Guía de Actividades Práctico-Experimentales Nro. 006

1. Datos Generales

Integrantes	Miguel Alejandro Armas Ordóñez Santiago Alexander Villamagua Jimenez
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:

Aleatorio 100

```
--- TEST 1: Citas (100 registros, Aleatorio) ---
Registros cargados: 100
Iniciando medición estadística (R=10, descartando 3 de calentamiento)...

Resultados (MEDIANA) para: Citas Aleatorias
+-----------+-------------+------------+-------------+-------------+
----------------------Bubble sort--------------------
Comparaciones:                  4940
Movimientos:                    8097
Tiempo:                         1052100
----------------------Selection sort-----------------
Comparaciones:                  4950
Movimientos:                    273
Tiempo:                         773600
----------------------Insertion sort-----------------
Comparaciones:                  2798
Movimientos:                    2897
Tiempo:                         333300
+-----------+-------------+------------+-------------+-------------+
```

Dataset	Algoritmo	Comparaciones	Movimientos	Tiempo
Citas	Burbuja	4940	8097	1052100
Citas	Selección	4950	273	773600
Citas	Inserción	2798	2897	333300

Casi Ordenado 100

```
--- TEST 2: Citas (100 registros, Casi Ordenado) ---
Registros cargados: 100
Iniciando medición estadística (R=10, descartando 3 de calentamiento)...

Resultados (MEDIANA) para: Citas Casi Ordenadas
+-----------+-------------+------------+-------------+-------------+
----------------------Bubble sort--------------------
Comparaciones:                  3519
Movimientos:                    711
Tiempo:                         165900
----------------------Selection sort-----------------
Comparaciones:                  4950
Movimientos:                    15
Tiempo:                         123700
----------------------Insertion sort-----------------
Comparaciones:                  336
Movimientos:                    435
Tiempo:                         55400
+-----------+-------------+------------+-------------+-------------+
```

Dataset	Algoritmo	Comparaciones	Movimientos	Tiempo
Citas	Burbuja	3519	711	165900
Citas	Selección	4950	15	123700
Citas	Inserción	336	435	55400

Datos Repetidos 500

```
--- TEST 3: Pacientes (500 registros, Repetidos) ---
Registros cargados: 500
Iniciando medición estadística (R=10, descartando 3 de calentamiento)...

Resultados (MEDIANA) para: Pacientes
+-----------+-------------+------------+-------------+-------------+
----------------------Bubble sort--------------------
Comparaciones:                  123760
Movimientos:                    178176
Tiempo:                         3378600
----------------------Selection sort-----------------
Comparaciones:                  124750
Movimientos:                    1389
Tiempo:                         1428700
----------------------Insertion sort-----------------
Comparaciones:                  59890
Movimientos:                    60390
Tiempo:                         1521500
+-----------+-------------+------------+-------------+-------------+
```

Dataset	Algoritmo	Comparaciones	Movimientos	Tiempo
Pacientes	Burbuja	123760	178176	3378600
Pacientes	Selección	124750	1389	1428700
Pacientes	Inserción	59890	60390	1521500

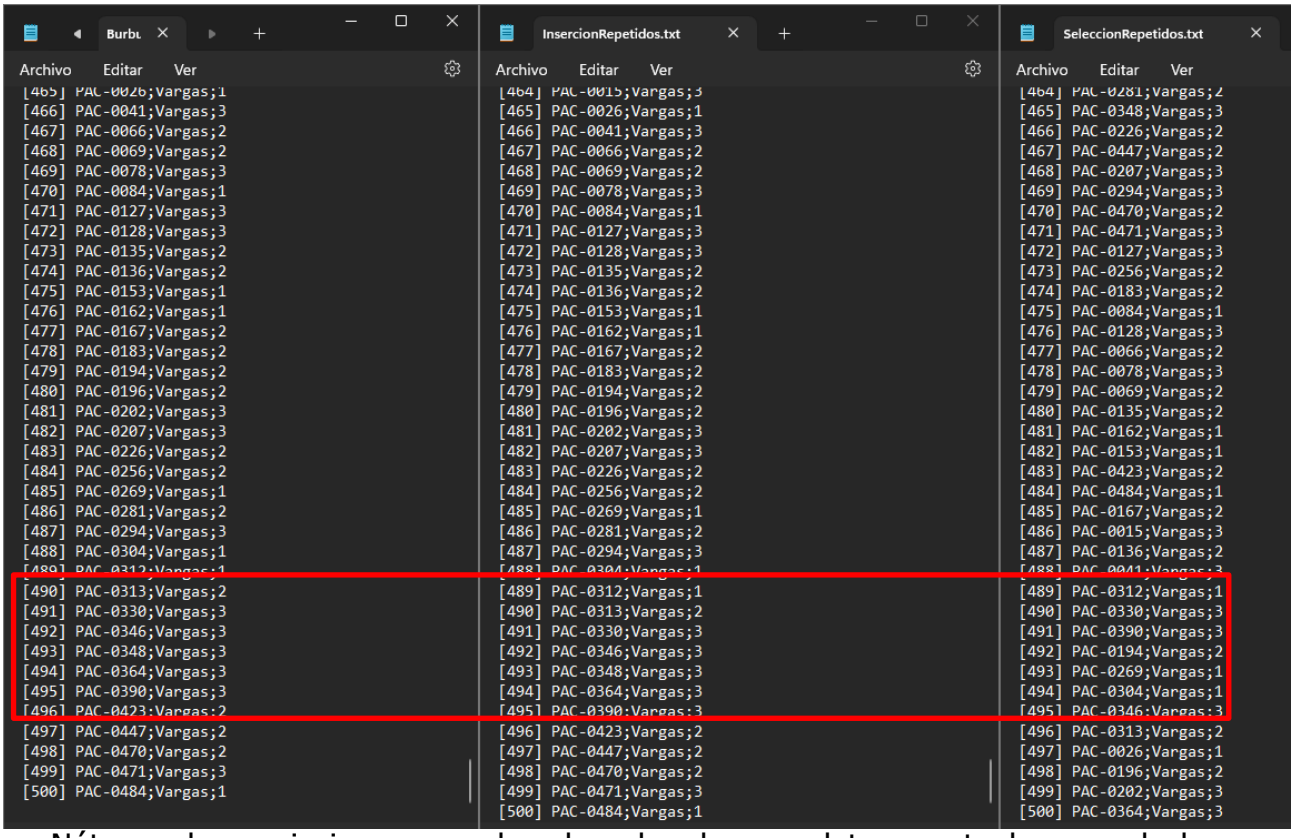
Inverso 500

```
--- TEST 4: Inventario (500 registros, Inverso) ---
Registros cargados: 500
Iniciando medición estadística (R=10, descartando 3 de calentamiento)...

Resultados (MEDIANA) para: Inventario Inverso
+-----------+-------------+------------+-------------+-------------+
----------------------Bubble sort--------------------
Comparaciones:                  124750
Movimientos:                    374250
Tiempo:                         2836900
----------------------Selection sort-----------------
Comparaciones:                  124750
Movimientos:                    750
Tiempo:                         1167300
----------------------Insertion sort-----------------
Comparaciones:                  124750
Movimientos:                    125748
Tiempo:                         2829400
+-----------+-------------+------------+-------------+-------------+
```

Dataset	Algoritmo	Comparaciones	Movimientos	Tiempo
Inventario	Burbuja	124750	374250	2836900
Inventario	Selección	124750	750	1167300
Inventario	Inserción	124750	125748	2829400

Estabilidad



Nótense las variaciones en el orden de algunos datos puntuales en el de selección.

Escenario A: N=100 (Aleatorio vs. Casi Ordenado)

Algoritmo	Tiempo Aleatorio (ns)	Tiempo Casi Ord. (ns)	Mejora (Speedup)	Movimientos (Casi Ord.)
Inserción	333,300	**55,400**	**6.0x (83% más rápido)**	435
Selección	773,600	123,700	6.2x	15 (Mínimo)
Burbuja	1,052,100	165,900	6.3x	711

Aunque todos mejoran en el escenario "Casi Ordenado" debido a que son conjuntos pequeños (N=100), Inserción es el ganador absoluto en velocidad, siendo 3 veces más rápido que Burbuja. Selección destaca por realizar la menor cantidad de movimientos (writes) en memoria.

Escenario A: N=100 (Aleatorio vs. Casi Ordenado)

Algoritmo	Tiempo Repetidos (ns)	Tiempo Inverso (ns)	Comparaciones (Inverso)	Movimientos (Inverso)
Selección	1,428,700	1,167,300	124,750	**750**
Inserción	1,521,500	2,829,400	124,750	125,748
Burbuja	3,378,600	2,836,900	124,750	374,250

En el caso Inverso (Peor Caso), Selección supera drásticamente a los demás.

- Inserción y Burbuja se degradan a comportamientos cuadráticos pesados (2.8 ms).
- Selección se mantiene constante (~1.1 ms) porque realiza solo N movimientos (750 en total), comparado con los 374,000 movimientos de Burbuja.

COMPARATIVA

Basado en la recolección empírica de datos se llegó a varias conclusiones.

Escenario / Condición	**Algoritmo Ganador**	**Justificación con Datos**
Datos Casi Ordenados (Swaps < 5%)	Inserción	Tiempo mínimo récord (55400 ns). Inserción detecta el orden localmente y realiza O(n) comparaciones.
Escritura Costosa (Minimizar movimientos)	Selección	Siempre realiza O(n) movimientos. En el caso Inverso, hizo 750 movimientos vs 374,250 de Burbuja.
Requerimiento de Estabilidad (Ej. Pacientes)	Inserción	Mantiene el orden relativo de claves iguales (apellidos). Burbuja también es estable, pero 2 veces más lento en este test (3.3ms vs 1.5ms).
Datos Inversos (Peor Caso)	Selección	Fue el único algoritmo que no duplicó su tiempo de ejecución. Inserción colapsó al nivel de Burbuja.
Propósito Educativo	Burbuja	Aunque tuvo el peor rendimiento general (llegando a 1M ns en N=100), su lógica de "burbujear" el máximo es fácil de aprender y entender.

Discusión Guiada

¿Cuándo conviene cada uno?

1. **Inserción:** Es el algoritmo "por defecto" para listas pequeñas o cuando sabemos que los datos ya tienen cierto orden (ej. añadir una cita nueva a una agenda ya ordenada).
2. **Selección:** Es ideal cuando escribir en memoria es costoso a nivel operacional ya que reduce el desgaste de escritura al mínimo teórico con los pocos cambios que realiza.
3. **Burbuja:** Prácticamente nunca se usaría en casos complejos o en el ámbito profesional debido a su ineficiencia en movimientos y comparaciones, solo para fines educativos y aprender conceptos básicos.

¿Qué sesgos introdujo la medición?

- **JIT (Just-In-Time Compiler):** Si no hubiéramos descartado las primeras 3 corridas, los tiempos iniciales habrían sido mucho más altos mientras Java optimizaba el código máquina.
- **Garbage Collection:** Aunque llamamos a System.gc(), la gestión de memoria automática de Java puede pausar la ejecución aleatoriamente, afectando la desviación estándar.
- **Granularidad del Reloj:** System.nanoTime() es preciso, pero en ejecuciones muy rápidas (como los 55,400 ns de Inserción), el ruido del sistema operativo influye porcentualmente más.

7. Preguntas de Control:

- **¿Por qué imprimir trazas durante la medición distorsiona los tiempos?**

La operación de Entrada/Salida (I/O) hacia la consola (`System.out.println`) es extremadamente lenta comparada con operaciones de CPU/RAM. Imprimir una sola línea puede tardar milisegundos, mientras que comparar dos enteros tarda nanosegundos. Si se mide el tiempo con impresiones, se está midiendo la velocidad de la consola, no la eficiencia del algoritmo.

- **Explica por qué Selección tiene comparaciones ~ n(n−1)/2 sin importar el orden inicial.**

El algoritmo de Selección busca el elemento más pequeño del sub-arreglo no ordenado en cada iteración. Para encontrar el mínimo seguro, debe revisar todos los elementos restantes, no puede detenerse antes.

- Fórmula: $n(n-1)/2$ se aproxima a O(n^2).
- Tanto en "Aleatorio" como en "Casi Ordenado" (N=100), Selección hizo exactamente 4950 comparaciones.

- **¿Por qué Inserción es competitivo en datos casi ordenados?**

Inserción toma un elemento y lo compara hacia atrás. Si el elemento anterior es menor, se detiene inmediatamente, asumiendo que el resto hacia atrás ya está ordenado.

- En datos ordenados, realiza solo 1 comparación por elemento.
- En los datos de prueba se notó esto, ya que bajó de 2798 comparaciones (Aleatorio) a 336, lo que explica su alta velocidad con los datos casi ordenados.

- **¿Qué papel juegan los duplicados en la estabilidad del resultado?**

Un algoritmo es estable si mantiene el orden original de registros con claves iguales (ej. dos pacientes "Castillo", uno con prioridad 1 y otro con 2).

- Inserción y Burbuja son estables cuando consideran “>” y no “>=”
- Selección es inestable por naturaleza ya que al hacer swaps largos puede saltar elementos iguales cambiando su orden.

- **¿Por qué Burbuja con corte temprano mejora en "casi ordenado" pero no en "inverso"?**

Casi ordenado: La bandera `swapped` permite detectar pasadas donde no se hicieron cambios, permitiendo acabar con las iteraciones antes en caso de no haberse hecho cambios en la última pasada.

Inverso: En un arreglo inverso, el elemento más pequeño está al final y debe subir hasta el principio. Esto obliga a Burbuja a realizar intercambios en **TODAS** las pasadas hasta el final, haciendo inútil el corte temprano.

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**	