Índice de contenido

1.1. Definición de una clase POJO

Para poder declarar e instanciar un objeto es necesario definir nuestra clase.

El proceso de definición de la clase incluye:

- Definir el nombre de la clase.
- Definir los atributos de la clase.
- Definir método Constructor de la clase.
- Definir métodos Getters y Setters de los atributos (según corresponda).
- Definir método toString (opcional).

Para nuestra clase definiremos una clase POJO llamada Productos. Estos productos tendrán los siguientes atributos: ID, Detalle, Precio Unitario, Cantidad. Se deberán crear métodos de acceso y asignación para todos los atributos y un método para mostrar la información del producto por consola.

Siguiendo estos lineamientos nuestra clase POJO Productos quedaría de la siguiente manera:

```java
package com.codewithomar.clase8.producto;

//Clase POJO que almacena la info de un Producto
public class Producto {
    //Declaración de atributos
    private long id;
    private String detalle;
    private double precioUnitario;
    private int cantidad;

    //Método Constructor vacío
    public Producto() {
    }

    //Método Constructor con parámetros
    public Producto(long id, String detalle,
                    double precioUnitario, int cantidad){
        this.id = id;
        this.detalle = detalle;
        this.precioUnitario = precioUnitario;
        this.cantidad = cantidad;
    }

    //Método Getter atributo id
    public long getId() {
        return id;
    }
```

```
28
29      //Método Setter atributo id
30      public void setId(long id) {
31          this.id = id;
32      }
33
34      //Método Getter atributo detalle
35      public String getDetalle() {
36          return detalle;
37      }
38
39      //Método Setter atributo detalle
40      public void setDetalle(String detalle) {
41          this.detalle = detalle;
42      }
43
44      //Método Getter atributo precioUnitario
45      public double getPrecioUnitario() {
46          return precioUnitario;
47      }
48
49      //Método Setter atributo precioUnitario
50      public void setPrecioUnitario(double precioUnitario) {
51          this.precioUnitario = precioUnitario;
52      }
53
54      //Método Getter atributo cantidad
55      public int getCantidad() {
56          return cantidad;
57      }
58
59      //Método Setter atributo cantidad
60      public void setCantidad(int cantidad) {
61          this.cantidad = cantidad;
62      }
63
64      /* Método toString para mostrar la info
65       * del producto por consola */
66      @Override
67      public String toString() {
68          return "Producto{" +
69                  "id=" + id +
70                  ", detalle='" + detalle + '\'' +
```

```
                ", precioUnitario=" + precioUnitario +
                ", cantidad=" + cantidad +
                '}';
    }
}
```

1.2. Definición de una clase donde declararemos el objeto

Ya tenemos nuestra clase POJO definida con todos sus atributos y métodos necesarios. Ahora, **para poder declarar un objeto de la clase Producto, es necesario definir una segunda clase en donde realmente se declarará ese objeto para darle uso.**

En este caso como es un programa sencillo, usaremos la clase Main junto con el método main para declarar, instanciar y utilizar nuestro objeto.

```
package com.codewithomar.clase8.main;

/* Clase para declarar, instanciar y utilizar objetos
 * de tipo Producto */
public class Main {
    public static void main(String[] args) {

    }
}
```

2. Declaración de un objeto

Una vez tenemos nuestra clase POJO y la clase donde declarar el objeto, pasamos a declarar el objeto.

La declaración de un objeto de una clase podría entenderse de la misma manera en la que declaramos atributos/variable de tipos de datos primitivos o Wrapper.

Por ejemplo:

- Si necesitamos un atributo numérico entero podríamos utilizar int, long o sus clases Wrapper Int, Long.
- Si necesitamos un atributo numérico decimal podríamos utilizar double o su clase Wrapper Long.
- Si necesitamos una cadena utilizamos String.

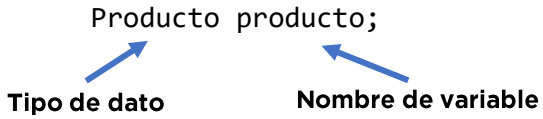
La sintaxis para declarar un atributo es la siguiente:

```
Tipo_de_dato nombre_variable;
```

Para declarar un objeto, seguimos la misma sintaxis:

- El tipo de dato será **el nombre de la clase** de la que queremos crear el objeto.
- El nombre del atributo puede ser cualquier nombre siempre y cuando se sigan los lineamientos de nombramiento de variables en Java.
    - SIN EMBARGO, **las buenas practicas nos indican que el nombre del objeto tendrá el mismo nombre de la clase o un acrónimo de este en el caso de que el nombre de la clase sea muy largo**.

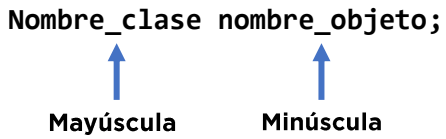
Por ejemplo:



- En donde Producto será nuestro tipo de dato (escrito en mayúscula).
- En donde producto será el nombre de la variable (escrito en minúscula).

Recordemos que Java es "case sensitive". Lo que significa que tiene importancia si una palabra esta escrita en Mayúscula o Minúscula. Por lo que, para Java, Producto (con P mayúscula) y producto (con P minúscula) son dos palabras distintas y no significan lo mismo.

De esta manera, tenemos que la **sintaxis para declarar un objeto de una clase es**:



Aplicando la sintaxis en nuestro proyecto, nuestro método main quedaría de la siguiente manera:

```java
package com.codewithomar.clase8.main;

import com.codewithomar.clase8.producto.Producto;

/* Clase para declarar, instanciar y utilizar objetos
 * de tipo Producto */
public class Main {
    public static void main(String[] args) {
        //Declaración de objeto de tipo Producto
        Producto producto;
    }
}
```

En la línea 10, podemos notar la declaración del objeto llamado `producto` del tipo de dato `Producto`.

En la línea 3, tenemos la declaración del import ya que la clase `Main` y la clase `Producto` se encuentra en dos paquetes distintos.

3. Instanciación de un objeto

Cuando declaramos un objeto simplemente le especificamos a Java el tipo de objeto que utilizaremos y el nombre de ese objeto. Sin embargo, esto no es suficiente para poder realmente utilizar ese objeto.

Para poder utilizar este objeto necesitamos que Java realice dos procesos:

1. **Guardar memoria para el objeto.**
   - Esto se logra ejecutando el operador **new**.
2. **Inicializar sus atributos.**
   - Esto se logra ejecutando el **método constructor** de la clase.

La sintaxis para que estos dos procesos se ejecuten y se instancie un objeto es la siguiente:

nombre_objeto = new metodo_constructor_objeto();

- **Operador new**
  - El operador new solicita memoria del sistema para almacenar una nueva instancia del objeto que se está creando.
  - Una vez ejecutado el método constructor, el operador new devuelve una referencia de la nueva instancia creada que se almacenará en el nombre del objeto.
- **Método Constructor**
  - Es el encargado de inicializar los atributos de la clase y realizar cualquier otra tarea de inicialización necesaria.
  - Dependiendo de cual método constructor utilicemos, los atributos de nuestros objetos tendrán valores iniciales o no.

3.1. Instanciación de un objeto utilizando el constructor por defecto.

Sin el método constructor, la instanciación no podría ser ejecutada. Por lo tanto, es necesario que todas las clases tengan mínimamente un método constructor.

En el caso de que la clase no tenga declarado ningún método constructor, Java proveerá uno en tiempos de ejecución.

El método constructor automático que Java proveerá tendrá la siguiente sintaxis:

public nombre_clase() { }

Es un constructor sin parámetros y sin ningún código en su contenido. Solo contiene la sintaxis para que no ocurra ningún error de compilación a la hora de instanciar un objeto.

Por buenas prácticas, se recomienda declarar explícitamente el método constructor vacío en nuestras clases.

En nuestro clase Producto tenemos declarado el siguiente constructor:

```java
package com.codewithomar.clase8.producto;

//Clase POJO que almacena la info de un Producto
public class Producto {
    //Declaración de atributos
    private long id;
    private String detalle;
    private double precioUnitario;
    private int cantidad;

    //Método Constructor vacío
    public Producto() {
    }

    [...]
}
```

Utilizaremos este constructor para instanciar nuestro primer objeto:

```java
package com.codewithomar.clase8.main;

import com.codewithomar.clase8.producto.Producto;

/* Clase para declarar, instanciar y utilizar objetos
 * de tipo Producto */
public class Main {
    public static void main(String[] args) {
        //Declaración de objeto de tipo Producto
        Producto producto;
        //Instanciación del objeto producto
        producto = new Producto();
    }
}
```

3.1.1. Objeto con atributos vacíos.

Cuando se utiliza el constructor por defecto, todos los atributos de nuestro objeto tendrán valores por defecto según el tipo de dato del atributo:

- Enteros -> 0
- Decimales -> 0.00
- String -> null

Lo podemos corroborar imprimiendo los datos de los atributos del objeto producto por consola:

```java
package com.codewithomar.clase8.main;

import com.codewithomar.clase8.producto.Producto;

/* Clase para declarar, instanciar y utilizar objetos
 * de tipo Producto */
public class Main {
    public static void main(String[] args) {
        //Declaración de objeto de tipo Producto
        Producto producto;
        //Instanciación del objeto producto
        producto = new Producto();

        //Impresión de los atributos del objeto producto
        System.out.println("Objeto producto:");
        System.out.println(producto.toString());
    }
}
```

Output por consola

```
"C:\Program Files\Java\jdk-21\bin\java.exe" [...]
Objeto producto:
Producto{id=0, detalle='null', precioUnitario=0.0, cantidad=0}

Process finished with exit code 0
```

3.1.2. Objeto con atributos no vacíos utilizando Setters.

Si bien al utilizar el constructor por defecto nuestros atributos no tendrán datos inicialmente, podemos asignarle los valores que necesitemos utilizando los métodos setter de cada atributo.

Vamos a crear un segundo objeto llamado producto2. Lo instanciaremos utilizando el constructor vacío por defecto al igual que con producto, pero le asignaremos valores a cada atributo utilizando los métodos Setters:

```java
package com.codewithomar.clase8.main;

import com.codewithomar.clase8.producto.Producto;

/* Clase para declarar, instanciar y utilizar objetos
 * de tipo Producto */
public class Main {
    public static void main(String[] args) {
        //Declaración de objeto de tipo Producto
        Producto producto;
        //Instanciación del objeto producto
        producto = new Producto();

        //Impresión de los atributos del objeto producto
        System.out.println("Objeto producto:");
        System.out.println(producto.toString());
        System.out.println(); //Salto de línea

        //Declaración de objeto de tipo Producto
        Producto producto2;
        //Instanciación del objeto producto2
        producto2 = new Producto();

        //Asignación de valores a los atributos de producto2
        producto2.setId(2);
        producto2.setDetalle("Impresora Canon PIXMA G4110");
        producto2.setPrecioUnitario(225.95);
        producto2.setCantidad(10);

        //Impresión de los atributos del objeto producto2
        System.out.println("Objeto producto2:");
        System.out.println(producto2.toString());
    }
}
```

Output por consola

```
"C:\Program Files\Java\jdk-21\bin\java.exe" [...]
Objeto producto:
Producto{id=0, detalle='null', precioUnitario=0.0, cantidad=0}

Objeto producto2:
Producto{id=2, detalle='Impresora Canon PIXMA G4110',
precioUnitario=225.95, cantidad=10}

Process finished with exit code 0
```

Podemos ver que utilizando los métodos Setter podemos asignarle valores a esos atributos vacíos que fueron inicializados por el método constructor por defecto. Esta técnica es bastante útil cuando no queremos inicializar todos los atributos del objeto y queremos inicializar solo algunos de ellos.

3.2. Instanciación de un objeto utilizando el constructor con parámetros.

Además del constructor por defecto, en nuestra clase Producto tenemos un segundo método constructor.

En Java existe un concepto llamado **sobrecarga de métodos**. Consiste en tener más de un método con exactamente el mismo nombre pero que realizan actividades distintas. Esto se logra definiendo distintos tipos y/o cantidad de parámetros en cada método.

De esta manera Java sabrá cuál método invocar en base a la cantidad y tipo de parámetros al momento en que el método fue llamado.

Para nuestro proyecto utilizaremos el segundo método constructor declarado en la clase Producto:

```java
package com.codewithomar.clase8.producto;

//Clase POJO que almacena la info de un Producto
public class Producto {
    //Declaración de atributos
    private long id;
    private String detalle;
    private double precioUnitario;
    private int cantidad;

    [...]

    //Método Constructor con parámetros
```

```
14      public Producto(long id, String detalle,
15                      double precioUnitario, int cantidad){
16          this.id = id;
17          this.detalle = detalle;
18          this.precioUnitario = precioUnitario;
19          this.cantidad = cantidad;
20      }
21
22      [...]
23  }
```

Para utilizar el constructor con parámetros debemos:

- Asignarles valores a todos los parámetros declarados en el constructor.
- No podemos omitir ningún atributo. De lo contrario ocurriría un error de compilación.
- Debemos asignarles los valores a los atributos siguiendo el mismo orden en el que están declarados en el método constructor.

Para nuestro proyecto vamos a declarar un objeto de tipo Producto llamado producto3. Lo instanciaremos utilizando el método constructor con parámetros definido en la línea 14 de la clase Producto. De esta manera le asignaremos valores iniciales a todos los atributos del objeto producto3 a partir del momento en el que el objeto es instanciado.

```
1   package com.codewithomar.clase8.main;
2
3   import com.codewithomar.clase8.producto.Producto;
4
5   /* Clase para declarar, instanciar y utilizar objetos
6    * de tipo Producto */
7   public class Main {
8       public static void main(String[] args) {
9           //Declaración de objeto de tipo Producto
10          Producto producto;
11          //Instanciación del objeto producto
12          producto = new Producto();
13
14          //Impresión de los atributos del objeto producto
15          System.out.println("Objeto producto:");
16          System.out.println(producto.toString());
17          System.out.println(); //Salto de línea
18
19          //Declaración de objeto de tipo Producto
20          Producto producto2;
21          //Instanciación del objeto producto2
22          producto2 = new Producto();
```

```
23
24          //Asignación de valores a los atributos de producto2
25          producto2.setId(2);
26          producto2.setDetalle("Impresora Canon PIXMA G4110");
27          producto2.setPrecioUnitario(225.95);
28          producto2.setCantidad(10);
29
30          //Impresión de los atributos del objeto producto2
31          System.out.println("Objeto producto2:");
32          System.out.println(producto2.toString());
33          System.out.println(); //Salto de línea
34
35          //Declaración de objeto de tipo Producto
36          Producto producto3;
37          //Instanciación del objeto producto3
38          producto3 = new Producto(3, "Silla de Oficina Herman Miller",
39                  300.00, 4);
40
41          //Impresión de los atributos del objeto producto3
42          System.out.println("Objeto producto3:");
43          System.out.println(producto3.toString());
44      }
45  }
```

Output por consola

```
"C:\Program Files\Java\jdk-21\bin\java.exe" [...]
Objeto producto:
Producto{id=0, detalle='null', precioUnitario=0.0, cantidad=0}

Objeto producto2:
Producto{id=2, detalle='Impresora Canon PIXMA G4110',
precioUnitario=225.95, cantidad=10}

Objeto producto3:
Producto{id=3, detalle='Silla de Oficina Herman Miller',
precioUnitario=300.0, cantidad=4}

Process finished with exit code 0
```

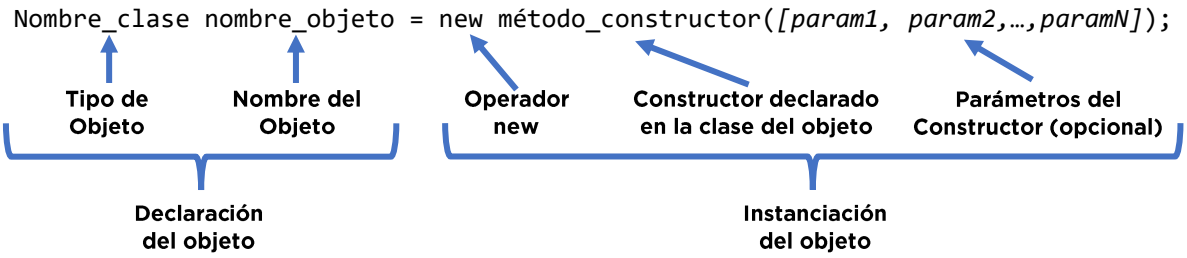
Podemos notar que, utilizando el constructor con parámetros al momento de instanciar el objeto, **podemos asignarle valores a todos los atributo del objeto sin la necesidad de asignarle valores individualmente uno por uno utilizando los métodos Setter**.

4. Declaración e instanciación de un objeto en una misma sentencia

En todos los ejemplos del punto 3, hemos primero declarado el objeto y luego lo instanciamos en un línea de código distinta.

En muchas ocasiones, esta práctica no es necesaria y podemos declarar e instanciar el objeto en una misma sentencia y línea de código.

Para lograrlo utilizaremos la siguiente sintaxis:



Podemos notar que del lado izquierdo de la expresión tenemos la declaración del objeto y del lado derecho de la expresión tenemos la instanciación del objeto.

Esto se puede utilizar tanto con cualquier método constructor ya sea vacío o con parámetros.

Para nuestro proyecto definiremos un último objeto llamado producto4. Declararemos e instanciaremos el objeto en la misma sentencia:

```
1  package com.codewithomar.clase8.main;
2
3  import com.codewithomar.clase8.producto.Producto;
4
5  /* Clase para declarar, instanciar y utilizar objetos
6   * de tipo Producto */
7  public class Main {
8      public static void main(String[] args) {
9
10         [...]
11
12         //Declaración e instanciación del objeto producto4
13         Producto producto4 = new Producto(4, "Pluma BIC Azul Punta Fina",
14                 0.25, 200);
15
16         //Impresión de los atributos del objeto producto4
17         System.out.println("Objeto producto4:");
18         System.out.println(producto4.toString());
19     }
20 }
```

Output por consola

```
"C:\Program Files\Java\jdk-21\bin\java.exe" [...]
Objeto producto:
Producto{id=0, detalle='null', precioUnitario=0.0, cantidad=0}

Objeto producto2:
Producto{id=2, detalle='Impresora Canon PIXMA G4110',
precioUnitario=225.95, cantidad=10}

Objeto producto3:
Producto{id=3, detalle='Silla de Oficina Herman Miller',
precioUnitario=300.0, cantidad=4}

Objeto producto4:
Producto{id=4, detalle='Pluma BIC Azul Punta Fina', precioUnitario=0.25,
cantidad=200}

Process finished with exit code 0
```

De esta manera podemos apreciar que obtenemos el mismo resultado en una sola sentencia que declarando e instanciando por separado.

Es importante aclarar que no hay una “mejor manera” de declarar e instanciar un objeto. Siempre dependerá de la implementación de tu código y la lógica de negocio para que el objeto se inicialice de la manera correcta.