Universidad Rafael Landívar
Facultad de ingeniería
Ingeniería industrial
Introducción a la programación, Sección 17
Catedrática: Damaris Campos

PROYECTO 2

FASE 1

Jennifer Fabiola Oseida Castillo
Carné: 2000223
Fecha de entrega: 31 de octubre 2023

ANALISIS Y DISEÑO DEL PROGRAMA

El análisis y diseño de un programa para un juego de Batalla Naval digital implica la definición de clases, atributos, métodos y un algoritmo para llevar a cabo todas las funciones requeridas en el juego. A continuación, se describe el diseño del programa:

Clases:

1. JuegoBatallaNaval: En este apartado se presentarán las instrucciones, es decir una pequeña descripción de como se desarrollará el juego entre los jugadores.
2. Descripción: Esta clase representa el juego en su totalidad y coordina las interacciones entre los jugadores y el tablero.
3. Barco:

- Representa un barco en el juego. Cada barco tiene propiedades como coordenadas iniciales, posición (vertical u horizontal), número de casillas, nombre y un área que almacena las casillas que ocupa.
- La clase Barco se utiliza para representar los barcos en el juego y realizar operaciones relacionadas con ellos.

4. Jugador:

- Representa a un jugador en el juego. Cada jugador tiene una flota (lista de barcos), un tablero (casillas donde coloca sus barcos) y una lista de ataques (tiros realizados).
- La clase Jugador se utiliza para llevar un seguimiento de las acciones de cada jugador y su flota de barcos.

5. Tiro:

- Representa un tiro realizado por un jugador. Cada tiro tiene coordenadas y un resultado que puede ser 0 (tiro fallido), 1 (tiro acertado) o 2 (barco hundido).
- La clase Tiro se utiliza para registrar los tiros de los jugadores y sus resultados.

Atributos:

1. Jugador jugador1: Representa al primer jugador.
2. Jugador jugador2: Representa al segundo jugador.
3. Tablero Jugador1: El tablero donde el primer jugador coloca sus barcos y dispara.
4. Tablero Jugador2: El tablero donde el segundo jugador coloca sus barcos y dispara.
5. Clase Casilla:

- Espacio_disp (str): Representa el contenido de la casilla ("B" para barco, "X" para tiro acertado, "-" para tiro fallido, " " para casilla vacía).
- Coordenada_x (int): Representa la coordenada X de la casilla en el tablero (valor entre 1 y 10).

- Coordenada_y (int): Representa la coordenada Y de la casilla en el tablero (valor entre 0 y 9).
- Estado (int): Representa el estado de la casilla (1 para casilla vacía, 2 para tiro acertado, 3 para tiro fallido, 4 para barco hundido).

6. Clase Barco:
- Casilla_inicial_x (int): La coordenada X de la casilla inicial del barco (valor entre 1 y 10).
- Casilla_inicial_y (int): La coordenada Y de la casilla inicial del barco (valor entre 0 y 9).
- Posicion (int): La posición del barco (0 para vertical, 1 para horizontal).
- Numero_casillas (int): El número de casillas que ocupa el barco.
- Nombre (str): El nombre del barco.
- Area (list): Una lista de objetos de la clase Casilla que representa las casillas que ocupa el barco en el tablero.

7. Clase Jugador:
- Flota (list): Una lista de objetos de la clase Barco que representa la flota de barcos del jugador.
- Tablero (list): Una lista de objetos de la clase Casilla que representa el tablero del jugador, con información de tiros y ubicación de barcos.
- Ataque (list): Una lista de objetos de la clase Tiro que representa los tiros realizados por el jugador.

8. Clase Tiro:
- Coordenadas (objeto Casilla): Representa las coordenadas (X, Y) del tiro realizado.
- Resultado (int): Representa el resultado del tiro (0 para tiro fallido, 1 para tiro acertado, 2 para barco hundido).

Estos atributos se utilizan para mantener un seguimiento de la configuración de los barcos, el estado de las casillas en el tablero y los resultados de los tiros realizados por los jugadores en el juego de batalla naval.

Métodos:

1. IniciarJuego(): Inicia el juego, permite a los jugadores colocar sus barcos en sus respectivos tableros y comienza los turnos.

2. `validar_int(msj)`:

- Descripción: Valida una entrada de usuario para asegurarse de que sea un número entero dentro del rango de 1 a 10.
- Parámetro: `msj` es un mensaje para solicitar la entrada del usuario.
- Retorna: El valor entero válido ingresado por el usuario.

3. `validar_int_pos(msj)`:
- Descripción: Valida una entrada de usuario para asegurarse de que sea un número entero que sea 0 o 1.
- Parámetro: `msj` es un mensaje para solicitar la entrada del usuario.
- Retorna: El valor entero válido ingresado por el usuario (0 o 1).

4. `validar_str(msj)`:
- Descripción: Valida una entrada de usuario para asegurarse de que sea una letra de la "A" a la "J".
- Parámetro: `msj` es un mensaje para solicitar la entrada del usuario.
- Retorna: La letra válida ingresada por el usuario convertida en un número entero correspondiente (0-9).

5. `dibujar_cuadricula(casilla_list)`:
- Descripción: Dibuja la cuadrícula del tablero del juego, mostrando las casillas vacías y el estado de las casillas con barcos, tiros acertados y tiros fallidos.
- Parámetro: `casilla_list` es una lista de objetos de la clase `Casilla`.

6. `dibujar_tiro(tiro, casilla_list2)`:
- Descripción: Dibuja el resultado de un tiro en el tablero, mostrando "X" para tiro acertado, "-" para tiro fallido y "@" para barco hundido.
- Parámetros:
- `tiro` es un objeto de la clase `Tiro` que contiene el resultado del tiro.
- `casilla_list2` es una lista de objetos de la clase `Casilla`.

7. `resultado_tiro(tiro, jugador, jugador_oponente)`:
- Descripción: Determina el resultado de un tiro (0 para tiro fallido, 1 para tiro acertado, 2 para barco hundido) y lo almacena en el objeto `Tiro`.
- Parámetros:
- `tiro` es un objeto de la clase `Tiro` que contiene las coordenadas del tiro.
- `jugador` es un objeto de la clase `Jugador` que representa al jugador que realiza el tiro.
- `jugador_oponente` es un objeto de la clase `Jugador` que representa al oponente.

8. `dibujar_barco(barco, casilla_list2)`:
- Descripción: Dibuja un barco en el tablero y verifica si el barco choca con otros barcos.
- Parámetros:
- `barco` es un objeto de la clase `Barco` que se está configurando.
- `casilla_list2` es una lista de objetos de la clase `Casilla` que representa el estado actual del tablero.

9. `solicitar_coordenadas(nombre, casilla_list2)`:
- Descripción: Solicita al jugador las coordenadas para configurar la posición de un barco en el tablero.
- Parámetros:
- `nombre` es el nombre del barco que se está configurando.
- `casilla_list2` es una lista de objetos de la clase `Casilla` que representa el estado actual del tablero.

10. `cambiar_espacio(estado)`:
- Descripción: Cambia el contenido de una casilla (espacio) en función de su estado (barco, tiro acertado, tiro fallido o casilla vacía).
- Parámetro: `estado` es un entero que representa el estado de la casilla.

11. `solicitar_tiro(jugador_oponente, jugador)`:
- Descripción: Permite al jugador seleccionar una casilla en la que disparar un tiro y registra el resultado en el tablero.
- Parámetros:
- `jugador_oponente` es el oponente del jugador que recibe el tiro.
- `jugador` es el jugador que realiza el tiro.

12. `main()`:
- Descripción: Función principal que controla la lógica del juego de batalla naval. Se encarga de la configuración inicial, la interacción entre jugadores y la finalización del juego.

Estas funciones/métodos se utilizan para gestionar aspectos clave del juego, como la configuración de los barcos, la realización de tiros, la representación gráfica del tablero y la determinación del ganador.

Condiciones y Restricciones:

El código proporcionado implementa un juego de batalla naval y tiene algunas condiciones y restricciones importantes a tener en cuenta:

1. Tamaño del tablero: El tablero del juego tiene un tamaño fijo de 10x10 casillas. Esto está representado por las letras de la A a la J (filas) y los números del 1 al 10 (columnas). El código asume que el tablero tiene estas dimensiones y valida que las coordenadas ingresadas por el usuario estén dentro de este rango.
2. Número de barcos por jugador: El juego permite que cada jugador configure una flota de cinco barcos. Los nombres y tamaños de los barcos son predefinidos: "Portaaviones" (5 casillas), "Acorazado" (4 casillas), "Crucero" (1 casilla), "Submarino" (3 casillas) y "Destructor" (2 casillas).
3. Posición de los barcos: El jugador puede elegir la posición de los barcos, ya sea vertical (0) u horizontal (1), y debe proporcionar las coordenadas iniciales para la ubicación de cada barco. El código verifica que las coordenadas sean válidas y que los barcos no se superpongan en el tablero.
4. Dibujo del tablero: El código dibuja el tablero en la consola, representando las casillas vacías, los barcos, los tiros acertados y los tiros fallidos utilizando caracteres especiales (" |" para casillas vacías, " B |" para barcos, " X |" para tiros acertados y " - |" para tiros fallidos).
5. Finalización del juego: El juego continúa hasta que uno de los jugadores haya realizado al menos 15 tiros acertados. Cuando esto sucede, se muestra un mensaje indicando el ganador del juego.
6. Validación de tiros: Antes de registrar un tiro en el tablero, el código verifica si la casilla ya ha sido atacada por el jugador. No permite que un jugador dispare a la misma casilla dos veces.
7. Restricciones de entrada de usuario: El código valida las entradas de usuario para garantizar que sean números enteros dentro de los rangos requeridos y letras de la "A" a la "J" para las coordenadas alfanuméricas.
8. Interacción por consola: El juego se juega en la consola y requiere que los jugadores ingresen las coordenadas y tomen decisiones utilizando la entrada y salida estándar.
9. Limitaciones gráficas y de interfaz: La representación visual del tablero se realiza mediante texto en la consola, por lo que no hay una interfaz gráfica de usuario (GUI).
10. Repetición de configuración de flota: El código permite a cada jugador configurar su flota de barcos, uno por uno, antes de comenzar la partida.

Estas condiciones y restricciones son parte integral del diseño del juego y deben cumplirse para que el juego funcione correctamente. Cualquier desviación de estas restricciones podría llevar a resultados inesperados o errores en la ejecución del código.

Algoritmo:

Resumen del algoritmo principal que rige el funcionamiento del juego:

1. La función `main` inicia el juego. Permite a cada jugador configurar su flota de barcos y luego inicia un bucle en el que los jugadores se turnan para realizar tiros hasta que uno de los jugadores haya realizado al menos 15 tiros acertados.
2. Se definen las clases `Casilla`, `Barco`, `Jugador`, y `Tiro` para representar los elementos del juego, como las casillas del tablero, los barcos, los jugadores y los tiros.
3. Se definen funciones para validar la entrada del usuario. Estas funciones garantizan que las coordenadas ingresadas estén dentro de los rangos válidos y cumplan con los requisitos del juego.
4. Se define la función `dibujar_cuadricula` que se encarga de mostrar el tablero en la consola, representando las casillas vacías, los barcos, los tiros acertados y los tiros fallidos.
5. Se define la función `dibujar_tiro` que se encarga de actualizar el tablero después de un tiro, mostrando si el tiro fue acertado, fallido o si hundió un barco.
6. Se define la función `resultado_tiro` que evalúa el resultado de un tiro, determinando si fue acertado, fallido o si hundió un barco. También actualiza la representación gráfica del tablero.
7. Se define la función `dibujar_barco` que coloca un barco en el tablero y verifica si choca con otros barcos previamente colocados.
8. Se define la función `solicitar_coordenadas` que permite a un jugador configurar su flota de barcos, eligiendo el nombre, posición y coordenadas iniciales de cada barco. Esta función utiliza las funciones de validación para garantizar que las coordenadas sean válidas y que los barcos no se superpongan.
9. Se define la función `cambiar_espacio` que cambia el estado de una casilla en función del resultado de un tiro.
10. Se define la función `solicitar_tiro` que permite a un jugador realizar un tiro, ingresando las coordenadas. Esta función también verifica si el tiro ya se ha realizado en esa casilla y actualiza el tablero en consecuencia.
11. Al final del juego, se muestra un mensaje que indica quién ganó.