Argentina Programa

Clase 22: Strings - Manejo de Fechas

Agenda

Familiarizarse con los conceptos básicos relacionados a Strings - Manejo de Fechas

- Strings - Manejo de Fechas
- Práctica Strings - Manejo de Fechas

String

La clase String en Java es una clase fundamental que representa una secuencia de caracteres Unicode.

Las cadenas son inmutables en Java, lo que significa que una vez que se crea una cadena, no se pueden cambiar sus caracteres.

String

Aquí hay algunos de los métodos más comunes de la clase String:

- length()
- charAt(int index)
- substring(int startIndex) o substring(int startIndex, int endIndex)
- concat(String str)
- replace(char oldChar, char newChar)
- toLowerCase() o toUpperCase()
- trim()

String

La clase String es muy versátil y se utiliza comúnmente en Java para representar texto, incluyendo nombres de variables, nombres de métodos, mensajes de error y mucho más.

La clase String proporciona una amplia gama de métodos que permiten la comparación, concatenación, búsqueda y reemplazo de texto.

Además, la clase String es un tipo de objeto en Java, lo que significa que se puede utilizar en conjunto con otros objetos para crear aplicaciones más complejas.

String

Conceptos relacionados con la clase String en Java:

- Declaración
- Concatenación
- Longitud
- Subcadena
- Búsqueda
- Comparación
- Mayúsculas/minúsculas
- Reemplazo
- División
- Expresiones regulares

String

Ejemplos

```
String nombre = "Ada Lovelace";

String saludo = "Hola, " + nombre + "!";

String password = "secret";
if (password.equals("secret")) {
System.out.println("La contraseña es correcta.");
}
```

String

En Java existen otras formas de representar texto, como la clase StringBuilder y la clase StringBuffer.

Sin embargo, la clase String es la más común y se utiliza en la mayoría de los casos debido a su simplicidad y facilidad de uso.

StringBuilder

La clase StringBuilder es una clase que permite crear y manipular objetos de tipo cadena en Java.

A diferencia de la clase String, que es inmutable, la clase StringBuilder es mutable, lo que significa que sus valores pueden ser modificados después de ser creados.

La clase StringBuilder es útil cuando se requiere hacer muchas operaciones de concatenación.

StringBuilder

```
StringBuilder sb = new StringBuilder();
sb.append("Hello");
sb.append(" ");
sb.append("World");
System.out.println(sb.toString()); // imprime "Hello World"
StringBuilder sb = new StringBuilder("Hello");
sb.insert(5, " World");
System.out.println(sb.toString()); // imprime "Hello World"
```

StringBuffer

La clase StringBuffer permite crear y manipular objetos de tipo cadena. A diferencia de la clase String, que es inmutable, la clase StringBuffer es mutable, lo que significa que sus valores pueden ser modificados después de ser creados.

A diferencia de la clase StringBuilder, la clase StringBuffer es segura en cuanto a concurrencia, lo que significa que se puede utilizar en entornos multithread sin problemas de seguridad.

StringBuffer

```
StringBuffer sb = new StringBuffer();
sb.append("Hello");
sb.append(" ");
sb.append("World");
System.out.println(sb.toString()); // imprime "Hello World"

StringBuffer sb = new StringBuffer("Hello"); sb.insert(5, " World");
System.out.println(sb.toString()); // imprime "Hello World"
```

Manejo de fechas

En Java, el manejo de fechas se realiza a través de la clase java.util.Date y su subclase java.util.Calendar.

Estas clases permiten representar fechas y horas, y realizar operaciones como la obtención de la fecha y hora actual, la comparación de fechas, la adición o sustracción de días, meses o años, entre otras.

java.util.Date

```
Date fechaActual = new Date();
System.out.println(fechaActual); // imprime la fecha y hora actual

Date fechaFutura = new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 24 *
7);
System.out.println(fechaFutura); // imprime la fecha y hora actual más una semana

if (fechaActual.before(fechaFutura)) {
System.out.println("La fecha actual es antes que la fecha futura");
}
```

java.util.Calendar

```
Calendar cal = Calendar.getInstance();
System.out.println(cal.getTime()); // imprime la fecha y hora actual
cal.add(Calendar.DATE, 7);
System.out.println(cal.getTime()); // imprime la fecha y hora actual más una semana

if (cal.before(Calendar.getInstance())) {
        System.out.println("La fecha actual es antes que la fecha futura");
}
```

java.util.time

En Java 8 y versiones posteriores, se introdujo una nueva API de fechas y horas llamada java.time que proporciona una forma más fácil y moderna de trabajar con fechas y horas.

La API de java.time ofrece una gran cantidad de clases y métodos para representar, manipular y formatear fechas y horas, incluyendo LocalDate, LocalTime, LocalDateTime entre otros.

java.util.time

```
LocalDate fechaActual = LocalDate.now();
System.out.println(fechaActual); // imprime la fecha actual
LocalDate fechaFutura = fechaActual.plusDays(7);
System.out.println(fechaFutura); // imprime la fecha actual más una semana

if (fechaActual.isBefore(fechaFutura)) {
	System.out.println("La fecha actual es antes que la fecha futura");
}

LocalDateTime fechaHoraActual = Local
```

CONSULTAS?

Muchas Gracias!