

Programación en Java

Inicio

Elementos de un programa informático

Programa y lenguajes de programación

El lenguaje Java

Variables, sentencias y paquetes

Tipos de datos

Datos primitivos

Datos no primitivos

String

Operadores y expresiones

Comentarios en Java

Constantes y literales

Entrada y salida de información por consola

Introducción a la programación orientada a objetos

Estructuras básicas de control

⌘ Recursividad

Programación orientada a objetos POO

Arrays

⚡ POO avanzada

Colecciones

Programación avanzada

Interfaces de usuario

Entrada/Salida (I/O) de la información

Persistencia de la información

Atajos de teclado para IntelliJ

About me

No primitivos u objetos

En Java, los tipos de datos no primitivos son los tipos de datos de referencia o los tipos de datos creados por el usuario. Todos los tipos de datos no primitivos se implementan utilizando conceptos de objeto. Cada variable del tipo de datos no primitivo es un objeto. Los tipos de datos no primitivos pueden utilizar métodos adicionales para realizar determinadas operaciones. El valor predeterminado de la variable de tipo de datos no primitivos es nulo.

String - Cadena de caracteres

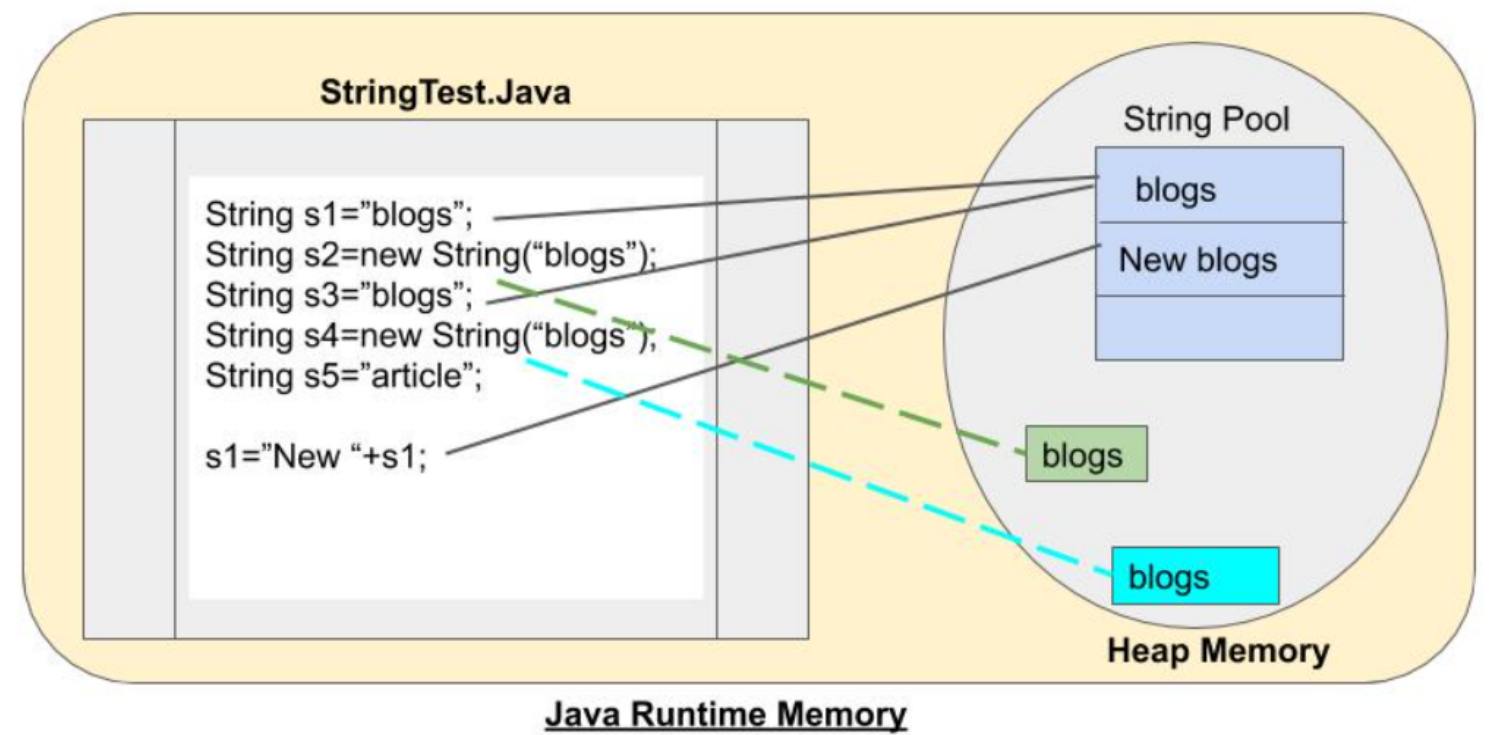
String es una clase integrada en el lenguaje Java ampliamente utilizada y definido en el paquete java.lang. Representa cadenas de caracteres y se utilizan para almacenar varios atributos como nombre de usuario, contraseña, etc. Los String son **inmutables**; es decir, no se pueden modificar una vez creados. Siempre que se modifica un objeto String, en realidad se crea uno nuevo String.

Existen varias formas para crear un String:

```
String texto = "Severo Ochoa";
String texto2 = new String("Severo Ochoa");
```

Asignación de memoria de objetos String

La memoria se divide en dos partes, el String Pool y la memoria Heap.



Veamos como funcionaría para la imagen anterior.

- Siempre que creamos un String con comillas dobles, se almacenan en String Pool. String Pool almacena el único valor en él. Es por eso que, String s1 = "blogs" se almacenan como el primer valor en el String Pool.
- Siempre que creamos un objeto usando la palabra clave *new*, se almacena en la memoria Heap pero fuera del String Pool. Aquí se puede almacenar valores duplicados ya que pertenece a diferentes objetos. Entonces, para la declaración String s2 = new String("blogs"), aunque el valor de s1 y s2 es el mismo, s2 se almacenará fuera del String Pool.
- Cuando creamos otro String usando comillas dobles, primero verifica todos los valores en el String Pool y si coincide con alguno se asigna la misma ubicación asignada a otro objeto de referencia. Por lo tanto, String s3 = "blogs" no agregará una nueva entrada en el String Pool.
- Pero si creamos otro objeto con un valor existente usando la palabra clave *new*. Asignará nueva memoria al nuevo objeto en el Heap. Por lo tanto, a String s4 = new String ("blogs") se le asignará nueva memoria.
- Ahora, si manipulamos el valor en s1 usando s1 = "New" + s1, no actualizará la entrada o referencia existente de s1. Este proceso creará una nueva entrada en el String Pool con el valor "New blogs" y la referencia de memoria cambiaría para el objeto s1.

Java String Class Methods

La clase `java.lang.String` proporciona muchos métodos útiles para realizar operaciones en la secuencia de valores char.

Date

En sus inicios se crearon las clases **java.util.Date** y **java.sql.Date** para almacenar y manejar fechas. Pero ambas clases son defectuosas en diseño e implementación. Por ejemplo, las clases existentes (como *java.util.Date* y *SimpleDateFormat*) no son thread-safe, lo que genera posibles problemas de concurrencia para los usuarios.

Por tanto, desde Java 8 y posteriores se ha desarrollado una nueva **API java.time** que resuelve los problemas que presentaban las librerías anteriores.

Tabla de contenidos

String - Cadena de caracteres


Asignación de memoria de objetos String

Java String Class Methods

Date

Enum en Java

Date-time types in Java & SQL	Legacy class	Modern class	SQL standard data type
Moment in UTC	java.util. Date java.sql. Timestamp	java.time. Instant	TIMESTAMP WITH TIME ZONE
Moment with offset-from-UTC (hours-minutes-seconds)	(lacking)	java.time. OffsetDateTime	TIMESTAMP WITH TIME ZONE
Moment with time zone ('Continent/Region')	java.util. GregorianCalendar javax.xml.datatype. XMLGregorianCalendar	java.time. ZonedDateTime	TIMESTAMP WITH TIME ZONE
Date & Time-of-day (no offset, no zone) Not a moment	(lacking)	java.time. LocalDateTime	TIMESTAMP WITHOUT TIME ZONE
Date only (no offset, no zone)	java.sql. Date	java.time. LocalDate	DATE
Time-of-day only (no offset, no zone)	java.sql. Time	java.time. LocalTime	TIME WITHOUT TIME ZONE
Time-of-day with offset (impractical, not used)	(lacking)	java.time. OffsetTime	TIME WITH TIME ZONE

 More information

[Why do we need a new date and time library?](#)

Ejemplo de código para mostara la fecha y la hora:

```
//mostrar fecha
LocalDate ld = LocalDate.now();
System.out.println(ld);

//mostrar hora
LocalTime lt = LocalTime.now();
System.out.println(lt);

//mostrar fecha y hora
LocalDateTime ldt = LocalDateTime.now();
System.out.println(ldt);

//formatear la fecha con un formato dado
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss")
String formatted = formatter.format(ldt);
System.out.println(formatted);
```

Enum en Java

El tipo enumerado es un tipo de datos especial que permite que una variable sea un conjunto de constantes predefinidas. La variable debe ser igual a uno de los valores que se han predefinido para ella.

Debido a que son constantes, los nombres de los campos del tipo enum deben estar en letras mayúsculas.

En Java, se define un enumerado utilizando la palabra clave **enum** seguido del nombre siguiendo la convención del nombrado de clases. Primera letra en mayúscula y CamelCase.

Para crear un enum en Java, botón derecho en el paquete --> new Java class y seleccionamos enum.

Ejemplo de enumerado:

```
public enum PuntosCardinales {
    NORTE, SUR, ESTE, OESTE
}

public class Main {

    public static void main(String[] args) {
        PuntosCardinales myVar = PuntosCardinales.ESTE;
        System.out.println(myVar);
    }

}
```