**Universidad de Costa Rica**

**Escuela de Ciencias de la Computación e Informática (ECCI)**

**CI-0112 Programación I**

**Grupo: 002**

**Documento de decisiones: Proyecto 1.**

**Docente:**

**Luis Campos Duarte**

**Estudiantes:**

**Josué Rodríguez Aguilar I C4J023**

**Kevin Calderon Martinez I C4D511**

**I Ciclo**

**Abril 2025**

**a. Acuerdos de Pareja:** Describir los acuerdos alcanzados entre los miembros del equipo, como la división del trabajo y las responsabilidades.

El acuerdo logrado entre ambos estudiantes fue el de, principalmente, la subdivisión mutua de los juegos. A Kevin el ahorcado, y a Josué la batalla naval. Ambos lograron terminar su respectiva clase, y luego Kevin junto a Josué por llamada se encargaron de trabajar el Main de la Interfaz. Josué se encargó de crear el repositorio del proyecto en GitHub y trabajar el README, y ahí se fueron subiendo los avances del proyecto. El doxygen del Man se hizo en conjunto esfuerzo, mientras que la clase ahorcado y batalla naval lo trabajaron Kevin y Josué respectivamente.

**b. Decisiones de Diseño:** Explicar cómo se diseñaron las clases, cómo se estructuraron los métodos y cualquier decisión significativa tomada durante el diseño.

**Batalla Naval:**

El juego batalla naval consiste en una lucha por turnos entre dos jugadores que colocan sus barcos en tableros 5X5 y los intentan destruir mediante mecánicas de disparos, con la finalidad de que quien logre destruir todos los barcos de su oponente gana. Por lo cual, después de que el profesor realizara ciertas correcciones, se decidió partir de tres diferentes clases, Jugador Batalla Naval, Tablero y Batalla Naval para mantener un orden adecuado en la creación del juego y su entendimiento. Donde el jugador básicamente controla el sistema de disparo, el tablero se encargará de la colocación de barcos y confirmaciones de tiro y la batalla naval se compactara con las clases anteriores para presentar el sistema por turnos.

**Ahorcado:**

El ahorcado consiste principalmente en un método que se ha denominado como dosRondas, que consiste en que dos jugadores (j1 y j2) traten de adivinar una palabra secreta dada por su adversario, con 6 intentos para cada jugador (Primero adivina j2, luego j1). Y que se compare la cantidad de intentos restantes con la que quedó cada jugador, para definir al ganador.

Tras los cambios recomendados por el profesor, ahora se generan tres clases para el juego: Jugador (Contiene el nombre del jugador), Ahorcado (Los métodos de si ha ganado o perdido, reiniciar las variables (IniciarJuego), el método de adivinar letra), y el ControladorAhorcado (Se encarga de la mecánica de las dos rondas, además de el método de declarar el ganador).

**c. Decisiones de Implementación:** Detallar las decisiones específicas de implementación, como la elección de estructuras de datos, algoritmos utilizados, etc.

El juego de batalla naval se subdividió en tres clases con las siguientes características:

Clase Tablero: cuenta con los atributos



Cuenta con los métodos:



Verifica si se puede colocar un barco en el tablero.

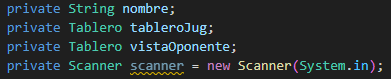


Muestra el tablero de los jugadores conforme se van colocando los barcos.



Se encarga de actualizar los disparos recibidos y verificar si fallaron, impactaron o se repitieron, con el fin de actualizar el tablero conforme se dispara en cada casilla.

Ahora la clase JugadorBatallaNaval cuenta con los atributos:



Y con los siguientes métodos:



Intenta colocar un barco en la posición indicada.



Se colocan los barcos a partir de la entrada en consola.



Muestra el tablero del jugador.



Muestra el tablero del oponente.



Básicamente la acción de disparar y actualizar el tablero del oponente en tiempo real.



Método que define ganador, verifica la existencia de barcos.

Como última clase, la BatallaNaval: Esta cuenta con los atributos



Y con los métodos:



Este método básicamente es el juego, se basa en la validación de turnos, verificación de entrada en consola y el proceso de todo el juego en general.



Este método se encarga de limpiar la pantalla alejando la información con saltos de línea en blanco.



Este método se encarga de que cada uno de los jugadores coloque sus barcos.

Para el ahorcado, se decidió usar Strings que definieron la palabra secreta, las letras buenas y letras malas, un int con los intentos restantes y un array de tipo char[] que guardase el avance de la palabra secreta mientras se adivinaba en la ronda. Primero, un método donde se verifique si las letras van en la palabra secreta o no, que sume o reste las letrasBuenas/Malas y que reste intentos. Dos booleanos que definan si el jugador ya adivinó la palabra (Ganó) o si se quedó sin intentos (Perdió). El super método de las dos rondas, que mediante un auxiliar guarde los intentos restantes de cada jugador, y que luego mediante un método declare quién ganó, o si fue un rotundo empate.

Tras las correcciones realizadas después de la presentación del proyecto, se divide el Ahorcado en tres clases:

Jugador: se define ahora el String nombre, que guarda el nombre de cada jugador.

Ahorcado: Se definen estas variables:

private String palabraSecreta;

private char[] progreso;

private String letrasBuenas;

private String letrasMalas;

private int intentosRestantes;

Y estos métodos

public void iniciarJuego(String palabraSecreta) / Se reinician todas las variables para cuando se inicie la ronda /

public boolean adivinarLetra(char letra) / Se verifica si la letra se encuentra en la palabra secreta o no /

public boolean haGanado() / Gana si todas las letras añadidas completan la palabra secreta /

public boolean haPerdido() / Pierde si se queda sin intentos restantes.

ControladorAhorcado: El encargado de correr las dos rondas, se divide en atributos:

private JugadorAhorcado jugador1;

private JugadorAhorcado jugador2;

private Ahorcado juego;

private Scanner scanner;

Y en estos métodos

public void jugar() /Metodo principal que corre las dos rondas en la interfaz/

public int jugarRonda(JugadorAhorcado jugadorPalabra, JugadorAhorcado jugadorAdivina) /Va dentro del método jugar, y aquí se hace la ronda para el jugador adivinador /

public void declararGanador(int puntajeJ1, int puntajeJ2) /Se define quien tiene mas puntos para declarar al ganador. Si son iguales se declara empate /

Cabe aclarar que no se menciona mucho sobre la clase interfaz pues está únicamente ejecuta el programa y permite al usuario determinar qué desea jugar y si quiere volver a jugar o no por lo que las decisiones de dicha clase no fueron determinantes solo se utilizo como método de acceso para las clases de ahorcado y batalla naval.

**d. Puntos de Mejora:** Identificar áreas que podrían mejorarse o ampliarse en futuras versiones del proyecto.

La comunicación entre ambos integrantes fue clara y concisa; no hubo problemas con respecto a la organización del proyecto, la codificación de las clases, la redacción de los doxygen y el documento de las decisiones.

Hay aspectos por mejorar para la clase Ahorcado; principalmente con la declaración poco necesaria de los setters que no se usaron en el Main, únicamente los getters en el mismo método del dosRondas de la clase Ahorcado. Obviando eso, el código fue bastante eficiente, y la mecánica de las dos rondas con la variable auxiliar que guarde los intentos restantes de cada jugador, bastante única.

Tomando en cuenta la división de las clases para el Ahorcado: Se cambió por completo el juego: Se corrigieron la división de las clases y se añadieron dos más al juego, para tener métodos ordenados, y ya se eliminaron los setters innecesarios.

Por último, con respecto a la clase de batalla naval lo que mejoraría sería la implementación de barcos con mayor tamaño y tableros con tamaños mayores y matrices no cuadradas pues en toda la implementacion del codigo, si el tamaño del tablero no es cuadrado ya no funcionara, y poco más la verdad que es muy eficiente la clase y cumple con lo requerido.