Departamento de Procesos y Sistemas.
Estructuras de Datos
Profesor Alejandro Koury

UNIVERSIDAD METROPOLITANA

Proyecto 1 (15%)

Usted ha sido encargado de diseñar y programar un juego de buscaminas. Para ello deberá utilizar el paradigma de Programación Orientada a Objetos (OOP) y el lenguaje Java. En este proyecto deberá utilizar adecuadamente todos los conceptos estudiados hasta el momento en la materia.

Sobre el juego:

Buscaminas es un juego de estrategia, con elementos de azar, para un solo jugador. El objetivo de este es vaciar un tablero rectangular sin activar ninguna de las minas escondidas que se encuentran en este. En un principio, todas las casillas del tablero se encuentran ocultas. Al seleccionar una posición, esta se muestra y puede ocurrir una de dos cosas:

1. Si el contenido de la casilla es una mina, esta explota y el jugador pierde el juego.
2. Si la casilla está vacía, se muestra en ella un número que indica cuántas minas hay en las 9 casillas colindantes.
    a. Si el número es 0, las 9 casillas colindantes son seleccionadas automáticamente, hasta ubicar los nuevos bordes.

Para facilitar su tarea, el jugador dispone de la habilidad de marcar casillas. De esta forma puede ir registrando qué cree que se encuentra en una casilla oculta. Las marcas que puede hacer son: una bandera, que indica una mina en esa posición; o un interrogante, que indica que el contenido de esa posición no puede ser determinado.

El juego continúa hasta que el jugador selecciona una mina o selecciona todas las casillas que no contienen una mina. Solo en el segundo caso se dice que el jugador ganó la partida. Al finalizar la partida, el programa debe regresar al usuario al menú principal, donde este puede iniciar una nueva partida.

Sobre las casillas:

Existen únicamente dos tipos de casillas, siendo la principal diferencia entre estas cómo actúan al ser seleccionadas. Cuando se selecciona una casilla con mina, esta se muestra como una mina y luego explota, mostrando las demás minas y terminando el juego. Cuando se selecciona una casilla sin mina, esta muestra el número de minas alrededor de ella y, en caso de que no tenga ninguna, selecciona de

forma automática las 9 casillas a su alrededor, repitiendo el proceso hasta conseguir casillas con minas alrededor. Las casillas tienen en común la forma en que se muestran cuando están ocultas, la capacidad de tener una marca del jugador (solo si está oculta) y la capacidad de ser reveladas.

Sobre el tablero:

El tablero está compuesto de casillas ordenadas en dos dimensiones. Al mostrarse el tablero se debe distinguir claramente el sistema de coordenadas, las casillas ocultas y el número de minas alrededor de las casillas abiertas, así como las marcas que haga el jugador. De igual forma se debe mostrar la cantidad de minas que se tienen en el tablero.

Sobre el jugador:

El programa debe pedirle el nombre al usuario al iniciar la ejecución y mantenerlo guardado por toda la sesión. El jugador realiza todas las entradas del programa, seleccionando revelar o marcar casillas específicas. Para ello debe preguntársele qué acción desea realizar y las coordenadas de la casilla afectada.

Sobre la dificultad:

La disposición de las minas en el tablero se deberá realizar de manera aleatoria. El programa solicitará al usuario que elija un nivel de dificultad antes de iniciar el juego, existen 4 niveles de dificultad:

Fácil: Tablero de 10x10 con 10 minas.

Media: Tablero de 15x15 con 40 minas.

Difícil: Tablero de 22x22 con 100 minas.

Personalizada: El usuario podrá determinar la cantidad de filas y columnas, al igual que la cantidad de minas en el tablero.

Notas:

- El proyecto puede realizarse de forma individual o en parejas.
- Debe entregar tanto el diagrama de clases de su aplicación, como el zip que contenga su proyecto a los correos jose.quevedo@correo.unimet.edu.ve y akoury@unimet.edu.ve antes de las 11:59 pm del 18 de febrero de 2018.
- El código de su proyecto debe estar correctamente identificado, documentado y comentado.

Criterios de evaluación

Todo código dentro del main debe poder ser justificado. Deben utilizar clases, herencia, polimorfismo, encapsulación, propiedades y metodos dinamicos y estaticos, clases abstractas y/o interfaces.

Originalidad: Si se detectan proyectos con enfoques muy similares, serán rechazados y obtendrán calificación 0 (cero).

Funcionalidad: Capacidad del producto software para proporcionar las funcionalidades que satisfacen las necesidades explicitas e implícitas cuando el software se usa bajo unas ciertas condiciones.

> **Adecuación**: El programa ofrece las todas funcionalidades que respondan a la necesidades, tanto explícitas (contenidas en el documento descriptivo del proyecto) como implícitas. Entendiendo como necesidades implícitas, aquellas que no estando descritas en el documento, surgen como resultado de un concienzudo análisis del problema planteado y que aseguran el correcto funcionamiento del programa. (20%)
>
> **Exactitud**: el programa genera los resultados o efectos correctos o acordados, con el grado necesario de precisión. (5%)

Fiabilidad: Capacidad del producto software para mantener un nivel especificado de prestaciones cuando se usa bajo unas cierta condiciones.

> **Madurez**: el programa no presenta fallas originadas por errores de programación, análisis o diseño. (5%)
>
> **Tolerancia a fallos**: el programa maneja adecuadamente el manejo inadecuado del usuario; es decir, mantiene su prestación aun cuando el usuario introduzca datos erróneos o manipule inadecuadamente las interfaces de usuario. (5%)

Usabilidad: Capacidad del producto software para ser entendido,aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas.

> **Comprensibilidad**: el programa ofrece una interfaz de fácil comprensión, facilitando su aprendizaje y correcta utilización. El programa emite mensajes de alerta cuando se introducen valores erróneos. Existen elementos informativos que indican al usuario como operar el programa. (5%)
>
> **Diseño**: el diseño de la interfaz, esto es: disposición de controles, esquema de colores, mensajes y demás elementos gráficos; es atractivo para el usuario. (5%). **Bono: La realización de una excelente interfaz gráfica representará un bono de 1 punto sobre el total del proyecto, sin embargo la misma no es obligatoria.**

Eficiencia: Capacidad del producto software para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.

> Utiliza eficientemente las estructuras de datos para la solución del problema (arreglos y clases) basandose en la programación orientada a objetos. (15%)

Modificabilidad: Capacidad del producto software para ser modificado. Las modificaciones podrían incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y requisitos y especificaciones funcionales. La modificabilidad está directamente relacionada con la modularidad.

Documentación: Utiliza la documentación interna para facilitar la comprensión del programa. Incluye el diagrama de clases del proyecto. (10%)

Modularidad: se utilizan clases como mecanismo de estructuración del sistema y herencia como mecanismo de reutilización (10%)

Encapsulación: Se utilizan clases con sus diferentes modificadores de acceso como mecanismo de encapsulación. (10%)