Ficha del Docente — Sesión 1 (Unidad: Cadenas)

Curso: Introducción a la Programación
Unidad: Cadenas (String & StringBuilder)
Sesión: 1/4 (presencial, 60 minutos)
Formato de impresión: Carta (Letter, 8.5"×11")
Proyección: 16:9 (Beamer Metropolis)

Propósito de la sesión

Activar el interés con un caso realista y sentar fundamentos de trabajo con texto: inmutabilidad de `String`, uso eficiente de `StringBuilder`, acceso y comparación segura, preparando al estudiante para búsqueda/transformación y archivos en las siguientes sesiones.

Materiales y preparación

- **Slides:** *Beamer — Unidad: Cadenas (String & StringBuilder)*.
- **Archivos demo:** `mensaje.txt` (cifrado César k=3), `Demo.java` (lector + descifrado).
- Si el entorno es **Java 8**, usar lectura con `Files.readAllBytes(Paths.get(...))`.
- **Ambiente:** JDK 11+ (ideal) o 8 (alternativa), proyector 16:9, editor/terminal listos.
- **Verificación previa (checklist):**
- `mensaje.txt` está en el **directorio de ejecución**.
- Consola/terminal en UTF-8 (evitar mojibake).
- Compila y ejecuta `Demo.java` localmente.

Cronograma sugerido (60′)

- Caso + demo (slides 2–6): **20–22′**
- Puente + objetivos + ruta (slides 7–9): **7–8′**
- Núcleo técnico (slides 10–12): **14–15′**
- Preguntas y margen: **10′**

> **Plan B:** si el tiempo se ajusta, mueve **slide 12** (Acceso y comparación) a la sesión 2.

Guion por diapositiva (breve)

1) Portada (1′)

- **Objetivo:** encuadre y relevancia (todo software lee/procesa texto).
- **Mensaje clave:** hoy: acceso, búsqueda, transformación con `String` / `StringBuilder`.
- **Transición:** “Veamos un caso realista de texto en tránsito”.

2) Caso de apertura: *Mensaje crítico en tránsito* (3–4′)

- **Objetivo:** activar curiosidad y necesidad de cadenas.

- **Contar:** archivo “sucio”, tiempo limitado; salida esperada: `Lugar:` y `Clave:`.
- **Detonadora:** “¿Qué operaciones textuales necesitamos **sin** arrays ni regex?”
- **Indicador de comprensión:** estudiantes mencionan `indexOf`, `substring`, `trim`, `toLowerCase`, `replace` literal.

3) Visual del caso (1–2′)

- **Objetivo:** fijar mentalmente: texto → cifrado/ruido → tiempo.
- **Remarcar:** se resuelve con **acceso, búsqueda, transformación**.

4) Reglas y restricciones (para pensar) (3′)

- **Objetivo:** acotar técnicamente.
- **Puntos:** sin regex ni arrays; `String` / `StringBuilder`; respetar signos; **UTF-8**; (nota) normalizar si hay acentos “raros”.
- **Pregunta:** “¿Por qué `replaceAll` no es apropiado aquí?” (usa regex).
- **Riesgo común:** olvidar especificar UTF-8 en lectura.

5) Guía técnica mínima (4′)

- **Objetivo:** plan paso a paso.
- **Pasos:** leer UTF-8 → (opcional) normalizar → recorrer con `charAt` → construir con `StringBuilder` → extraer por índices `Lugar:` / `Clave:`.
- **Tip:** no concatenar con `+` en bucles.

6) Mini-demostración (Java) (7–8′)

- **Objetivo:** evidenciar solución con operaciones básicas.
- **Acción:** ejecutar con `mensaje.txt` (k=3), mostrar salida limpia.
- **Variación:** cambiar `k` a `-2` para ver el efecto.
- **Compatibilidad:** Java 8 → `Files.readAllBytes` + `new String(..., UTF_8)`.
- **Cierre:** “Sin arrays ni regex, obtuvimos la info crítica”.

7) Puente con la unidad (2–3′)

- **Objetivo:** mapear el caso a contenidos próximos.
- **Mapa:** acceso (`length`, `charAt`, `substring`) → búsqueda (`indexOf`) → transformación (`trim`, `lower/upper`, `replace` literal) → eficiencia (`StringBuilder`) → UTF-8/ normalización.
- **Pregunta:** “¿Qué parte del caso correspondió a cada operación?”

8) Objetivos de aprendizaje (2′)

- **Objetivo:** alinear expectativas evaluables.
- **Enfatizar:** inmutabilidad, acceso seguro, conteo sin regex, transformaciones clave, lectura UTF-8 + comparación tras normalizar.

9) Ruta de la unidad (con overlays) (2–3′)

- **Objetivo:** roadmap claro.
- **Hoy:** fundamentos (`String`, `StringBuilder`, acceso/comparación).
- **Próximo:** búsqueda/transformación y luego Unicode/archivos.

- **Tip:** señalar “dónde estamos”.

10) String: inmutabilidad y costo de concatenar (5′)

- **Objetivo:** justificar `StringBuilder`.
- **Explicar:** cada `+` crea objeto; en bucles escala mal.
- **Ejemplo verbal:** concatenar 1000 caracteres vs `StringBuilder(1000)`.

11) `StringBuilder` en bucle (4′)

- **Objetivo:** patrón de construcción incremental.
- **Código:** pre-dimensionar cuando sea posible; `append` en bucle; `toString()`.
- **Mini-ejercicio oral:** “¿Cómo imprimen n asteriscos?”

12) Acceso y comparación (5′)

- **Objetivo:** operaciones base y comparaciones correctas.
- **Código:** `charAt`, `substring`, `equals` vs `==`, `compareTo` (orden lexicográfico).
- **Errores a evitar:** índices fuera de rango; `==` en vez de `equals`.

Micro-chequeos y participación

- **Check 1 (slide 4):** ¿Por qué `replaceAll` no aplica? → Porque usa regex y no queremos patrones.
- **Check 2 (slide 6):** ¿Qué cambia si `k=-2`? → Desplaza en sentido opuesto; observar mayúsc/minúsc.
- **Check 3 (slide 12):** `s == "texto"` vs `s.equals("texto")` → referencia vs contenido.

Salida de la sesión

- Estudiantes pueden: explicar inmutabilidad, usar `StringBuilder` en bucle, aplicar `indexOf`/`substring`, y describir por qué se especifica **UTF-8**.

Preparación para Sesión 2

- Dejar tarea corta: contar ocurrencias de una palabra **sin regex** y normalizar espacios (trim + colapso simple) en una línea de entrada.