Contenido

1. EL CONSTRUCTOR DE INSTANCIAS 3

1.1. LLAMADAS ENCADENADAS.....4

2. EL CONSTRUCTOR ESTÁTICO 5

3. OTRAS CONSIDERACIONES..... 6

1. EL CONSTRUCTOR DE INSTANCIAS

Para **crear una instancia** de una clase y generar de ese modo un objeto, es necesario invocar al **método constructor** de la clase. El constructor es un método que poseen todas las clases y que permite construir un objeto, ubicándolo en memoria y devolviendo una referencia al mismo.

Además, el constructor inicializa y configura los miembros del nuevo objeto, de manera que, antes de invocar a un método, el estado del objeto esté preparado para la ejecución de sus métodos sin generar excepciones.

Cualquier clase a la que no se le ha proporcionado explícitamente un constructor dispone de un **constructor por defecto**, implícito (no es necesario escribirlo en el código fuente), con accesibilidad pública y que no recibe parámetros:

```
public class Test
   {
   // Si esta clase carece de constructores, el compilador define un
   // constructor por defecto. A todos los efectos, aunque el programador no
   // lo incluya en el código, existe el siguiente constructor:

   // constructor 'fantasma'
   public Test()
      { }
   }
```

Si se añade un constructor que reciba parámetros, el constructor por defecto queda invalidado y, en caso de necesitar ambos, hay que implementar el constructor por defecto explícitamente.

En C#, la sintaxis del constructor difiere ligeramente de la sintaxis de un método convencional: se define con el **mismo identificador de la clase**, sin especificar un tipo de retorno, como en el ejemplo siguiente:

```
public class Test
   {
   // constructor sin parámetros
   public Test()
      {
      // [...]
      }
   }
```

1.1. LLAMADAS ENCADENADAS

Cuando una clase dispone de varios constructores, suele ser habitual que uno de los constructores actúe realizando una inicialización base y los restantes constructores amplifican ese comportamiento con tareas de inicialización adicionales.

En estos casos, resulta conveniente que un constructor pueda invocar a otro, de manera que no sea necesario repetir el mismo código en todos los constructores. A esta llamada de un constructor desde otro se le denomina encadenamiento de constructores.

En Java, un constructor puede llamar a otro utilizando la auto-referencia `this`[1], como en el siguiente ejemplo:

```
public class Test
   {
   // Constructor sin parámetros
   public Test()
      {
      // [...]
      }

   // Llama al constructor sin parámetros y después
   // ejecuta su propia implementación
   public Test(string param1)
      {
      this();
      // [...]
      }

   // Llama al constructor con parámetro y después
   // ejecuta su propia implementación
   public Test(string param1, string param2)
      {
      this(param1);
      // [...]
      }
   }
```

- Al encadenar llamadas a un constructor es obligatorio que la llamada a `this` sea la primera instrucción del constructor.

[1] Recuerda que hemos abordado la auto-referencia `this` al estudiar los miembros de una clase.

2. EL CONSTRUCTOR ESTÁTICO

Aunque aún no hemos abordado el estudio de los elementos estáticos en profundidad, ya hemos introducido en documentos previos el concepto de miembro estático: un miembro que no es construido en los objetos durante la instanciación, sino que queda definido en la clase, existiendo un único valor de ese miembro, al que se accede a través del identificador de la clase.

Es habitual que en una clase se encuentren definidos campos y propiedades estáticas. Para inicializar estos elementos estáticos con unos valores determinados, *puede* utilizarse un constructor especial en la clase (un ***constructor estático***), que se ocupa de dar un valor inicial a los miembros estáticos, antes de que la clase pueda ser invocada.

Abordaremos las características y el uso del constructor estático en el documento dedicado a miembros estáticos.

3. OTRAS CONSIDERACIONES

- A menudo encontrarás situaciones en las que los métodos de una clase requieren que el estado interno de la clase se ajuste a determinados requerimientos. Si la instancia no cumple con ese estado interno requerido, la ejecución del método generará excepciones.

- Por ejemplo, puede ocurrir que un método requiera que una propiedad de la instancia no sea nula. Si, nada más construir el objeto, la propiedad es nula y se invoca al método, se emitirá una excepción.

- En estos casos, nunca se debe confiar en que el código cliente inicializará la instancia con el estado adecuado antes de invocar al método, ni siquiera si dicho comportamiento ha sido documentado.

- Por el contrario, en estos casos, se debe garantizar una invocación segura de los métodos. Una opción es utilizar un constructor que obligue a proporcionar el estado interno necesario para que la ejecución de los métodos sea consistente.

- Como norma de diseño, se debe evitar lanzar excepciones desde el cuerpo de un constructor, especialmente si la clase mantiene recursos reservados que deben ser liberados.

- Emitir una excepción desde el constructor deja a la instancia en un estado inconsistente, ya que, hasta que el constructor no finaliza, no se tienen garantías de que la instancia ha sido debidamente construida y su estado interno ha quedado definido.