Guía de Actividades Práctico-Experimentales Nro. 006

1. Datos Generales

Asignatura	Estructura de datos
Ciclo	3 A
Unidad	2
Resultado de aprendizaje de la unidad	Aplica los métodos de ordenación y búsqueda en la resolución de problemas, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
Título de la Práctica	Ordenación básica en Java: Burbuja, Selección e Inserción
Nombre del Docente	Andrés Roberto Navas Castellanos
Fecha	Jueves 20 de noviembre Viernes 21 de noviembre
Horario	07h30 – 10h30 07h30 – 09h30
Lugar	Aula
Tiempo planificado en el Sílabo	5 horas

2. Objetivo(s) de la Práctica:

- Ejecutar y analizar comparativamente los algoritmos de Burbuja, Selección e Inserción sobre casos de prueba, para determinar cuándo conviene cada uno en función de tamaño, grado de orden y duplicados.

3. Materiales y reactivos:

- Guía de pruebas con datasets y salidas esperadas.

4. Equipos y herramientas

- JDK OpenJDK (obligatorio).
- IDE: Visual Studio Code (extensión “Extension Pack for Java”) o IntelliJ IDEA Community.
- Sistema de control de versiones: Git; repositorio en GitHub.
- EVA/Moodle institucional: para entrega de evidencias.
- Herramientas de documentación: README Markdown, editor ofimático (Google Docs/LibreOffice/Word).

5. Procedimiento / Metodología

Enfoque metodológico: ABPr (Aprendizaje Basado en Proyectos).

Inicio

- Presentación del objetivo comparativo y criterios de éxito.
- Formación de equipos (3–4) y revisión de la rúbrica.
- Creación de repo Git.
- Lineamientos de uso responsable de IA.

Desarrollo

- Paso 1. Instrumentación (obligatorio)
  - Añade contadores a tus algoritmos:
    - comparisons++ al comparar dos claves,
    - swaps++ al intercambiar posiciones.
  - Mide tiempo con `System.nanoTime()` sin imprimir durante la medición (las trazas distorsionan).
  - Ejecuta R repeticiones por caso (sug.: R=10), descarta las 3 primeras (calentamiento/JIT) y reporta la mediana de tiempo.
  - Aísla IO: carga CSV fuera de la medición; mide sólo el ordenamiento del array en memoria.
- Paso 2. Casos de prueba
  - Define clave de orden (p. ej., `fechaHora` en `citas`, `apellido` en `pacientes`, `stock` en `inventario`).
  - Convierte a array de la clave (o a registros con `Comparable` por clave).
  - Ejecuta: Insertion, Selection, Bubble (con “corte temprano” en Burbuja).
  - Registra: n, %casi-ordenado, %duplicados, comparisons, swaps, tiempo(ns) (mediana de R-3 corridas).
- Paso 3. Análisis
  - Tablas comparativas por caso (n, orden, duplicados) y gráficos (tiempo vs. n; tiempo vs. %casi-ordenado).
  - Matriz de recomendación (reglas prácticas):
    - Casi ordenado + n pequeño/medio → Inserción gana (menos movimientos).
    - Muchos duplicados → Inserción tiende a mantener estabilidad útil; Selección hace n(n−1)/2 comparaciones siempre, con pocos swaps.
    - Inverso o aleatorio (n pequeño/educativo) → cualquiera, pero Burbuja penaliza; Selección constante en comparaciones; Inserción peor en inverso pero mejor si detecta localmente orden.

Cierre

- Discusión guiada: ¿Cuándo conviene cada uno? ¿Qué sesgos introdujo la medición?
- Completar README e informe con evidencias y la matriz de recomendación.

6. Resultados esperados:

- Tabla por dataset: `n, tipo (aleat/casi-ord/dup/inverso), algoritmo, comparisons, swaps, tiempo_mediana(ns)`.
- Gráficos (opcional): barras o líneas para tiempo y comparaciones.
- Matriz de recomendación (texto/tabla): "si casi ordenado y n≤500 → Inserción", "si minimizar swaps → Selección", etc.
- Capturas/Logs de ejecución (sin trazas durante medición).
- Código con instrumentación y scripts de generación de datasets (si aplica).

7. Preguntas de Control:

- ¿Por qué imprimir trazas durante la medición distorsiona los tiempos?
- Explica por qué Selección tiene comparaciones ~ n(n−1)/2 sin importar el orden inicial.
- ¿Por qué Inserción es competitivo en datos casi ordenados?
- ¿Qué papel juegan los duplicados en la estabilidad del resultado?
- ¿Por qué Burbuja con corte temprano mejora en "casi ordenado" pero no en "inverso"?

8. Evaluación

Criterio	4 – Excelente	3 – Bueno	2 – Básico	1 – Insuficiente	Pts
Instrumentación (contadores + tiempo)	Corrección y limpieza; medición sin IO/impresiones	Menor detalle	Parcial	No funcional	**2.5**
Diseño experimental	R≥10, descarta 3 corridas, mediana; casos variados	Algún ajuste menor	Parcial	Inadecuado	**2.0**
Ejecución y datos	Tablas completas por dataset	Tablas con huecos	Datos escasos	Sin datos	**2.0**
Análisis y matriz	Conclusiones claras y justificadas	Aceptables	Superficiales	Ausentes	**2.5**
Entrega y código	README/Informe claros; código limpio	Aceptable	Pobre	Deficiente	**1.0**

9. Bibliografía

[1] OpenDSA Project, "Sorting and Searching Modules," Virginia Tech, 2021–2024 (REA con visualizaciones y ejercicios).

[2] P. W. Bible and L. Moser, An Open Guide to Data Structures and Algorithms. PALNI Open Press, 2023.

[3] Oracle, "Java SE 17–21 Documentation: `Arrays`, Collections, and I/O (`java.nio.file`), and benchmarking notes," 2021–2025.

[4] OpenJDK, "JMH – Java Microbenchmark Harness: Samples and Guidance," 2020–2025 (guía práctica de mediciones reproducibles).

10. Elaboración y Aprobación

Elaborado por	Andrés R Navas Castellanos **Docente**	Firmado electrónicamente por: ANDRES ROBERTO NAVAS CASTELLANOS Validar únicamente con FirmaEC
Revisado por **Solo si es realizado en laboratorios**	Luis Sinche **Técnico Docente**	No Aplica
Aprobado por	Edison L Coronel Romero **Director de Carrera**	