Universidad de los fuerzas armadas "ESPE"

Nombre: David Caluopiña
NRC: 1940
Fecha: 30/10/24

Tipos de datos en Java

Una variable en Java debe ser un tipo de dato especifico
Los tipos de datos se dividen en dos grupos:

Tipos de datos primitivos:

Incluyen byte, short, int, long, float, double, boolean y char.

Tipos de datos no primitivos

son strings, Arrays y clases

Tipos de datos primitivos

Un tipo de dato primitivo especifica el tamaño y el tipo de valores de variables y no tiene métodos adicionales.
Hay ocho tipos de datos primitivos en Java

Tipo de dato	Tamaño	Descripción
byte	1 byte	Almacena números enteros del -128 al 127
short	2 bytes	Almacena números enteros desde -32,768 hasta 32,767
int	4 bytes	Almacena números enteros desde -2,147,483,648 hasta 2,147,483,647
long	8 bytes	Almacena números enteros desde -9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807.
float	4 bytes	Almacena números fraccionarios. Suficiente para almacenar de 6 a 7 dígitos decimales
double	8 bytes	Almacena números fraccionarios. Suficiente para almacenar de 15 a 16 decimales
boolean	1 bit	Almacena valores verdaderos o falsos
char	2 bytes	Almacena un solo carácter/letra o valores ASCII

Tipos de coma flotante.

Se debe utilizar un tipo de coma flotante siempre que necesite un número con un decimal como 9,99 o 3,14515.
Los tipos de datos flotantes y dobles pueden almacenar números fraccionarios. Para esto se debe finalizar el valor con una "f" para flotantes y una "d" para doubles

ESTILO

ej:

- float myNum = 5.75f;

- double myNum = 19.99d;

Un número de coma flotante también puede ser un número científico con una "e" para indicar la potencia de 10:
ej:

- float f1 = 35e3f;

- double d1 = 12E4d;

Tipos booleanos

En programación se necesita un tipo de dato que solo pueda tener uno de dos valores, como:
sí/No, verdadero/falso.

Los valores booleanos se utilizan principalmente para pruebas condicionales.

Un tipo booleano se declara con la palabra clave booleana y sólo puede tomar los valores verdadero o falso.

ej:

- boolean isJavaFun = true;

Esto es útil para desarrollar la lógica y encontrar respuestas. por ejemplo se puede utilizar un operador de comparación, como el operador ~~mayor~~ (>), para saber si una variable o expresión es verdadera o falsa.
ej:

```
int x = 10;
int y = 9;

System.out.println(x>y)
```

Se usa el operador igual a (==) para evaluar una expresion.
ej:

```
int x = 10;

System.out.println(x==10)
```

Tipo char

Se usa para almacenar un solo carácter. El carácter debe estar entre comillas simples, como 'A' o 'c'
ej:

```
char myGrade = 'B';
System.out.println(myGrade);
```

Tipos de datos no primitivos

Se denominan tipos de referencia porque se refieren a objetos.
La diferencia entre tipos de datos primitivos y no primitivos es:

- Los tipos primitivos en Java están predefinidos e integrados en el lenguaje, mientras que los tipos no primitivos los crea el programador con excepción de "String".

- Los tipos no primitivos se pueden utilizar para llamar a métodos para realizar determinadas operaciones, mientras que los primitivos no

- Los tipos primitivos se escriben al inicio con una letra minúscula como int, mientras que los no primitivos inician con una letra mayúscula como String.

- Los tipos primitivos siempre tienen un valor, mientras que los tipos no primitivos pueden ser nulos

Los tipos no primitivos son: Strings, arrays, clases.

Strings

Se usa para almacenar texto.
Una variable String contiene un conjunto de caracteres entre comillas dobles
ej:

```
String greeting = "Hello";
```

Strings Length

Un String en Java es un objeto que contiene métodos que pueden realizar ciertas operaciones en cadenas
Por ejemplo la longitud de un String se encuentra con length
ej:

```
String txt = "ABCDEF"
System.out.println("la longitud de txt es:" + txt.length());
```

ESTILO

otros métodos para los strings son:
.toUpperCase() y .toLowerCase():
ej:

```
String txt = "Hola mundo";
System.out.println(txt.toUpperCase()); outputs: "HOLA MUNDO"
System.out.println(txt.toLowerCase()); outputs: "hola mundo"
```

Encontrar un caracter en una cadena

El método indexOf() indica la posición de un texto específico en una cadena
ej:

```
String txt = "Hola mundo lindo";
System.out.println(txt.indexOf("mundo")); outputs: 3
```

Arrays

Los Arrays se usan para almacenar múltiples valores en una sola variable, en vez de declarar variables separados para cada valor.
Para declarar un array, se define el tipo de variable entre corchetes
ej:

```
String[] cars;
```

Para colocar valores a la variable declarada se los coloca con comas y dentro de llaves.
ej:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
int[] myNum = {10, 20, 30, 40};
```

Para acceder a un elemento dentro de la matriz se coloca el puesto en el que se encuentra dicho elemento
ej:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]); // outputs Volvo
```

Para cambiar un elemento de una matriz se coloca el puesto en el que se encuentra el elemento a cambiar.
ej:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
```

System.out.println(cars[0]);

ahora Opel esta en el puesto de Volvo

Longitud de un arreglo.

Para saber cuántos elementos conforman una matriz se usa length
ej:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars.length); // outputs: 4