **this(parameter list)**.

Información tomada de (BeginnersBook,2019)

Referencias

Sun Microsystems (2008). Manual de Lenguaje de programación Java SL-275-SE6, Broomfield, EE.UU.

BeginnersBook. (2019). Constructors in Java – A complete study!! Obtenido de BeginnersBook: https://beginnersbook.com/2013/03/constructors-in-java/

Datos de contacto

M.C.C. Iliana Castillo Pérez
Profesora-Investigadora
Área Académica de Computación y Electrónica
Instituto de Ciencias Básicas e Ingeniería
Universidad Autónoma del Estado de Hidalgo
Correo-e: ilianac@uaeh.edu.mx
Teléfono: (01) 771 7172000 ext. 6734