UNIVERSIDAD DE COSTA RICA

Escuela de Ciencia de la Computación e Informática

Programación I

Grupo 002

I semestre 2025

Profesor: Luis Campos Duarte

Estudiantes: Alondra Peña Quirós, C4I281; Leonardo Solano Ramírez, C4K103

DOCUMENTACIÓN EXTERNA DEL PRIMER PROYECTO PROGRAMADO

El presente documento tiene como objetivo documentar el proceso de desarrollo del primer proyecto de programación, explicando las decisiones de diseño, decisiones de implementación y las posibles mejoras que se identificaron durante la creación del programa.

Acuerdo de pareja

El presente documento tiene como finalidad dejar constancia de la distribución de tareas y del desarrollo correspondiente al primer proyecto de la asignatura Programación I, durante el primer ciclo del año 2025.

Tras un análisis y discusión conjunta del documento con los lineamientos y tareas a realizar, se ha acordado la siguiente distribución de responsabilidades:

Menú de inicio

La elaboración del menú de inicio será realizada de forma conjunta, dividiendo el trabajo en partes iguales (aproximadamente 50/50).

Batalla Naval	Ahorcado
Declaración de variables, creación de clases, inclusión de comentarios internos, implementación de métodos, y aplicación de buenas prácticas de programación. En síntesis, Leonardo será responsable del 100% del desarrollo del juego “Batalla Naval”	Declaración de variables, creación de clases, inclusión de comentarios internos, implementación de métodos, y aplicación de buenas prácticas de programación. En síntesis, Alondra será responsable del 100% del desarrollo del juego “Ahorcado”

Observación

En caso de ser necesario, ambos integrantes del equipo podrán intervenir en el código del juego que no les fue asignado con el fin de corregir errores o mejorar su funcionamiento, especialmente en situaciones de revisión o evaluación.

Decisiones de diseño

Clase Ahorcado

En cuanto a las decisiones de diseño, se decidió encapsular toda la lógica del juego Ahorcado dentro de una clase propia, esto para que sea modular e independiente, permitiendo que el juego se pueda integrar de una manera sencilla en el menú de la clase principal, desde donde se controlan los dos juegos.

Se definieron atributos privados como los arreglos progreso, letrasAdivinadas, intentosRestantes, letrasAdivinadas y cantidadLetrasAdivinadas, se accede a ellos mediante métodos públicos. El diseño de esta clase está pensado para que haya interacción por medio de consola, entonces se dejaron mensajes claros al usuario para que tenga una guía de los pasos que debe ir siguiendo para optimizar la experiencia del juego.

Clase BatallaNaval

Para diseñar la clase BatallaNaval, se priorizó la claridad y la organización del código dividiendo la lógica del juego en partes específicas. Se crearon tableros independientes para cada jugador usando matrices de caracteres (char [][]) que representan una cuadricula 5x5, donde se colocan los barcos y se registran los disparos de cada jugador. Además, se mantuvieron en matrices separadas para controlar y mostrar los disparos realizados por cada jugador, lo que ayuda a evitar confusiones entre las posiciones ocupadas y las que ya han sido atacadas.

La estructura de este programa se organizó en métodos con responsabilidades claras: inicializarTab, posicionDelBarco, disparos, mostrar el estado actual del juego y verificarSiHayGanador. Estas separaciones permiten la legibilidad y la facilidad de hacer algunas modificaciones o ampliaciones del código en algún futuro. También se implementó un control de turnos alternados, para que cada jugador juegue uno tras otro, siguiendo el orden correcto y respetando las reglas del juego.

Clase Menú

La clase Menu fue diseñada como el centro de control principal del programa. Es decir, es el punto de partida que conecta al usuario con los diferentes juegos desarrollados: Ahorcado y Batalla Naval. Desde el principio, se buscó que fuera intuitiva, interactiva y fácil de navegar para cualquier persona que ejecute el programa, sin importar su nivel de experiencia con la programación o el uso de consolas.

Una de las decisiones principales fue utilizar un menú textual en consola con opciones numeradas. Esta elección tiene como objetivo hacer que la navegación sea clara y directa: el usuario simplemente elige un número para acceder a un juego o salir del programa. Además, se reutiliza la misma estructura cada vez que el juego termina, lo que permite una experiencia fluida y continua, sin necesidad de reiniciar el programa manualmente.

Otra decisión importante fue el uso de un bucle while controlado por una variable booleana (sigueJugando). Esta estructura permite que el menú se repita indefinidamente hasta que el usuario decida salir. Esto brinda al jugador la libertad de probar ambos juegos varias veces, cambiar de uno a otro, o simplemente salir cuando lo desee.

También se decidió utilizar una estructura switch para manejar las diferentes opciones del menú. Esta decisión se tomó porque el switch permite organizar mejor el código cuando se trata de múltiples opciones basadas en un solo valor (en este caso, el número que el usuario digita). Así, cada opción del menú está claramente separada en su propio bloque, lo cual mejora la legibilidad y el mantenimiento del código.

Finalmente, se añadió una confirmación después de cada juego preguntando al jugador si desea jugar de nuevo. Esto da al usuario el control total sobre la experiencia, y refuerza la idea de un sistema flexible y centrado en quien lo está usando.

Decisiones de implementación

Clase Ahorcado

Se utilizaron arreglos de una dimensión, entre ellos char [], que es un arreglo de caracteres para manejar tanto el progreso del jugador como las letras adivinadas. Se utilizo char en vez de string por temas de eficiencias y ajuste a las necesidades del programa. Además, se creó una validación usando un ciclo for que recorre el arreglo letrasAdivinadas para asegurarse de que no se repita una letra que ya se intentó, tomando en cuenta que cada jugador tiene 6 intentos de adivinanza, por cada uno, una sola letra.

En cuanto a los métodos de esta clase, adivinarLetra centraliza la lógica del intento por letra y valida si pertenece a la palabra, al hacer esto actualiza el arreglo progreso o resta un intento. El método jugar, se añade para que permita ejecutar toda una partida desde la clase Ahorcado,

dejando que las funciones creadas interactúen entre ellas. Para garantizar un buen control del juego se utilizaron bucles y condicionales, por ejemplo, el while dentro del método jugar, además de condicionales if y else para controlar los flujos del juego.

Clase BatallaNaval

Se decidió usar arreglos bidimensionales (char [][]) para lograr representar visualmente el tablero porque es una estructura simple y eficiente para almacenar y acceder a posiciones en una cuadricula fija. Cada posición del tablero puede contener un símbolo que indica si hay un barco, un disparo, o un acierto. Esto facilita la lógica para validar los tiros y actualizar el estado del juego.

El estado de los barcos se representa con variables enteras que indican la "vida" restante de cada jugador, inicialmente con 300 puntos de vida, que se reduce 100 unidades por cada acierto. Esta forma numérica permite llevar un control sencillo y rápido del progreso del juego sin necesidad de rastrear cada barco individualmente.

En cuanto a las interacciones, se optó por usar la consola para solicitar las coordenadas de los jugadores, con validaciones para evitar estar afuera de rango o posiciones ya ocupadas o incluso si están vacías, esto por medio de símbolos como: "-", "X", "B" y "O"

El juego de desarrolla en el método "jugar ()" que orquesta todo el flujo: desde la colocación de los barcos hasta el fin de la partida. También se incluyó una variable booleana para alternar turnos "jugadorActual ()" y verifica la condición de victoria mediante el método "verificarSiHayGanador ()".

Cada jugador tiene su propio tablero y registro de disparos, lo cual refuerza muchísimo el diseño de turnos sin interferencia entre los datos.

Clase Menú

Durante la implementación de la clase Menu, se optó por organizar el programa dentro del método main, ya que este sería el punto de entrada del sistema. Desde allí, se creó un bucle while controlado por una variable booleana llamada siguejugando, con el propósito de mantener activo el menú mientras el usuario así lo desee. Dentro de este bucle, se imprime un menú textual con instrucciones claras que indican al usuario cómo acceder a cada juego disponible o cómo salir del programa.

Para la selección de las opciones, se utilizó una estructura switch, que permite gestionar de forma clara y ordenada las distintas elecciones del usuario. Cuando se selecciona la opción 1, el programa solicita una palabra secreta por parte del jugador 1 para iniciar el juego del ahorcado, y luego llama al método jugar () de la clase Ahorcado. En el caso de la opción 2, se instancia un objeto de la clase BatallaNaval y también se llama a su método jugar (), lo cual da inicio al segundo juego.

Después de cada partida, se pregunta al usuario si desea jugar de nuevo. Esta decisión se implementó con una simple comparación de texto (equals("s")) para detectar si el jugador desea seguir en el programa. Si responde que sí, el bucle del menú se repite; si no, el programa finaliza mostrando un mensaje de despedida. Esta estructura general permite que el usuario tenga una experiencia interactiva, clara y continua dentro del programa.

Puntos de mejora

Clase Ahorcado

Los puntos de mejora de esta clase se resumen en los siguientes puntos:

- Ocultar la palabra secreta ingresada por el jugador de manera más efectiva ya que la limpieza de pantalla actualmente no es tan optima.
- Se podría hacer más dinámico si se pudiera implementar otro arreglo que le muestre al usuario las letras incorrectas que ingresó.
- Actualmente no se maneja correctamente si la palabra tiene espacios o caracteres especiales, entonces se puede mejorar el soporte de estas palabras y la robustez del programa.
- Podría ser más interactivo si se mostrará mejor gráficamente en consola.

Clase BatallaNaval

Los puntos de mejora de esta clase se resumen en los siguientes puntos:

- Se podría mejorar separando la lógica en múltiples clases como “jugador”, “Tablero”, “Barco” entre otros. Esto podría hacer el código más reutilizable y fácil de mantener
- Se podrían reforzar las validaciones al momento de ingresar coordenadas, evitando errores como números fuera de rango o letras.

- Se podría añadir un modo para jugar contra la máquina (IA simple), o introducir distintas configuraciones de tablero y cantidad de barcos.
- Guardar el estado del juego o el historial de partidas sería una mejora útil para proyectos más complejos.

Clase Menú

Los puntos de mejora de esta clase se resumen en los siguientes puntos:

- En el futuro, se podría separar el menú en una clase independiente o delegar a métodos específicos para cada juego.
- Sería conveniente agregar validaciones adicionales para asegurar que el usuario no ingrese valores no numéricos o fuera de rango.
- En vez de repetir el mensaje de "¿Deseas jugar de nuevo?" dentro de cada caso, se podría centralizar esta lógica después del switch, haciendo el código más limpio y fácil de mantener.
- Se podrían agregar más elementos visuales o un diseño gráfico simple para hacerlo más atractivo.