**INFORME DE LABORATORIO**

**Avance Proyecto Final**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN BÁSICA** | | | | | | |
| **ASIGNATURA:** | *Fundamentos de la Programación 2* | | | | | |
| **TÍTULO DE LA PRÁCTICA:** | *HashMaps* | | | | | |
| **NÚMERO DE PRÁCTICA:** | *08* | **AÑO LECTIVO:** | *2024-B* | **NRO. SEMESTRE:** | | *II* |
| **FECHA DE PRESENTACIÓN** | *29/11/2024* | **HORA DE PRESENTACIÓN** | *17:20:00* | | | |
| **INTEGRANTE (s)**  *Subia Huaicane Edson Fabricio* | | | | **NOTA (0-20)** | | *Nota colocada por el docente* |
| **DOCENTE(s):**  *Lino José Pinto Oppe* | | | | | | |
|  | | | | |  | |
| **RESULTADOS Y PRUEBAS** | | | | | | |
| * **EJERCICIOS RESUELTOS:**   Actividad VIDEOJUEGO de SOLDADOS:   * Cree un Proyecto llamado Laboratorio7 * Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores. * Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero). * El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada. * Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados y sus puntos de vida (usar caracteres como | \_ y otros y distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.   En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: <https://github.com/Q3son/Videojuego_Soldados.git>  **Mis COMMITS:**   * Este es el primer commit destacable que realicé, acomodé el método para Inicializar el juego y logré que este mismo reconociese la creación de ejércitos mediante HashMaps:      * **Seguidamente, controlé la creación de los mismos en el método de Inicialización:** * **Se actualizaron los métodos de ordenamiento, asimismo, la creación y mejora del tablero.**      * **Para esta versión, implementé un nuevo método de ordenamiento (MergeSort), para que funcione acorde a nuestra creación de HashMaps.**      * **Para la versión final definitiva, logré que el programa funcione de forma iterativa y a su vez, mejoré el main signficativamente.**   **En la siguiente sección mostraré el código fuente y ejecución de la versión final de mi código fuente del programa, trabajado en Visual Studio, en cada captura de pantalla se visualizará el buen funcionamiento de los nuevos métodos adicionados y fundamentados correctamente. (El código fuente se visualiza mucho mejor en mi repositorio)**    ***EJECUCIÓN DEL PROGRAMA (v6.0.1): “JUEGO DE NAVE*** **EBAS**  ***¿Con qué valores comprobaste que tu práctica estuviera correcta?* Comprobé la práctica utilizando valores aleatorios al generar soldados para dos ejércitos en HashMaps. Asigné nombres autogenerados como "Soldado0X1", "Soldado1X1", variando aleatoriamente la fila y columna de cada soldado y asegurando que no se repitieran las posiciones en el tablero. También generé valores de nivel de vida aleatorios entre 1 y 5 para verificar que el tablero se mostrara correctamente y que cada ejército mantuviera sus soldados en posiciones distintas. Para probar el ranking, utilicé los métodos de ordenamiento de selección y burbuja en soldados de cada ejército.**  ***¿Qué resultado esperabas obtener para cada valor de entrada? Esperaba que el programa tuviera el siguiente comportamiento:***   * **Al ordenar soldados por nivel de vida (Algoritmo MergeSort): Los soldados de cada ejército se ordenarían correctamente de acuerdo con sus niveles de vida, mostrando el orden descendente esperado.** * **Al mostrar el soldado con mayor nivel de vida por ejército: Se esperaba que se identificara correctamente el soldado con el mayor nivel de vida en cada ejército.** * **Al calcular el promedio y el total de puntos de vida por ejército: El programa mostraría los valores adecuados para el promedio de vida y el total de vida de cada ejército.** * **Al determinar el ejército ganador: Se esperaba que el ejército con mayor total de puntos de vida fuera declarado como ganador.**   ***¿Qué valor o comportamiento obtuviste para cada valor de entrada? Los resultados obtenidos fueron los esperados:***   * **Al ordenar soldados por nivel de vida: Los soldados de ambos ejércitos se mostraron ordenados correctamente en función de sus niveles de vida, validando la eficacia del algoritmo de selección y burbuja para el ranking.** * **Al mostrar el soldado con mayor nivel de vida: El programa identificó correctamente el soldado con el mayor nivel de vida en cada ejército.** * **Al calcular el promedio y total de puntos de vida por ejército: Los resultados mostraron los valores esperados, confirmando que los métodos de cálculo se implementaron correctamente.** * **Al determinar el ejército ganador: El programa identificó correctamente al ejército con el mayor total de puntos de vida, cumpliendo con las métricas definidas para decidir el ganador de la batalla.** | | | | | | |
| 1. **CUESTIONARIO:**   Los datos más importantes que consideré para realizar el código del juego de Soldados, especialmente en relación con las nuevas funcionalidades y requisitos, fueron los siguientes:   1. **Atributos del Soldado**: Cada soldado cuenta con atributos clave como nombre, fila, columna y nivel de vida. Estos atributos son fundamentales para definir las características de cada soldado en el tablero. El nivel de vida es especialmente crítico, ya que determina la resistencia de cada soldado en la batalla y, por ende, influye en la estrategia del juego. 2. **ArrayList bidimensional**: Utilizar un arreglo bidimensional para almacenar los objetos de tipo Soldado es esencial. Este arreglo simula el tablero de juego, permitiendo que los soldados se ubiquen en posiciones específicas y facilitando la gestión de sus interacciones durante la partida. Además, esta estructura es crucial para aplicar los métodos de ordenamiento que permiten organizar a los soldados de acuerdo a su nivel de vida o posición. 3. **Generación aleatoria de soldados**: La creación de soldados con valores aleatorios, como el nivel de vida y las posiciones, asegura un entorno dinámico y desafiante para el juego. Esto incluye garantizar que no haya dos soldados en la misma posición del tablero, lo que agrega un nivel de estrategia adicional al juego. 4. **Métodos de ordenamiento (MergeSort)**: La implementación de métodos de ordenamiento es crucial para organizar a los soldados según diferentes criterios: 5. **Cálculo de promedios y totales**: Los métodos que calculan el promedio y total de puntos de vida por ejército son esenciales para evaluar la fuerza general de cada ejército. Esto ayuda a determinar quién ganará la batalla, basándose en el total de vida de los soldados, y aporta a la lógica del juego. 6. **Determinación del ejército ganador**: Definir qué ejército ganará la batalla es fundamental para la dinámica del juego. Se basa en la métrica del total de puntos de vida, permitiendo a los jugadores evaluar el rendimiento de sus ejércitos de manera clara y concisa. 7. **Interacción con el usuario**: Todos estos elementos mejoran la experiencia del usuario, al permitirle ver los soldados organizados y calcular los resultados de manera eficiente. Los jugadores pueden evaluar fácilmente la capacidad de su ejército y tomar decisiones informadas, enriqueciendo así la jugabilidad. | | | | | | |
|  | | | | | | |
| **CONCLUSIONES** | | | | | | |
| *Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.*  La implementación del juego de Soldados ha sido exitosa tanto en su estructura como en la interacción que ofrece al usuario. La incorporación de algoritmos de ordenamiento ha permitido gestionar los atributos de los soldados, como el nivel de vida y el nombre, de manera eficiente. Al ordenar por nivel de vida utilizando el método de selección, el jugador puede visualizar fácilmente qué soldados tienen mayor resistencia, lo que facilita la toma de decisiones estratégicas. Por otro lado, el ordenamiento por nombre mediante el método de burbuja mejora la gestión alfabética de los soldados, optimizando la experiencia de búsqueda y análisis. | | | | | | |
|  | | | | | | |
| **METODOLOGÍA DE TRABAJO** | | | | | | |
| *Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.*   1. **Comprensión del problema:** En esta etapa, revisé cada una de las actividades propuestas, identificando cuidadosamente las restricciones y los objetivos a alcanzar. 2. **Diseño del algoritmo:** Planifiqué la secuencia lógica necesaria para implementar la solución, aplicando los conocimientos adquiridos en Fundamentos de Programación I y II. 3. **Codificación:** Procedí a implementar los programas solicitados, asegurándome de utilizar correctamente los arreglos y métodos. 4. **Pruebas:** Realicé pruebas adicionales para verificar que el código funcionara de manera correcta con diferentes casos de prueba. | | | | | | |
|  | | | | | | |
| **REFERENCIAS Y BIBLIOGRAFÍA** | | | | | | |
| *Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE*  *M. W. Aedo López, Fundamentos de programación I: Java Básico, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, jul. 2019. ISBN: 978-612-4337-55-0. 116 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 21 x 29.7 cm*  [*https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo\_Emanuel\_LABORATORIO\_01.pdf*](https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf)  *https://github.com/Q3son/Videojuego\_Soldados.git* | | | | | | |

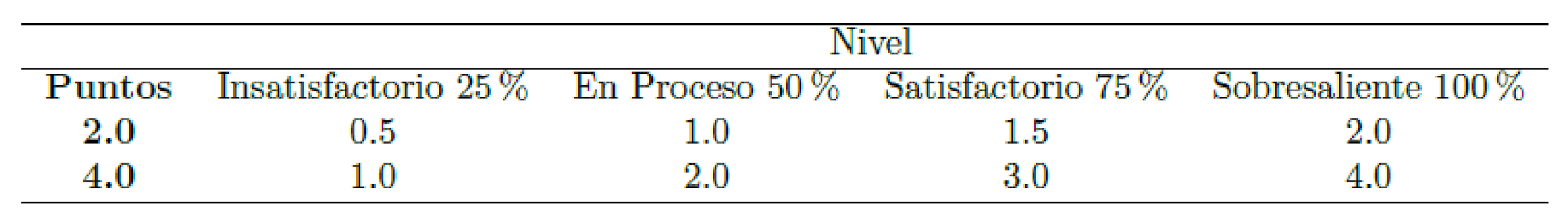
**RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN**

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo con la siguiente tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Contenido y demostración** | | **Puntos** | **Checklist** | **Estudiante** | **Profesor** |
| 1. GitHub | Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar. | 2 | X | 2 |  |
| 2. Commits | Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación). | 4 | X | 4 |  |
| 3. Código fuente | Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones. | 2 | X | 2 |  |
| 4. Ejecución | Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente. | 2 | X | 2 |  |
| 5. Pregunta | Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación). | 2 | X | 2 |  |
| 6. Fechas | Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos. | 2 | X | 2 |  |
| 7. Ortografía | El documento no muestra errores ortográficos. | 2 | X | 2 |  |
| 8. Madurez | El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación). | 4 | X | 3 |  |
| TOTAL | | 20 | 8 | 19 |  |

Tabla 1: Niveles de desempeño