Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

FCQI

Reporte de actividad

Programación orientada a objetos.

Práctica No. 5:

Investigación y práctica sobre la clase String en Java

Alumno

Joshua Osorio Osorio - 1293271

Docente

Itzel Barriba Cazares

3 de marzo de 2025

ÍNDICE

Objetivo

Investigar las características y métodos principales de la clase String en Java, comprender su inmutabilidad y aplicar sus métodos en ejercicios prácticos.

Introducción

En Java, la clase String es una de las más utilizadas, ya que permite la manipulación y el almacenamiento de cadenas de caracteres. Sin embargo, String tiene una característica fundamental: es inmutable, lo que significa que una vez creado un objeto String, su contenido no puede cambiar.

Debido a esta inmutabilidad, Java proporciona otras clases como StringBuilder y StringBuffer, que permiten manipulación eficiente de cadenas sin crear múltiples objetos en memoria. En este reporte, se investigarán sus diferencias y se explorarán los métodos más importantes de String, aplicándolos en un programa práctico que permitirá al usuario realizar diversas operaciones con una frase ingresada.

El propósito de esta actividad es comprender el funcionamiento de las cadenas en Java, su importancia en el desarrollo de software y la optimización del uso de memoria al trabajar con ellas.

Para llevar a cabo esta investigación y práctica, se requirió lo siguiente.

Lista de materiales

- Computadora con entorno de desarrollo compatible con Java.
- JDK (Java Development Kit) instalado (versión 8 o superior recomendada).
- Editor de texto.
- Documentación oficial de Java para referencia sobre la clase String.
- Recursos de apoyo como libros y artículos en línea sobre String en Java.
- Compilador de Java (incluido en el JDK) para ejecutar y probar el código.

Desarrollo

¿Qué es la clase String en Java?[1].

La clase String en Java es una de las clases más utilizadas y forma parte del paquete java.lang. Representa una secuencia de caracteres y permite manipular cadenas de texto con diversos métodos predefinidos.

Características principales:

- Es una clase inmutable.
- Se puede inicializar con "" (comillas dobles).

```java
String texto = "";
```

- Permite concatenación de cadenas con + o concat().

```java
String texto = "Hola";
String texto1 = texto + " Mundo"; //salida> Hola mundo
String texto2 = concat(texto, " Mundo"); //salida> Hola mundo
```

- Ofrece múltiples métodos para manipulación de cadenas, como length(), toUpperCase(), replace(), entre otros.

Inmutabilidad de los objetos String[2].

Los objetos de tipo String en Java son inmutables, lo que significa que una vez creados, su contenido no puede ser modificado.

```java
String texto = "Hola";
texto = texto + " Mundo";  // Se crea un nuevo objeto en memoria
```

Aquí, texto no se modifica, sino que se crea una nueva cadena "Hola Mundo", y la referencia texto apunta a esta nueva instancia.

Ventajas de la inmutabilidad

- Seguridad: Evita modificaciones accidentales en los datos.
- Optimización: Java usa un String Pool, donde reutiliza cadenas idénticas para mejorar el rendimiento.

Diferencias entre String, StringBuilder y StringBeffer

Característica	String	StringBuilder	StringBuffer
Mutabilidad	Inmutable	Mutable	Mutable
Rendimiento	Más lento (crea nuevos objetos)	Más rápido (modifica el mismo objeto)	Más rápido, pero más lento que StringBuilder
Seguridad	Seguro en hilos múltiples	No es seguro para hilos	Es seguro para hilos (synchronized)
Uso recomendado	Cadenas constantes	Manipulación eficiente de cadenas en un solo hilo	Uso en entornos concurrentes

StringBuilder es mutable y no está sincronizado.
Si varios hilos acceden al mismo objeto StringBuilder al mismo tiempo, pueden modificarlo simultáneamente y generar datos corruptos.

StringBuffer es seguro en hilos
StringBuffer es mutable pero está sincronizado (synchronized), lo que significa que protege su acceso en entornos de múltiples hilos.
Usa la palabra clave synchronized, que hace que solo un hilo pueda modificar el objeto a la vez, evitando problemas de concurrencia.

Ejemplo de StringBuilder (eficiente):

```java
StringBuilder sb = new StringBuilder("Hola");
sb.append(" Mundo");
System.out.println(sb.toString()); // "Hola Mundo"
```

10 métodos importantes de la la clase String [3].

1. `length()` - Devuelve la longitud de la cadena.

```java
String texto = "Hola";
System.out.println(texto.length()); // 4
```

2. `charAt(int index)` - Obtiene el carácter en la posición dada.

```java
System.out.println(texto.charAt(1)); // 'o'
```

3. `substring(int inicio, int fin)` - Extrae una subcadena.

```java
System.out.println(texto.substring(1, 3)); // "ol"
```

4. `toUpperCase() / toLowerCase()` - Convierte a mayúsculas o minúsculas.

```java
System.out.println(texto.toUpperCase()); // "HOLA"
System.out.println(texto.toLowerCase()); // "hola"
```

5. `replace(String old, String new)` - Reemplaza caracteres o palabras.

```java
System.out.println(texto.replace("o", "0")); // "H0la"
```

6. `contains(String str)` - Verifica si la cadena contiene un texto.

```java
System.out.println(texto.contains("la")); // true
```

7. `startsWith(String prefix)` / endsWith(String suffix) - Verifica inicio o fin.

```java
String texto = "Hola";
System.out.println(texto.length()); // 4
```

8. `indexOf(String str)` - Encuentra la posición de una subcadena.

```java
System.out.println(texto.indexOf("la")); // 2
```

9. `split(String regex)` - Divide la cadena en partes.

```java
String[] palabras = "Hola Mundo".split(" ");
System.out.println(palabras[1]); // "Mundo"
```

10. `trim()` - Elimina espacios en blanco al inicio y fin.

```java
System.out.println(" Hola ".trim()); // "Hola"
```

Ejemplos de los usos de los métodos

Implementación de programa en Java.

```java
import java.util.*;

public class Principal{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Solicitar la frase al usuario
        System.out.println("Ingrese una frase:");
        String frase = input.nextLine();
        // Mostrar la longitud de la frase
        System.out.println("Longitud de la frase: " + frase.length());
        // Convertir la frase a mayúsculas y minúsculas
        System.out.println("En mayúsculas: " + frase.toUpperCase());
        System.out.println("En minúsculas: " + frase.toLowerCase());
        // Contar cuántas veces aparece una letra dada
        System.out.println("Ingrese una letra para contar su frecuencia:");
        char letra = input.next().charAt(0);
        int contador = 0;
        for (int i = 0; i < frase.length(); i++) {
            if (frase.charAt(i) == letra) {
                contador++;
            }
        }
        System.out.println("La letra '" + letra + "' aparece " + contador + "
veces.");

        input.nextLine(); // Limpiar buffer

        // Verificar si la frase empieza y termina con una palabra específica
        System.out.println("Ingrese la palabra para verificar si la frase
inicia con ella:");
        String inicio = input.nextLine();
        System.out.println("¿Empieza con \"" + inicio + "\"? " +
frase.startsWith(inicio));

        System.out.println("Ingrese la palabra para comprobar si la frase
termina con ella:");
```

```
        String fin = input.nextLine();
        System.out.println("¿Termina con \"" + fin + "\"? " +
frase.endsWith(fin));

        // Reemplazar una palabra en la frase
        System.out.println("Ingrese la palabra a reemplazar:");
        String palabraVieja = input.nextLine();
        System.out.println("Ingrese la nueva palabra:");
        String palabraNueva = input.nextLine();
        System.out.println("Nueva frase: " + frase.replace(palabraVieja,
palabraNueva));
    }
}
```

Conclusiones

Durante esta práctica, se investigaron las características de la clase String en Java y se comprendió su inmutabilidad, lo que implica que sus objetos no pueden ser modificados directamente. Además, se analizaron las diferencias con StringBuilder y StringBuffer, destacando sus usos en optimización de memoria y concurrencia.

Se implementó un programa en Java que demuestra la manipulación de cadenas mediante diversos métodos de String, permitiendo obtener su longitud, convertir a mayúsculas/minúsculas, contar caracteres, verificar prefijos y sufijos, y realizar reemplazos.

Durante la práctica, surgieron problemas al ingresar texto después de capturar un número, lo cual se solucionó utilizando nextLine() para limpiar el buffer. También se observó que el uso de clases facilitó la organización del código.

Este ejercicio permitió reforzar los conocimientos sobre String y su aplicación en situaciones prácticas.

Bibliografía

[1] “Java a fondo,” *Google Books*. [Online]. Available:

https://books.google.com.mx/books?hl=es&lr=&id=WcL2DQAAQBAJ&oi=fnd&pg

=PA1&dq=%C2%BFQu%C3%A9+es+la+clase+String+en+Java%3F&ots=iRwaAyCxI7&sig=Y9QsnqLumorAhget_jn3QCjEE7w#v=onepage&q=%C2%BFQu%C3%A9%20es%20la%20clase%20String%20en%20Java%3F&f=false.

[2] I. González Pérez, A. J. Sánchez Martín, D. Vicente Hernández, y Departamento de Informática y Automática Universidad de Salamanca, «Java Threads (Hilos en Java)», *Google Scholar*, p. 1, may 2002, Disponible en: http://lsi.vc.ehu.es/pablogn/docencia/ISO/7%20Gest%20de%20Procesos/HilosJavaUSAL.pdf

[3] Dotnet-Bot, “String Class (Java.Lang).” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/java.lang.string?view=net-android-35.0.