Métodos

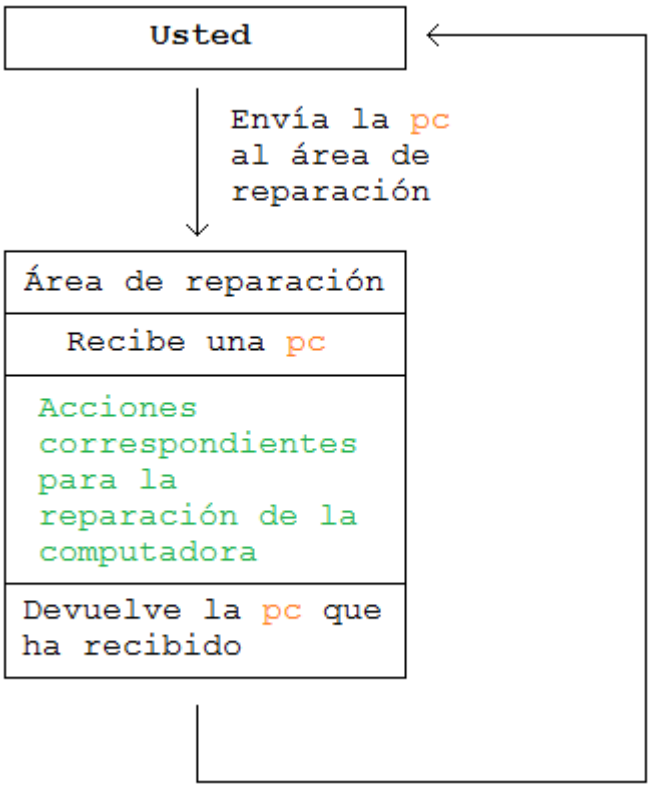
Hasta el momento, se ha hecho uso de métodos, por ejemplo cuando se pedía al usuario que ingrese un número entero por medio del método nextInt(), cuando se hacían conversiones de tipos de datos por medio de los métodos toString(), parseInt(), etc., mismo cuando se escribió código ejecutable, siempre se hizo dentro del método main.

Los métodos hacen referencia a la programación modular, un tipo de programación que es una evolución de la programación estructurada, y de la cual se hace uso en Java. La misma consiste en dividir un programa en varios módulos con el fin de que el desarrollo del programa resulte más fácil, a igual manera que la mantención del mismo, y mediante estos módulos poder hacer porciones de código reutilizables.

En Java estos módulos serán los métodos (funciones y/o procedimientos en otros lenguajes). Los mismos serán llamados o invocados desde algún otro método como puede ser el método main, y estos devolverán un resultado o ejecutarán ciertas instrucciones definidas por el programador.

Se pueden hacer varias analogías para entender cómo funcionan los métodos, por ejemplo, suponga que usted tiene una empresa de informática, la cual posee un área, sección o módulo de reparación de pc. Si se rompe una pc de la empresa, la misma será enviada a la sección de reparación, luego esta área realizará acciones para arreglar la computadora, y finalmente devolverá la computadora a usted reparada.

Si se analiza este ejemplo, puede observarse que el área de reparación recibe como parámetro de entrada una pc, y que devolverá o retornará dicha pc arreglada.



Éste método posee un parámetro de entrada y retorna algo, en este caso una pc. En otros casos puede suceder, que un método posea varios parámetros de entrada, ninguno, y que no retorne ningún valor.

Como ya se ha explicado en la unidad Nº2, cuando se define un nombre a un método, éste debe representar una acción, la primera palabra debe ir escrita en minúsculas, y luego de ésta, si existieran otras palabras, las mismas deberán comenzar con su primera letra en mayúsculas.

Para definir un método se debe indicar el tipo de valor que retornará, en el caso de que el método no retorne valor se deberá indicar mediante el tipo de dato **void**, luego se debe escribir el nombre del mismo, y entre paréntesis se deben definir los datos que recibirá el mismo como parámetros, en el caso de que el método no reciba parámetros se deben escribir los paréntesis sin nada adentro. Finalmente entre llaves se escribirá el código correspondiente al método y en el caso de devolver algún valor se lo debe indicar mediante la palabra clave **return**, en el caso de que no se retorne ningún valor, se debe obviar este último paso.

```
1  tipoDeRetorno nombreDeLaFuncion(tipo parametro1,
2                 tipo parametro2, …, tipo parmetroN) {
3
4    // Acción que realizará el método
5    return valor;  // En el caso de que el método retorne
6                   // algún valor
7
8  } // Fin del método
```

Luego de que el método se encuentre definido, se lo podrá llamar o invocar, desde cualquier otro método. Para llamar a un método se debe escribir el nombre de el mismo y entre paréntesis indicar los valores que se pasarán como parámetros al mismo, si así fuera el caso, si el método no recibe parámetros de entrada, se deberá escribir los paréntesis sin nada adentro. Para finalizar se escribe ";". En el caso de que el método retorne algún valor, deberá ser guardado en algún atributo, o bien, mostrado por consola, o utilizado para realizar alguna operación.

Ejemplo:

```
1  public class Principal {
2
3    public static void main(String[] args) {
4
5      int n1 = 5, n2 = 7, resultado = 0;
6
7      resultado = sumar (n1, n2);
8
9      mostrar(resultado);
10
11   } // Fin del método main
12
13   static int sumar (int n1, int n2){
14
15     return (n1+n2);
16
17   } // Fin del método sumar
18
19   static void mostrar (int valor){
20
21     System.out.println(valor);
22
23   } // Fin del método mostrar
24 } // Fin de la clase Principal
```

El código anterior ilustra el uso de métodos, analizando el código se tiene lo siguiente:

- **Línea 1:** Se declara la clase Principal.
- **Línea 3:** Se define el método main que tiene como parámetro de entrada un vector de tipo String.
- **Línea 5:** Se declaran los atributos de tipo entero n1, n2 y resultado, inicializándolos con los valores 5, 7 y 0 respectivamente. Tenga en cuenta que estos atributos son válidos solo dentro del método main, si se intentase usar por ejemplo, el atributo resultado en algún otro lugar que no sea dentro del método main, Java acusará error, ya que no lo encontraría.
- **Línea 7:** Se llama al método sumar, y se le pasa por parámetro, los valores de los atributos n1 y n2, es decir, se pasa por parámetro 5 y 7. Luego de que el método retorne un valor, el mismo se guardará en el atributo resultado.
- **Línea 9:** Se llama al método mostrar, al cual se le pasa por parámetro el valor del atributo resultado, es decir, el valor 12. Como el método mostrar no retorna ningún valor, no se hace ninguna operación de asignación como sucede con el método sumar.
- **Línea 13:** Se define el método sumar, el cual retorna un valor de tipo entero, y recibe por parámetro dos valores enteros. Si bien estos dos atributos enteros se llaman n1 y n2, los mismos son dos atributos diferentes a n1 y n2 definidos en el método main. Estos dos atributos tendrán validez dentro del método sumar, una vez que el mismo finaliza, los atributos se borran de memoria.
- **Línea 15:** Se retorna el valor resultante de la operación n1 + n2, es decir, se retorna el valor 12. Si bien en el ejemplo se ha escrito esta operación entre paréntesis, no es necesario indicar los paréntesis, solo se hizo para una mejor legibilidad del código.
- **Línea 19:** Se define el método mostrar, el cual no retorna ningún valor, y recibe por parámetro un valor de tipo entero.
- **Línea 21:** Se muestra por consola el valor que contiene el atributo **valor**, o sea, el valor 12.

Parámetros por valor y por referencia

Cuando se pasan parámetros a un método en Java se debe prestar atención, ya que estos pueden ser por valor o por referencia.

Parámetros por valor

Pasar un parámetro por valor implica copiar, por así decirlo, el valor existente dentro de un atributo, o bien indicar explícitamente un valor, el cual se guardará en un atributo declarado en la definición de un método, y que servirá, o podrá usarse desde que inicia el método hasta que finaliza el mismo. Una vez que se escribe la llave de cierre correspondiente al método, este atributo no existirá más en memoria.

A continuación se muestran varios ejemplos para ilustrar el funcionamiento de parámetros por valor.

Funcionamiento

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 10;
6
7       mostrar(valor1);
8
9     }
10
11    static void mostrar (int valor1){
12
13      System.out.println(valor1);
14
15    }
16
17  } // Fin de la clase Principal
```

Se copia el contenido del atributo valor1 correspondiente al método main, en el atributo valor1 correspondiente al método mostrar

Diferencia entre atributos de métodos

Ejemplo 1:

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 10;
6
7       mostrar(valor1);
8
9     }
10
11    static void mostrar (int valor1){
12
13      System.out.println(valor1);
14
15    }
16
17  } // Fin de la clase Principal
```

El atributo valor1 correspondiente al método main, será valido dentro de las llaves que inician y finalizan dicho método. Lo que se pasa como parámetro al método mostrar es el valor que contiene éste atributo y no el atributo en sí.

El atributo valor1 correspondiente al método mostrar, no tiene nada que ver con el atributo valor1 correspondiente al método main. Este será válido dentro del mismo método, y en ningún otro lugar. El valor de dicho atributo será el que se le indique como parámetro a la hora de llamar al método.

Ejemplo 2:

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 10;
6
7       mostrar(valor1);
8
9       System.out.println(valor1);
10    }
11
12    static void mostrar (int valor1){
13
14      valor1 = 300;
15
16      System.out.println(valor1);
17
18    }
19
20  } // Fin de la clase Principal
```

Se define el atributo valor1 correspondiente al método main y se le asigna el valor 10

Se pasa por parámetro el valor contenido dentro del atributo valor1 al método mostrar

Mostrará 10

Recibe el valor 10 y se guarda en el atributo valor1 correspondiente al método mostrar

Se modifica el atributo valor1 correspondiente al método mostrar

Mostrará 300

Ejemplo 3:

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 10;
6
7       mostrar(valor1);
8
9     }
10
11    static void mostrar (int v){
12
13      System.out.println(v);
14
15    }
16
17  } // Fin de la clase Principal
```

El mismo programa se puede escribir de esta otra manera, y aquí, se ve más la diferencia entre un atributo y otro, dando a entender que son dos atributos diferentes.

Ejemplo 4:

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 10;
6
7       mostrar(valor1);
8
9       System.out.println(v1);
10    }
11
12    static void mostrar (int v1){
13
14      v1 = 300;
15
16      System.out.println(valor1);
17
18    }
19
20  } // Fin de la clase Principal
```

Error: Se está intentando mostrar un atributo v1 que no existe dentro del método main

Error: Se está intentando mostrar un atributo valor1 que no existe dentro del método mostrar

Parámetros por referencia

Existe otro tipo de parámetros llamados por referencia. A grandes rasgos, al pasar un parámetro por valor lo que realmente se pasa es, o el contenido de un atributo, o el valor en sí, y no el atributo propiamente dicho. En este sentido, tomando los ejemplos anteriores, si se modifica el parámetro del método mostrar, tendrá efecto en el atributo del mismo método, pero el atributo que se pasó como parámetro por valor correspondiente al método main continuará con el mismo valor que poseía desde un principio. Al hablar de parámetros por referencia si se modificara el atributo correspondiente al método mostrar, también modificará el valor del atributo que se ha pasado como parámetro correspondiente al método main, aunque en la práctica no sea así, ya que los tipos de datos primitivos siempre se pasarán como parámetros por valor y no por referencia. Esto se debe a que cuando se pasa un parámetro por referencia, lo que se pasa no es el valor del atributo sino la dirección de memoria a la que apunta el mismo.

Ejemplo:

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int valor1 = 5;  <-- El valor 5 se guarda en la posición de memoria 0x01100101
6
7       mostrar(valor1); <-- Se pasa por parámetro la dirección de memoria 0x01100101
8
9       System.out.println(valor1); <-- Se muestra el contenido de la posición de memoria 0x01100101 que ahora es 15
10
11    } // Fin del método main
12
13    static void mostrar (int v1){ <-- El atributo v1 del método mostrar apunta a la dirección de memoria 0x01100101
14
15      System.out.println(v1); <-- Se muestra el contenido de la dirección de memoria 0x01100101 que es 5
16
17      v1 += 10; <-- Se le suma el valor 10 al contenido de la posición de memoria 0x01100101, por lo tanto dicha dirección de memoria contendrá ahora el valor 15
18
19    }
20
21  } // Fin de la clase Principal
```

El ejemplo mostrado, es a modo ilustrativo, solo para entender el concepto de cómo funcionan los parámetros por referencia, ya que como ya se ha mencionado, los tipos primitivos siempre se pasan por valor.

En java los parámetros que se pasan por referencia son los objetos. Por lo tanto todo objeto que se pase por parámetro a un método, al ser modificado dentro del mismo, se verá afectado en todas sus ocurrencias dentro del programa.

Un ejemplo sencillo para poder apreciar esto, es el paso de vectores por parámetro a un método, ya que como se ha visto, java trata a los vectores como objetos.

```
1   public class Principal {
2
3     public static void main(String[] args) {
4
5       int[] v = {1,3,5,7};
6
7       System.out.println(v[2]);  // Mostrará el valor 5
8
9       modificar(v);
10
11      System.out.println(v[2]);  // Mostrará el valor 38
12
13    } // Fin del método main
14
15    static int modificar (int[] vec){
16
17      vec[2] = 38;
18
19    } // Fin del método modificar
20
21  } // Fin de la clase Principal
```