Clases: String, Integer, Float, Date

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1. String
2. Integer
3. Float
4. Date

1 | String

String

Para utilizar datos de tipo texto, vamos a declararlos como String. Las Strings nos permiten utilizar funciones ya programadas, que le pertenecen. Las llamamos métodos.

```
{}
public static void main(String[] args){
    String nombre;
}
```

A partir de esta variable vamos a ver cómo utilizar algunos de estos métodos de uso frecuente.

{código}

Métodos usados: **.length(), .toUpperCase(), .equals(), .toChar()**

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt();
```

{código}

```
String nombre="Juan";
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals("JUAN"))
{
    System.out.println("Se pasó a mayúscula");
}
inicial = nombre.charAt();
```

Declaramos tres variables, una de tipo **String,** otra de tipo **int** y la última de tipo **char.** A la variable **nombre** le asignamos la cadena "Juan".

{código}

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt();
```

Mediante el método length() de la clase String podemos calcular la longitud de la cadena almacenada en nombre.

{código}

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt();
```

Convierte a mayúscula todos los caracteres contenidos en nombre.

{código}

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt();
```

Comprueba que se convirtió a mayúscula correctamente. La cadena contenida en la variable **nombre** es exactamente “JUAN”, recordar que “Juan” no es igual a “JUAN”.

{código}

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt();
```

Se muestra un mensaje, comprobando que los caracteres están en mayúscula.

{código}

```
String nombre=”Juan”;
int cantidad;
char inicial;

cantidad= nombre.length();

nombre=nombre.toUpperCase();

if (nombre.equals(“JUAN”))
{
    System.out.println(“Se pasó a mayúscula”);
}
inicial = nombre.charAt(0);
```

Obtenemos el primer carácter de la cadena, en la variable inicial de tipo char.

String vacía

Si aún no hemos asignado nada a las String, entonces, contiene un valor null, en ese caso no se pueden usar los métodos.

```
String nombre=null;

if (nombre==null)
{
    System.out.println(“Cadena con valor nulo”);
}
```

{código}

```
String nombre=null;
```

Definimos la variable nombre, pero no se le asigna una cadena.

```
if (nombre==null)
{
    System.out.println(“Cadena con valor nulo”);
}
```

{Código}

```
String nombre=null;

if (nombre==null){
    System.out.println(“Cadena con valor nulo”);
}
```

Comprueba si aun no se ha inicializado.

String vacía

```
String nombre=null;

if (nombre==null)
{
    System.out.println("Cadena con valor nulo");
}
```

Indica que contiene un valor nulo.

String vacía y String nula

En una String podemos tener las dos situaciones: puede tener un valor nulo o estar vacía.

`String nombre=null;`	Cadena que aún no se ha inicializado, esta en null. Aun no puedo utilizar métodos propios.
`nombre =””;`	Cadena vacía.
`nombre=”Juan”;`	Cadena inicializada con el valor “Juan”.

2 | Integer

Integer

Integer como clase y no como tipo primitivo se utiliza de una forma distinta. Para comenzar a utilizar un Integer tenemos dos posibilidades:

```
Integer valor=0;
En este caso definimos y creamos un Integer, dándole un valor inicial 0-

Integer num= new Integer(1);
```

En la segunda forma hacemos algo similar, pero la parte de la izquierda es la definición y la parte de la derecha la creación con un valor inicial 1.

Cuando solo definimos algo de tipo Integer, su valor inicial es null, es necesario darle un valor inicial.

{código}

Comprobamos la relación entre dos números enteros, utilizando clases
Métodos usados: `.equal()`, `.compareTo()`

```
Integer valor1=10;
Integer valor2=30;
int comparar;

if (valor1.equals(valor2))
    System.out.println(“Son iguales”);
else
{   comparar=valor1.compareTo(valor2);
    if (comparar>0)
        System.out.println(“valor1 es mayor que valor2”);
    else
        System.out.println(“valor2 es mayor que valor1”);
}
```

{código}

```
Integer valor1=10;
Integer valor2=30;
int comparar;
```

Definición de variables, que vamos a utilizar.

```
if (valor1.equals(valor2))
    System.out.println(“Son iguales”);
else
{   comparar=valor1.compareTo(valor2);
    if (comparar>0)
        System.out.println(“valor1 es mayor que valor2”);
    else
        System.out.println(“valor2 es mayor que valor1”);
}
```

{código}

```
Integer valor1=10;
Integer valor2=30;
int comparar;

if (valor1.equals(valor2))
    System.out.println("Son iguales");
else
{   comparar=valor1.compareTo(valor2);
    if (comparar>0)
        System.out.println("valor1 es mayor que valor2");
    else
        System.out.println("valor2 es mayor que valor1");
}
```

Comprobamos si son iguales.

{código}

```
Integer valor1=10;
Integer valor2=30;
int comparar;

if (valor1.equals(valor2))
    System.out.println("Son iguales");
else
{   comparar=valor1.compareTo(valor2);
    if (comparar>0)
        System.out.println("valor1 es mayor que valor2");
    else
        System.out.println("valor2 es mayor que valor1");
}
```

Compara la relación entre dos valores, si valor1 es mayor, dará 1, si valor2 es mayor, dará -1.

{código}

Comprobamos la relación entre dos números enteros, utilizando clases
Métodos usados: `.equal()`, `.compareTo()`

```
Integer valor1=10;
Integer valor2=30;
int comparar;

if (valor1.equals(valor2))
    System.out.println("Son iguales");
else
{   comparar=valor1.compareTo(valor2);
    if (comparar>0)
        System.out.println("valor1 es mayor que valor2");
    else
        System.out.println("valor2 es mayor que valor1");
}
```

Muestra el resultado, según lo obtenido en comparar.

3 | Float

Float

Float como clase y no como tipo primitivo se utiliza de una forma distinta.
Para comenzar a utilizar un Float tenemos dos posibilidades:

```
Float coeficiente=2.5f;
```

En este caso definimos y creamos un Float, dándole un valor inicial 2.5f, la f quiere decir float, si no lo ponemos se asume que es algo de tipo Double.

```
Float num= new Float(0.5);
```

En la segunda forma hacemos algo similar, pero la parte de la izquierda es la definición y la parte de la derecha la creación con un valor inicial 0.5.

Al igual que Integer, si no tiene un valor inicial, está en **null**.

Cuando solo definimos algo de tipo Float, su valor inicial es null, siempre es necesario darle un valor inicial.

4 | Date

Date

La clase **Date** permite trabajar con fechas. A diferencia de las clases que vimos hasta ahora, si definimos un objeto de tipo **Date**, no es posible hacerlo vacío. Un objeto **Date** se crea con un valor inicial que el la fecha actual.

```
import java.util.Date;

public class Main {
     public static void main(String[] args) {
          Date fecha=new Date();
          System.out.println(fecha.toString());
```

Muestra la fecha actual

Para usar Date es necesario agregar import java.util.Date;

Date

Para crear un **Date** con otro valor, lo hacemos de la siguiente manera:

```
public static void main(String[] args) {
     Date fecha=new Date(120,11,5);
     System.out.println(fecha.toString());
```

Muestra 5/12/2020

Los parámetros que usamos son año, mes, día, teniendo en cuenta:
al valor que colocamos para año le suma 1900: 1990+120 =2020
los meses los enumera desde cero o sea 11 es en realidad 12 (diciembre)

De esta forma obtenemos la fecha que necesitamos


DigitalHouse>
Coding School