**1. `length()`:** Retorna la longitud de la cadena, es decir, el número de caracteres que contiene.

**2. `charAt(int index)`:** Retorna el carácter en la posición especificada por `index`.

**3. `concat(String str)`:** Une dos cadenas, adjuntando `str` al final de la cadena actual.

**4. `indexOf(String str)`:** Retorna el índice de la primera aparición de `str` dentro de la cadena.

**5. `substring(int beginIndex)`:** Retorna una subcadena empezando desde `beginIndex` hasta el final de la cadena.

**6. `substring(int beginIndex, int endIndex)`**: Retorna una subcadena que empieza en `beginIndex` y termina justo antes de `endIndex`.

**7. `toUpperCase()`:** Retorna una nueva cadena con todos los caracteres en mayúscula.

**8. `toLowerCase()`:** Retorna una nueva cadena con todos los caracteres en minúscula.

**9. `trim()`:** Retorna una copia de la cadena sin espacios en blanco al inicio y al final.

**10. `startsWith(String prefix)`:** Verifica si la cadena comienza con el prefijo especificado.

**11. `endsWith(String suffix)`:** Verifica si la cadena termina con el sufijo especificado.

**12. `replace(char oldChar, char newChar)`:** Retorna una nueva cadena donde los caracteres `oldChar` son reemplazados por `newChar`.

**13. `replaceAll(String regex, String replacement)`:** Retorna una nueva cadena donde todas las coincidencias de `regex` son reemplazadas por `replacement`.

**14. `split(String regex)`:** Divide la cadena en subcadenas usando el patrón `regex` como delimitador y retorna un array de subcadenas.

**15. `isEmpty()`:** Verifica si la cadena está vacía, es decir, si su longitud es cero.

**16. `contains(CharSequence s)`:** Verifica si la cadena contiene la secuencia de caracteres especificada por `s`.

**17. `valueOf(dataType value)`:** Convierte otros tipos de datos en cadenas de caracteres.

**18. `compareTo(String anotherString)`:** Compara dos cadenas lexicográficamente.

**19. `equalsIgnoreCase(String anotherString)`:** Compara dos cadenas ignorando diferencias de mayúsculas y minúsculas.

**20. `codePointAt(int index)`:** Retorna el valor Unicode del carácter en la posición `index`.

**21. `codePointBefore(int index)`:** Retorna el valor Unicode del carácter antes de la posición `index`.

**22. `getBytes()`:** Convierte la cadena en un array de bytes usando la codificación por defecto de caracteres.

**23. `join(CharSequence delimiter, CharSequence... elements)`:** Une varias cadenas usando el delimitador especificado.

**24. `matches(String regex)`:** Verifica si la cadena coincide con el patrón regex completo.

**25. `strip()`:** Retorna una cadena sin espacios en blanco al inicio y al final.

**26. `repeat(int count)`:** Retorna una nueva cadena concatenando la cadena actual `count` veces.

**27. `lines()`:** Retorna un `Stream` de líneas divididas por saltos de línea.

**28. `isBlank()`:** Verifica si la cadena está vacía o contiene solo espacios en blanco.

**29. `stripLeading()`:** Elimina los espacios en blanco del inicio de la cadena.

**30. `stripTrailing()`:** Elimina los espacios en blanco del final de la cadena.

**31. `lines()` (Java 11):** Retorna un `Stream` de líneas de la cadena.

**32. `formatted(Object... args)` (Java 13):** Reemplaza cada instancia de `%s` en la cadena con los argumentos proporcionados.

**33. `repeat​(int count)` (Java 11):** Retorna una cadena que es una concatenación de la cadena actual repetida `count` veces.

**34. `indent​(int n)` (Java 12):** Retorna una cadena en la que cada línea está precedida por espacios de sangrado.

**35. `translateEscapes()` (Java 15):** Reemplaza las secuencias de escape Unicode en la cadena por sus valores Unicode correspondientes.

**36. `stripIndent()` (Java 15):** Elimina el sangrado común a todas las líneas de la cadena.

**37. `formatted()` (Java 15):** Permite formatear una cadena utilizando los métodos `String` y otros tipos de datos.

Estos métodos proporcionan diversas funcionalidades para manipular, comparar, buscar, transformar y formatear cadenas de caracteres en Java.