Unidad: Cadenas — Paquete Docente

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

Curso: Introducción a la Programación · Modalidad: Presencial · Nivel: 1er semestre

Este documento reúne los resúmenes conceptuales y las guías operativas de las Sesiones 1 y 2.

Resumen rápido: equals vs == y por qué usar StringBuilder

== compara **referencias** (mismo objeto en memoria). **equals** compara **contenido** (mismas letras en el mismo orden).

```
String a = new String("Hola");
String b = new String("Hola");
System.out.println(a == b);       // false (objetos distintos)
System.out.println(a.equals(b));  // true  (contenido igual)
```

Buenas prácticas:

• Para comparar texto: **equals** (o **equalsIgnoreCase** si no importa mayúsc/minúscula).

• Evitar NullPointerException: usar **"OK".equals(estado)**.

StringBuilder: idea y uso

• **String es inmutable**: concatenar en bucles crea muchos objetos intermedios.
• **StringBuilder es mutable**: permite construir el resultado paso a paso sin copias.

```
// Patrón correcto con StringBuilder
StringBuilder sb = new StringBuilder(10000); // capacidad opcional
for (int i = 0; i < 10000; i++) {
    sb.append('*');
}
String s = sb.toString();
```

Operaciones útiles: **append**, **setLength(0)** (limpiar), **toString()** (resultado final).

Ficha del Docente — Sesión 1 (60′)

Propósito: activar interés con un caso realista y sentar fundamentos: inmutabilidad de String, uso de StringBuilder, acceso y comparación segura.

Materiales: slides de la unidad; mensaje.txt (César k=3); Demo.java; JDK 11+ (o lectura alternativa en JDK 8).

Cronograma sugerido:

• Caso + demo (frames 2–6): 20–22′

• Puente + objetivos + ruta (frames 7–9): 7–8′

• Núcleo técnico (frames 10–12): 14–15′

• Preguntas/margen: 10′

Guion por diapositiva (resumen):

• Portada: relevancia del texto en software; transición al caso.

• Caso de apertura: archivo “sucio”; objetivo: obtener Lugar/Clave.

• Visual del caso: texto → ruido/cifrado → tiempo (motivación).

• Reglas: trabajar con String/StringBuilder; respetar signos; usar UTF-8; (nota) normalizar si hay acentos extraños.

• Guía técnica mínima: leer, (opcional) normalizar, recorrer con charAt, construir con StringBuilder, extraer por índices.

• Mini-demostración (Java): descifrar César k=3 con lectura UTF-8.

• Puente: mapear el caso a acceso, búsqueda, transformación, eficiencia, UTF-8/normalización.

• Objetivos de aprendizaje: inmutabilidad, acceso seguro, conteo básico, transformaciones, lectura UTF-8.

• Ruta de la unidad: hoy fundamentos; próximas sesiones: búsqueda/transformación y luego UTF-8/archivos.

• String: inmutabilidad y costo de concatenar; por qué StringBuilder.

• StringBuilder en bucle: patrón correcto; pre-dimensionar si se puede.

• Acceso y comparación: charAt, substring, equals vs ==, compareTo.

Ficha del Docente — Sesión 2 (60′) — Solo demo en clase; ABP para casa

Objetivo: dominar búsqueda y transformación con operaciones básicas y ver validación posicional; dejar ABP para el viernes.

Operaciones de hoy: indexOf/lastIndexOf, startsWith/endsWith, equals, toLowerCase/UpperCase(Locale), trim/strip, replace literal, construcción con StringBuilder.

Diapositivas a usar hoy:

• Búsqueda y conteo (operaciones básicas)

• Transformaciones comunes

• Acceso y comparación (repaso rápido)

Código de ejemplo (fragmentos clave)

Conteo de ocurrencias (no superpuesto, sin distinguir mayúsculas):

```
int countCI(String text, String pat){
  String a = text.toLowerCase(java.util.Locale.ROOT);
  String b = pat.toLowerCase(java.util.Locale.ROOT);
  int c=0, i=0, m=b.length();
  if (m==0) return 0;
  while (true) {
    int k = a.indexOf(b, i);
    if (k < 0) break;
    c++; i = k + m;
  }
  return c;
}
```

Transformaciones: colapso de espacios + capitalización por palabra:

```
String collapseSpaces(String s){
  StringBuilder out = new StringBuilder(s.length());
  boolean prevSpace = false;
  for (int i=0;i<s.length();i++){
    char c = s.charAt(i);
    if (c==' '){
      if (!prevSpace) { out.append(' '); prevSpace = true; }
    } else { out.append(c); prevSpace = false; }
  }
  int start=0, end=out.length();
  while (start<end && out.charAt(start)==' ') start++;
  while (end>start && out.charAt(end-1)==' ') end--;
  return out.substring(start, end);
}

String capitalizeWordsSimple(String s){
  StringBuilder r = new StringBuilder(s.length());
  boolean newWord = true;
  for (int i=0;i<s.length();i++){
    char c = s.charAt(i);
    if (c==' '){ r.append(c); newWord = true; }
    else {
      r.append(newWord ? Character.toUpperCase(c)
                       : Character.toLowerCase(c));
      newWord = false;
    }
  }
  return r.toString();
}
```

Validación posicional y acumulación de mensajes con StringBuilder:

```
static boolean isUpperAZ(char c){ return c>='A' && c<='Z'; }
static boolean isDigit  (char c){ return c>='0' && c<='9'; }

static String validarMatriculaReporte(String s){
  StringBuilder rep = new StringBuilder();
  boolean ok = true;
  if (s.length()!=3+1+4+1+6){ ok=false; rep.append("Longitud incorrecta; "); }
  if (s.length()>=3 && !(isUpperAZ(s.charAt(0))&&isUpperAZ(s.charAt(1))&&isUpperAZ(s.charAt(2)))){
    ok=false; rep.append("Prefijo debe ser 3 letras mayúsculas; ");
  }
  if (s.length()>3 && s.charAt(3)!='-'){ ok=false; rep.append("Falta guion tras prefijo; "); }
  if (s.length()>=8 && !(isDigit(s.charAt(4))&&isDigit(s.charAt(5))&&isDigit(s.charAt(6))&&isDigit(s.charA
    ok=false; rep.append("Año debe tener 4 dígitos; "); }
  if (s.length()>8 && s.charAt(8)!='-'){ ok=false; rep.append("Falta guion tras año; "); }
  if (s.length()>=15){
    for (int i=9;i<15;i++) if(!isDigit(s.charAt(i))){ ok=false; rep.append("Secuencia final debe tener 6 d
  }
  if (ok) return "OK";
  int n = rep.length();
  if (n>=2 && rep.substring(n-2).equals("; ")) rep.setLength(n-2);
  return rep.toString();
}
```

Tarea para el viernes: ABP 1 (Limpieza), ABP 2 (Matrícula), ABP 3 (Extracción de campos).

Anexo: Extracción de campos rotulados (helper)

Localiza un rótulo (p.ej., “AULA: ”), toma desde el final del rótulo hasta el separador ‘|’ o fin, y recorta espacios:

```
String extractLabeled(String line, String label){
  int i = line.indexOf(label);
  if (i < 0) return "";
  i += label.length();
  int j = line.indexOf('|', i);
  if (j < 0) j = line.length();
  return line.substring(i, j).trim();
}
// Ejemplo de uso:
// String line = "NOMBRE: Ana Soto | AULA: B-203 | HORA: 10:00";
// System.out.println(extractLabeled(line, "AULA: ")); // "B-203"
```