

Unidad: Cadenas

String e inmutabilidad, búsqueda y transformación; StringBuilder; UTF-8 y normalización

Introducción a la Programación

Octubre de 2025

Caso de apertura: Mensaje crítico en tránsito

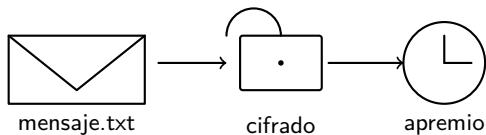
Situación: Durante una evacuación nocturna, el equipo recibe un archivo de texto con instrucciones mezcladas y ruido (espacios de más, mayúsculas/minúsculas inconsistentes, símbolos sueltos). El tiempo apremia: hay que limpiar el mensaje, buscar el punto de encuentro y confirmar la clave de acceso.

Desafío: ¿Cómo extraer la información útil de un texto sucio sin estructuras adicionales?

Herramientas de la unidad: Acceso (índices), búsqueda (`indexOf`), transformación (`trim/toLowerCase/replace`), y eficiencia con `StringBuilder`.

Entrega observable: Una salida limpia con Lugar: y Clave: lista para operar.

Visual del caso



Texto en tránsito, cifrado/ruido y tiempo limitado: motivación para operar con cadenas (acceso, búsqueda, transformación) y construir respuestas rápidas.

Reglas y restricciones (para pensar)

- ▶ Usaremos únicamente `String` y `StringBuilder` (sin otras herramientas aún).
- ▶ Mantener espacios y signos; respetar mayúsculas/minúsculas si aplica.
- ▶ Lectura en UTF-8 (evitar problemas de caracteres).
- ▶ (Nota breve) Si hay acentos raros, normalizar (NFC) antes de comparar.

Guía técnica mínima

1. Leer el archivo en UTF-8.
2. (Opcional) Normalizar a NFC.
3. Recorrer con `charAt(i)` y construir con `StringBuilder`.
4. Filtrar/ajustar: espacios, caso, símbolos.
5. Extraer Lugar: y Clave: usando índices y subcadenas.

Mini-demostración (Java)

```
import java.nio.file.*; import java.nio.charset.*;
class Demo{
    static String cesar(String s, int k){
        k=((k%26)+26)%26; StringBuilder out=new StringBuilder(s.length());
        for(int i=0;i<s.length();i++){
            char c=s.charAt(i);
            if(c>='A'&&c<='Z') out.append((char)('A'+(c-'A'+k)%26));
            else if(c>='a'&&c<='z') out.append((char)('a'+(c-'a'+k)%26));
            else out.append(c);
        } return out.toString();
    }
    public static void main(String[] a) throws Exception{
        String msg=Files.readString(Path.of("mensaje.txt"), StandardCharsets.
        UTF_8);
        System.out.println(cesar(msg,-3));
    }
}
```

Puente con la unidad

- ▶ **Acceso a texto:** `length`, `charAt`, `substring`.
- ▶ **Búsqueda:** `indexOf/lastIndexOf`; `contains`; prefijos/sufijos.
- ▶ **Transformación:** `trim/strip`, `toLowerCase/toUpperCase`, `replace` (literal).
- ▶ **Eficiencia:** Inmutabilidad de `String` vs `StringBuilder` en bucles.
- ▶ **Codificación/normalización:** UTF-8; NFC para comparar "lo mismo" de forma robusta.

Objetivos de aprendizaje

Al finalizar, podrás:

1. Explicar inmutabilidad de `String` y cuándo usar `StringBuilder`.
2. Acceder y recorrer cadenas con seguridad.
3. Buscar y contar ocurrencias con operaciones básicas de cadenas.
4. Aplicar transformaciones comunes (may/min, trim, replace literal).
5. Leer texto en UTF-8 y comparar tras normalizar.

Ruta de la unidad

1. Modelo de datos de `String` e inmutabilidad.
2. Acceso, comparación y recorrido.
3. Búsqueda y conteo con operaciones básicas.
4. Transformaciones clave.
5. UTF-8 y normalización (NFC/NFD).
6. Lectura de archivos `.txt` y mini-casos ABP.
7. Cierre y preparación para evaluación.

String: inmutabilidad y costo de concatenar

- ▶ Cada concatenación crea un nuevo objeto (costo en bucles).
- ▶ **Recomendación:** usa `StringBuilder` para construir resultados.

StringBuilder en bucle

```
String repeat(char c, int n){  
    StringBuilder sb = new StringBuilder(n);  
    for(int i=0;i<n;i++) sb.append(c);  
    return sb.toString();  
}
```

Acceso y comparación

```
String s = "San Jose JR";  
char first = s.charAt(0);           // 'S'  
String team = s.substring(0, 3);    // "San"  
boolean ok = s.equals("San Jose JR"); // true  
boolean sameRef = (s == new String("San Jose JR")); // false  
int cmp = "abc".compareTo("abd");  // < 0
```

Búsqueda y conteo (operaciones básicas)

```
int count(String text, String pat){  
    int c=0, i=0, m=pat.length();  
    while(true){  
        int k = text.indexOf(pat, i);  
        if(k<0) break; c++; i = k + m; // avanzar por longitud del patrón  
    }  
    return c;  
}
```

Transformaciones comunes

```
String raw = "  Jose   Perez  ";  
String t1  = raw.strip();           // quitar bordes  
String t2  = t1.replace("  ", " "); // colapsar (simple)  
String u   = t2.toUpperCase(java.util.Locale.ROOT);  
String v   = t2.toLowerCase(java.util.Locale.ROOT);  
String w   = t2.replace("Perez","Pérez"); // literal (usa replace)
```

UTF-8 y normalización (NFC)

```
import java.text.Normalizer;
String nfc(String s){
    return Normalizer.normalize(s, Normalizer.Form.NFC);
}
boolean iguales(String a, String b){
    return nfc(a).equals(nfc(b));
}
```

Lectura de archivos (.txt) en UTF-8

```
import java.nio.file.*; import java.nio.charset.*;  
String leer(String ruta) throws Exception{  
    return Files.readString(Path.of(ruta), StandardCharsets.UTF_8);  
}
```


ABP 1 Limpieza de formularios

- ▶ Trim, colapso de espacios, capitalización por palabras.
- ▶ Reemplazos literales; comparar tras normalizar.

ABP 2 Validación simple de matrícula/CI

- ▶ Verificar longitud, prefijos y caracteres permitidos usando solo operaciones de caracteres.
- ▶ Mensajes claros de error; salida depurada.

ABP 3 Extracción de campos rotulados

- ▶ Localizar rótulos y extraer valores: `NOMBRE:`, `AULA:`, `HORA:` (separador `|` o fin de línea).
- ▶ Operaciones: `indexOf` para inicios y separadores, `substring` y `trim` para limpiar.
- ▶ Resultado: `NOMBRE=...`, `AULA=...`, `HORA=...`
- ▶ `StringBuilder` para armar el reporte final línea a línea.

ABP 4 Cifrado César básico

```
String cesar(String s, int k){  
    k=((k%26)+26)%26; StringBuilder r=new StringBuilder(s.length());  
    for(int i=0;i<s.length();i++){  
        char c=s.charAt(i);  
        if(c>='A'&&c<='Z') r.append((char)('A'+(c-'A'+k)%26));  
        else if(c>='a'&&c<='z') r.append((char)('a'+(c-'a'+k)%26));  
        else r.append(c);  
    } return r.toString();  
}
```

Errores comunes que evitaremos

- ▶ Usar `==` en lugar de `equals`.
- ▶ Concatenar en bucles con `+` (en vez de `StringBuilder`).
- ▶ Índices fuera de rango en `substring/charAt`.
- ▶ Asumir el encoding por defecto (no especificar UTF-8).
- ▶ Usar `replaceAll` pensando que es literal (interpreta patrones; no es literal).

Cierre y siguiente paso

- ▶ Practicar acceso, búsqueda y transformación en mini-ejercicios.
- ▶ Lectura de archivos y normalización para comparaciones robustas.
- ▶ Próxima sesión: consolidación + mini-casos ABP guiados.