Tipos de datos primitios en Java

Doscker Quan, Diego Gonzalez, Fernando
Lindsey, Maite Elgueta, Juan José Mundo, Adriana

September 8, 2018

1 Tipos de datos primitivos

Los tipos de datos definen valors que una variable puede tomar o representar. Actualmente en java se utilizan dos tipos de datos, los primitivos y los no primitivos. Arreglos y cadenas de carácteres son no primitivos mientras que el resto son primitivos.
Java es un lenguaje que es escrito estaticamente, es decir, todas las variables y métodos deben de ser declarados antes de ser compilado. Un ejemplo de como declarar una variable es:

```
int num;
```

Java utiliza estos tipso de datos para poder mantener la portabilidad de los programas escritos en este, ya que estos datos no cambian de tamaño de ordenador a ordenador. Los tipos de datos primitivos se demuestran en la siguiente tabla:

Nombre	**Descripción**	**Tamaño**	**Rango o Ejemplo**
boolean	true or false	1 bit	true, false
byte	númerico entero	8 bits	-128 a 127
char	Unicode character	16 bits	”a”, ”b”
short	númerico entero	16 bits	-32,768 a 32767
int	númerico entero	32 bits	-2,147,483,648 a 2,147,483,647
long	númerico entero	64 bits	-9 billones a 9 billones
float	números fraccionales	32 bits	números de 7 digitos
double	números fraccionales	64 bits	números de 15 digitos

1.1 Boolean

El tipo de dato "Boolean", puede contener unicamente dos valores, True or False. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {

     boolean b = false;
     System.out.println(b);
    }
}
Output: false
```

1.2 byte

Este tipo de de dato se utiliza usualmente para guardaar valores númericos entre -128 y 127. Debido al tamaño que este tiene, se use cuando el usuario sabe que los valores van a estar adentro del rango.

```
class JavaExample {
    public static void main(String[] args) {
     byte num;

     num = 113;
     System.out.println(num);
    }
}
Output: 113
```

1.3 char

Char se utiliza para guardar valors alfanúmericos. Tiene un tamaño de 2 bytes.

```
class JavaExample {
    public static void main(String[] args) {
     char ch = 'Z';
     System.out.println(ch);
```

```
    }
}
Output: Z
```

1.4 short

En términos de tamaño *short* es mayor que *byte* y menor que *int* y se utiliza para guardar memoria cuando no es necesario un *int*. Tiene un tamaño de 2 bytes. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {
     short num;

     num = 918;
     System.out.println(num);
    }
}
Output: 918
```

1.5 int

Siendo mayor que *byte* y *short*, *int* se utiliza de una manera generalizada para guardar valores númericos. Tiene un tamaño de 4 bytes. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {
     int num;

     num = 150;
     System.out.println(num);
    }
}
Output: 150
```

1.6 long

Usado cuando *int* no es lo suficientemente grande como para guardar los valores asignados. Tiene un tamaño de 8 bytes. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {
     long num = -12332252626L;
     System.out.println(num);
    }
}
Output: -12332252626
```

1.7 float

Todos los tipos de datos númericos previos a *float* se utilizan para números enteros, mientras que float se utiliza para guardar númercos con decimales. Tiene un tamaño de 4 bytes y puede guardar hasta 7 decimales. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {
     float num = 19.98f;
     System.out.println(num);
    }
}
Output: 19.98
```

1.8 double

Al igual que float se utiliza para guardar números con decimales, pero tiene una mayor capacidad. Su tamaño es de 8 bytes y puede guardar hasta 15 decimales. Ejemplo:

```
class JavaExample {
    public static void main(String[] args) {
     double num = -42937737.9d;
     System.out.println(num);
    }
}
Output: -4.293377379e7
```