Funciones en Java

Programación I - UNGS

Variables

Para almacenar valores utilizamos variables, que se declaran con un tipo de datos asociado:

Variables

Para almacenar valores utilizamos variables, que se declaran con un tipo de datos asociado:

```
1  public class Ejemplo {
2
3      public static void main (String[] args) {
4          String saludo = "Hola, mundo";
5          System.out.println(saludo);
6      }
7
8  }
```

Tipos de datos

- Un **tipo de datos** es ...
  1. ... un conjunto de valores (llamado el *conjunto base* del tipo),
  2. ... junto con una serie de operaciones para trabajar con los elementos de ese conjunto.

Tipos de datos

- Un **tipo de datos** es ...
  1. ... un conjunto de valores (llamado el *conjunto base* del tipo),
  2. ... junto con una serie de operaciones para trabajar con los elementos de ese conjunto.

- Por ejemplo, **int** es un tipo de datos.
  1. El conjunto base de **int** es $C = \{-2^{31}, \ldots, 2^{31} - 1\} \subseteq \mathbb{Z}$.
  2. Las operaciones son la suma (+), resta (-), producto (*), división (/), resto (%), etc.

Concordancia de tipos

En Java, para asignar un valor a una variable, es obligatorio que coincidan con su tipo. Se dice que Java es un lenguaje fuertemente tipado (strongly typed).

Concordancia de tipos

En Java, para asignar un valor a una variable, es obligatorio que coincidan con su tipo. Se dice que Java es un lenguaje fuertemente tipado (strongly typed).

```
1  public class Ejemplo {
2
3    public static void main (String[] args) {
4      String saludo = 5; // MAL! 5 es un int.
5    }
6
7  }
```

Declaración y Asignación

- **Declaración:** Creación de la variable, con su tipo de datos.
- **Asignación:** Asociación de un valor a la variable, que no cambia a menos que sea explícitamente modificado por otra asignación.
- **Inicialización:** La primera asignación a una variable.

```
1  public class Ejemplo {
2    public static void main (String[] args) {
3        String saludo = "Hola"; // Declaración más inicialización
4        saludo = saludo + ", mundo"; // Asignación
5    }
6  }
```

Asignación y expresiones

- Una variable no se puede leer/usar si no fue inicializada

```
1 String nombre;
2 System.out.println(nombre); // Error!
```

Asignación y expresiones

- Una variable no se puede leer/usar si no fue inicializada

```
1 String nombre;
2 System.out.println(nombre); // Error!
```

- El elemento del lado derecho de una asignación es una expresión del mismo tipo de la variable:

```
1 int suma = 2 + 3; // Ok
2 int suma = "Hola " + "mundo!"; // Error!
```

Asignación y expresiones

- Una variable no se puede leer/usar si no fue inicializada

```
1 String nombre;
2 System.out.println(nombre); // Error!
```

- El elemento del lado derecho de una asignación es una expresión del mismo tipo de la variable:

```
1 int suma = 2 + 3; // Ok
2 int suma = "Hola " + "mundo!"; // Error!
```

- Esta expresión también puede incluir llamadas a funciones:

```
1 int suma = sumar(2, 3);
2 String nombre = scan.nextLine();
```

Funciones con valores de retorno

- Para declarar nuestras propias funciones debemos especificar:
  1. Tipo de retorno
  2. Nombre
  3. Argumentos (o parámetros)

- Las funciones se declaran dentro del bloque "class", y por ahora las precederemos con el modificador **static**.

- Los argumentos se especifican separados por comas, y cada uno debe tener un tipo de datos asociado. Cuando se llama a la función, el código "llamador" debe respetar el orden y tipo de los argumentos.

- La función devuelve el valor que se especifica mediante la sentencia **return**.

Funciones con valores de retorno

Por ejemplo:

```
1  public static String saludarFijo() {
2      return "Hola Pepe";
3  }
4
5  public static void saludar(String nombre) {
6      System.out.println("Hola " + nombre);
7  }
```

Funciones con valores de retorno

Ejemplo de uso:

```
1  public class Ejemplo {
2
3      public static String pedirNombre() {
4          Scanner scan = new Scanner(System.in);
5          System.out.println("Escribí tu nombre:");
6          return scan.nextLine();
7      }
8
9      public static void main (String[] args) {
10         String saludo = "Hola ";
11         saludo = saludo + pedirNombre();
12         System.out.println(saludo);
13     }
14
15 }
```

Funciones con valores de retorno

Otro ejemplo:

```
1  public static String pedirNombre() {
2      Scanner scan = new Scanner(System.in);
3      System.out.println("Escribí tu nombre");
4      return scan.nextLine();
5  }
6
7  public static void saludar(String nombre) {
8      System.out.println("Hola " + nombre);
9  }
10
11 public static void main (String[] args) {
12     String nombre = pedirNombre();
13     saludar(nombre);
14 }
```

Composición

Podemos hacer lo mismo pasándole a **saludar** el resultado de **pedirNombre**. Esto es un ejemplo de composición.

```
1  public static String pedirNombre() {
2    Scanner scan = new Scanner(System.in);
3    System.out.println("Escribí tu nombre");
4    return scan.nextLine();
5  }
6
7  public static void saludar(String nombre) {
8    System.out.println("Hola " + nombre);
9  }
10
11 public static void main (String[] args) {
12   saludar(pedirNombre());
13 }
```

Expresiones y Métodos

En caso de que haya más de un parámetro, se separan por comas:

```
1  public class Ejemplo {
2
3    public static int suma(int a, int b) {
4      return a + b;
5    }
6
7    public static void main (String[] args) {
8      int a = suma(2, 3);
9      System.out.println("La suma de 2 más 3 es " + a);
10   }
11
12 }
```

Expresiones y Métodos

El mismo ejemplo, pero utilizando composición:

```
1  public class Ejemplo {
2
3    public static int suma(int a, int b) {
4      return a + b;
5    }
6
7    public static void main (String[] args) {
8      System.out.println("La suma de 2 más 3 es " + suma(2, 3));
9    }
10
11 }
```

En el libro...

Lo que vimos en esta clase, lo pueden encontrar en los **capítulos 3 y 5** del libro.