Taller de preparación para la OCI - PUC

Tipos de datos, asignaciones, expresiones y métodos básicos

1. Introducción

En este capítulo se tratarán tópicos básicos de todo lenguaje de programación. Estos tópicos son los tipos de datos, las asignaciones, expresiones y métodos básicos del lenguaje.

En todo lenguaje de programación existen distintos **tipos de datos** ya definidos, como por ejemplo los números enteros (`int`), los números reales (`double`), texto (`String`), los tipos *booleanos* que representan valores de verdad (`boolean`) entre varios otros. Los tipos de datos son fundamentales en todo lenguaje de programación porque de ellos depende cómo las constantes, variables y valores definidos en nuestros programas son manejados por el computador.

Las asignaciones y expresiones son los bloques básicos con los que empezamos a construir nuestros programas. Por ejemplo, las asignaciones nos permiten definir variables y definirles valores. Las expresiones nos permiten hacer interactuar nuestros valores, como por ejemplo realizar operaciones aritméticas entre números o también operaciones lógicas entre valores de verdad.

Finalmente, los métodos básicos de Java son funciones ya implementadas en este lenguaje de programación. Algunas de los métodos más importantes nos permiten pedirle valores al usuario o también notificar al usuario mediante mensajes.

2. Tipos de datos en Java

Los tipos de datos son atributos asociados a cada dato que indica al computador cómo estos deben ser administrados. En Java existen varios tipos de datos, pero destacaremos los siguientes:

1. Número entero (`int`): este tipo de dato representa a los números enteros. Ejemplos son -2, -1, 0, 1, 2. El valor mínimo que puede tomar un entero en Java es $-2,147,483,648$ y el máximo es $2,147,483,647$.
2. Número real (`double`): este tipo de dato representa a los números reales. Ejemplos son -3.14, 2.17, 0.0, 3.14, $1,003.72$.
3. Caracteres (`char`): este tipo de dato representa a los caracteres, como por ejemplo `'a'`, `'b'`, `'c'`, `'A'`, `'1'`.

   Observación 2.1. Es muy importante notar que el caracter `'a'` es distinto al caracter `'A'`. También los valores 1 y `'1'` son distintos, porque el primero es de tipo `int` y el segundo es de tipo `char`.

4. Cadenas de caracteres (`String`): las cadenas de caracteres, o *strings*, nos sirven para representar texto. Un ejemplo es `''Hola mundo!''` o también `''Esto es un texto''`.
5. Valores *booleanos* (`boolean`): los tipos *booleanos* nos sirven para representar valores que pueden ser verdaderos (`true`) o falsos (`false`).

3. Asignaciones y expresiones

3.1. Asignaciones

Parte fundamental de un lenguaje de programación son las variables. Las variables son los elementos a los que les asignaremos valores para después trabajar. Pensemos en una variable como una caja donde puedo guardar un dato. Existen distintos tamaños de cajas para distintos tipos de datos. El nombre de la variable es la etiqueta de la caja y el contenido (el dato) puede cambiar sin cambiar la etiqueta de la caja. En Java, una variable se declara señalando su tipo y su nombre. Por ejemplo:

```
1  int myNumber;
```

En el código estoy creando una nueva variable, llamada `myNumber`, que es de tipo `int` y aún no tiene ningún valor. El término técnico es que estoy instanciando una variable.
Las asignaciones nos permiten darle valor a las variables. Pensemos en una asignación como tomar una caja con su etiqueta y guardar un dato dentro de ella. Las asignaciones se realizan con `=`. En el siguiente código se muestran distintas asignaciones.

```
1  int myNumber = 0;
2  String myText = "Soy un texto";
3  myText = "Ahora soy otro texto";
4  boolean myBoolean;
5  myBoolean = true;
6  myBoolean = false;
```

La explicación del código es la siguiente:

- En la línea 1 estamos creando una nueva variable llamada `myNumber` que es de tipo entero y le asignamos el valor 0.
- En la línea 2 estamos creando una variable llamada `myText` de tipo *string* y estamos asignándole el valor ‘‘`Soy un texto`’’. Luego en la línea 3 estamos cambiando el valor de la variable por ‘‘`Ahora soy otro texto`’’.
- Cuando cambiamos el valor de una variable no hay que volver a declarar el tipo de esta. Finalmente en la línea 4 estamos definiendo una variable llamada `myBoolean` de tipo booleano y no le asignamos ningún valor. Luego en la línea 5 asignamos a la variable el valor `true`. Después cambiamos su valor a `false`.

3.2. Expresiones

Ahora detallaremos algunas expresiones básicas que permiten hacer operaciones entre variables. Existen más expresiones de las que mencionaremos, pero estas son las más importantes.

3.2.1. Expresiones aritméticas

Las expresiones aritméticas más importantes son la suma (`+`), resta (`-`), multiplicación (`*`), división (`/`) y módulo (`%`). Esta última nos permite obtener el resto de la división entera entre dos números. Veamos un código de ejemplo.

```
1  int a = 7;
2  int b = 4;
3
4  int result = a + b;
```

```
// El valor de result es 11

result = a - b;
// Ahora el valor de result es 3

result = 2 * a;
// Ahora el valor de result es 14

result = a / b;
// Ahora el valor de result es 1

result = a % b;
// ahora el valor de result es 3
```

En el programa anterior definimos dos variables `a` y `b`. Luego la variable `result` es definida en función de las otras dos variables. Es importante notar que:

- La variable `result` fue definida en una asignación como la multiplicación de `2` y `a`. Esto muestra que podemos utilizar expresiones entre variables y literales.
- La división entre dos números enteros debe ser un entero. Si deseamos obtener la división real podemos hacer lo siguiente:

  ```
  double a = 7;
  double b = 4;

  double result = a / b;
  // El valor de result es 1.75
  ```

  También es posible pedir que trate a números enteros como si fuesen de tipo `double`:

  ```
  int a = 7;
  int b = 4;

  double result = ((double) a) / ((double) b)
  // El valor de result es 1.75
  ```

- En la línea que dice `result = a% b`, la variable `result` toma el valor del resto de la divisón `7 / 4`.

3.3. Asignaciones a sí mismo

Es posible hacer asignaciones a una variable utilizando su valor actual. En programación es muy frecuente hacer cambios en una variable, haciendo incrementos o decrementos del valor que tiene almacenado. Por ejemplo, si tenemos el dato 42 en la variable `number`, podríamos hacer la asignación `number = number + 1` para incrementar en uno su valor.

```
int a = 3;
a = a + 1;
// ahora a vale 4

a = 2*a;
// ahora a vale 8
```

3.4. Comparación entre números

Ahora vamos a revisar las expresiones que nos permiten comparar números. Estas son:

- Comparar si dos variables son iguales (==).
- Comparar si una variable es menor o mayor que otra (< o >). Para menor igual o mayor igual podemos usar (<= o >=).

Veamos algunos ejemplos:

```
1  int a = 3;
2  boolean comparison = (a == 3);
3  // comparison es true
4
5  int b = 4;
6  comparison = (a == b);
7  // comparison ahora es false
8
9  comparison = (a >= 10);
10 // comparison es false, porque a no es mayor o igual a 10
```

Hay que destacar que estos operadores sirven solamente para comparar valores numéricos. Es importante notar que `comparison` es de tipo booleano (tipos que pueden ser sólo verdadero o falso), ya que la comparación puede ser `true` o `false`.

4. Métodos Básicos

Ahora revisaremos algunos de los métodos básicos de Java. Estas son funciones ya implementadas en Java que ejecutan acciones después de recibir ciertos parámetros.

- `System.out.println`: esta función nos permite imprimir[1] valores al usuario. Por ejemplo:

  ```
  1  int a = 2;
  2  int result = 2*a;
  3
  4  System.out.println(result);
  ```

  Nos imprime el resultado. También podemos juntar el resultado a un texto.

  ```
  1  int a = 2;
  2  int result = 2*a;
  3
  4  System.out.println("El resultado es: " + result);
  ```

- `Scanner`: la clase `Scanner` nos permite recibir *input* del usuario. Lo que el usuario nos escriba con el teclado, lo podemos utilizar en nuestro programa. Para utilizarla debemos importarlo de esta forma: `import java.util.Scanner`.

  ```
  1  // Mensaje con el que pedimos el input
  2  System.out.println("Ingrese su nombre de usuario: ");
  3
  ```

[1] Al decir imprimir nos referimos a mostrar en la pantalla.

```java
// Creamos un Scanner
Scanner scanner = new Scanner(System.in);

// Almacenamos lo que escriba el usuario en la variable username
String username = scanner.nextLine();

// Imprimimos el nombre del usuario
System.out.println("Su nombre de usuario es: " + username);
```

- Transformar *strings* en números: para transformar un *string* que representa un número entero podemos hacer lo siguiente:

```java
String number = "42";
int result = Integer.parseInt(number);
System.out.println(result);
```

Para transformar un *string* que representa un número real podemos hacer lo siguiente:

```java
String number = "10.3";
double result = Double.parseDouble(number);
System.out.println(result);
```

- Funciones matemáticas: en Java están implementadas algunas funciones matemáticas en la librería `Math`. Veamos un ejemplo.

```java
// Calculamos 3 elevado a 4
int number = 3;
double pow = Math.pow(3, 4);
System.out.println(pow);
```

La función pow recibe dos números reales a y b (`Math.pow(double a, double b)`) para calcular el valor de a^b. La libreria `Math` tiene muchas otras funciones[2] que puedes averiguar por tu cuenta.

5. Ejemplos

A continuación presentamos algunos ejemplos resueltos.

Ejemplo 5.1. Haga un programa que pida al usuario su nombre y luego imprima "Hola " acompañado del nombre de la persona.

Solución[3]

```java
package Main;

import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    System.out.println("Ingrese su nombre: ");
```

[2]Puedes revisar por ejemplo: `https://www.tutorialspoint.com/java/lang/java_lang_math.htm`.
[3]En las soluciones se mostrará el código Java completo, incluyendo la declaración del Main.

```
8       Scanner scanner = new Scanner(System.in);
9       String username = scanner.nextLine();
10      System.out.println("Hola " + username);
11    }
12  }
```

Ejemplo 5.2. Haga un programa que pida dos números enteros al usuario e imprima en pantalla la suma de ellos.

Solución 1

```
1   package Main;
2
3   import java.util.Scanner;
4
5   public class Main {
6     public static void main(String[] args) {
7
8       Scanner scanner = new Scanner(System.in);
9       // Pedimos los números como String
10      System.out.println("Ingrese el primer número: ");
11      String numberA = scanner.nextLine();
12      System.out.println("Ingrese el segundo número: ");
13      String numberB = scanner.nextLine();
14
15      // Ahora convertimos los números
16      int a = Integer.parseInt(numberA);
17      int b = Integer.parseInt(numberB);
18      int result = a + b;
19      System.out.println("La suma de ambos números vale: " + result);
20
21    }
22  }
```

Ahora veremos una solución alternativa, en que el número es guardado inmediatamente como `int`.

Solución 2

```
1   package Main;
2
3   import java.util.Scanner;
4
5   public class Main {
6     public static void main(String[] args) {
7
8       Scanner scanner = new Scanner(System.in);
9       // Pedimos los números como enteros
10      System.out.println("Ingrese el primer número: ");
11      int a = scanner.nextInt();
12      System.out.println("Ingrese el segundo número: ");
13      int b = scanner.nextInt();
14      int result = a + b;
15      System.out.println("La suma de ambos números vale: " + result);
16
```

```
17    }
18  }
```

Ejemplo 5.3. Haga un programa que pida al usuario un número que representa el radio de un círculo e imprima su área y su perímetro.

Solución

```
1  package Main;
2
3  import java.util.Scanner;
4
5  public class Main {
6    public static void main(String[] args) {
7
8      Scanner scanner = new Scanner(System.in);
9      System.out.println("Ingrese el radio: ");
10     int r = scanner.nextInt();
11
12     double perimeter = 2 * Math.PI * r;
13     double area = Math.PI * Math.pow(r, 2);
14     System.out.println("El perímetro vale: " + perimeter
15               + " y el área vale: " + area);
16   }
17 }
```

Notamos que en esta solución separamos la función `println` en dos líneas para hacer más legible el código. Además ahora concatenamos las dos variables al mismo tiempo en el *string* de respuesta.

6. Ejercicios

Ejercicio 6.1. Haga un programa que pida dos números al usuario e imprima su resta, su multiplicación y su división.

Ejercicio 6.2. Haga un programa que pida tres números al usuario e imprima su promedio.

Ejercicio 6.3. Haga un programa que reciba tres valores a, b y c que representen a la ecuación $ax + b = c$. Imprime en pantalla el valor de x.