Astucia Naval

Melisa Sánchez 614212787

Karen Mora 614222714

Juan Avila 614222706

Fundación Universitaria Konrad Lorenz

Fundamentos de programación

Nelson Armando Vargas Sánchez

25 de noviembre de 2022

Índice

Introducción

En este documento describiremos los temas del curso que nos brindaron la herramienta para resolver el reto propuesto como proyecto final del mismo.

Fases de creación de algoritmos

En esta etapa describiremos como implementamos el conocimiento conceptual adquirido en las primeras sesiones del curso, pero con un desarrollado junto con las herramientas adquiridas durante el curso al momento de realizar este proyecto.

Análisis

Nos han proveído un documento con los detalles de algunos casos de uso, y las descripciones pertinentes del producto a desarrollar, por ende nuestro análisis está enfocado en como dar solución a dicha rúbrica.

Al momento de leer detenidamente dicha rúbrica nos encontramos con patrones en común para algunos puntos de la misma, teniendo en cuanta esto nuestro diseño se enfocó en dividir en métodos que brinden un equilibrio entre la lectura del código y la reutilización de funcionalidad.

Diseño

El diseño lo explicaremos en diferentes secciones, persistencia y/o estructuras globales, Métodos para el tratamiento de estructuras, métodos de UI, funcionalidad del juego.

Persistencia y/o estructuras globales

Necesitaremos las siguientes estructuras a nivel global descritas para cada uno de los ítems que compondrán nuestra solución del problema.

- Variables Constantes: Tenemos un conjunto de estas que representan la posición en la cual almacenaremos el dato correspondiente a su nombre en una estructura de datos, ya sea un arreglo o una matriz:

```
static int X = 0, Y = 1;
// -- Ejemplo de uso en un metodo --
barcos[i][X] = cordenadas[X];
barcos[i][Y] = cordenadas[Y];
```

- Estructuras de datos: Para mantener el estado de nuestro juego decidimos usar 3 estructuras de datos, puntaje, barcos, tablero

```
/*
* M = MOVIMIENTOS
* F = FALLOS
*                        { M,  F } */
static int[] puntaje = { 0, 0 };
static int[][] tablero = new int[6][6];
/*
* X = ubicacion en x
* Y = ubicacion y
* O = ORIENTACION = Vertical | Horizontal
* L = longitud
*/
static int[][] barcos = {
/*{  X,   Y,  O,  L  } */
  { -1, -1, -1, 2, },
  { -1, -1, -1, 3, },
  { -1, -1, -1, 3, },
  { -1, -1, -1, 4, },
};
```

- Por último, pero no menos importante nuestra instancia de Scanner para poder tener una conexión constante con la terminal.

```
static Scanner sc = new Scanner(System.in);
```

Métodos para el tratamiento de estructuras

Los métodos que hacen parte de este conjunto son aquellos que realizan transformaciones con las estructuras descritas previamente, tales como:

- Restablecer a los valores iniciales.
- Almacenar elementos.
- Operar sobre los valores almacenados.
- Generar u obtener nuevos estados.

Estos métodos, si sus acciones transforman estructuras globales o locales, van a retornar dichas estructuras, aunque no sea necesario, esta decisión es consistente, en tanto se usa para mejorar la lectura y comprensión del código, por ejemplo:

```
static int[][] limpiarTablero() {
for (int y = 0; y < tablero.length; y++) {
  for (int x = 0; x < tablero[y].length; x++) {
    tablero[y][x] = 0;
  }
}
return tablero;
}
// -- Ejemplo de uso del metodo --
tablero = limpiarTablero();
```

Al momento de reasignar el valor retornado por el método no es necesario ir a su implementación para comprender que este está realizando una transformación sobre dicha estructura.

Métodos de UI

Como objetivo principal de estos serian brindar concesión con el usuario tanto de entrada como de salida de datos, evitando reescribir código constantemente para este objetivo, como ejemplo de este conjunto de métodos el siguiente se encarga de recibir por argumento las opciones para imprimir un menú en pantalla y retorna el índice del valor seleccionado:

```
static int crearMenu(String[] opciones) {
    int seleccion = 0;
    do {
        for (int i = 0; i < opciones.length; i++) {
            System.out.println((i + 1) + ". " + opciones[i]);
        }
        System.out.println("Seleccione una opcion del menu.");
        seleccion = sc.nextInt() - 1;
    } while (seleccion < 0 || seleccion >= opciones.length);
    return seleccion;
}
```

Funcionalidad del juego

Los métodos aquí descritos disponen de variables y/o estructuras locales que están directamente vinculadas con la funcionalidad del juego, o elementos de UI y sus métodos, también disponen uso de las estructuras globales previamente descritas junto con sus métodos de transformación haciendo uso de estas herramientas para ofrecer en ellos la lógica del juego y de cada una de sus opciones. Como ejemplo de este grupo mostraremos atacar barco.

```
static void atacarBarco() {
    int[] coordenadas;
    int[][] tableroConBarcos;
    tableroConBarcos = ubicarBarcos();
    do {
        System.out.println("Ingresar posicin mapa:");
        coordenadas = leerCoordenadas();
    } while (coordenadasFueraDeLimites(coordenadas));
    puntaje[MOVIMIENTOS] += 1;
    if (tableroConBarcos[coordenadas[Y]][coordenadas[X]] == 1) {
        tablero[coordenadas[Y]][coordenadas[X]] = 1;
        System.out.println("Barco atacado");
        mostrarTablero(tablero);
    }
    if (tableroConBarcos[coordenadas[Y]][coordenadas[X]] == 0) {
        System.out.println("Barco NO atacado");
        puntaje[FALLOS] += 1;
    }
}
```

Solución

Nuestra solución está dada por la composición de los métodos y estructuras previamente descritas.

Sentencia de control

Sentencia *if*

La sentencia, *if* tras evaluar una expresión lógica, ejecuta un bloque de código en caso de que la expresión lógica sea verdadera, esta la usamos en gran cantidad de lugares de nuestro juego, para manejar errores estados de la aplicación o casos base de algún método. pro ejemplo:

```
System.out.println("Ingresar posicin inicial del barco de longitud " +
    barcos[i][LOGITUD] + ":");
int[] coordenadas = leerCoordenadas();
barcos[i][X] = coordenadas[X];
barcos[i][Y] = coordenadas[Y];
if (coordenadasFueraDeLimites(coordenadas)) {
  System.out.println("Error: Barco fuera de los limites, intenta de nuevo.");
  continue;
}
```

Donde usamos la sentencia *if* para validar si la coordenada leída está fuera de los límites del tablero.

Sentencia *switch*

La sentencia *switch* conocida también como máquina de estados, recibe una variable y con base a su valor va a ejecutar el caso o bloque de código relacionado con este, nosotros lo empleamos en el método que va a ejecutar la sección correspondiente seleccionada en el menú principal.

```
static void seccion(String option) {
    int[][] tableroConBarcos;
    switch (option) {
        case "Reiniciar juego":
            reiniciarJuego("reiniciado");
            break;
        case "Atacar barco":
            atacarBarco();
            break;
        case "Cambiar barco":
            cambiarBarco();
            break;
        case "Ver tablero original":
            tableroConBarcos = ubicarBarcos();
            mostrarTablero(tableroConBarcos);
            break;
        case "Salir":
            System.out.println("G1racias por participar en el juego de Astucia
                Naval.");
            break;
        default:
            break;
    }
}
```

Ciclos

Sentencia *while*

La sentencia *while* ejecutará un bloque de código mientras al evaluar una expresión lógica esta sea verdadera. La usamos repetir una acción hasta que las condiciones de validación sean las esperadas.

```
while (true) {
   System.out.println("Ingresar posicin inicial del barco de longitud " +
       barcos[i][LOGITUD] + ":");
   int[] coordenadas = leerCoordenadas();
   barcos[i][X] = coordenadas[X];
   barcos[i][Y] = coordenadas[Y];
   if (coordenadasFueraDeLimites(coordenadas)) {
       System.out.println("Error: Barco fuera de los limites, intenta de
           nuevo.");
       continue;
   }
   System.out.println("Ingresar la orientacion del barco:");
   orientacion = crearMenu(opcionesDeOrientacion);
   barcos[i][ORIENTACION] = orientacion;
   longitud = barcos[i][LOGITUD];
   if ((orientacion == VERTICAL && barcos[i][Y] + longitud >= tablero.length)
   || (orientacion == HORIZONTAL && barcos[i][X] + longitud >=
       tablero[0].length)) {
       System.out.println("Error: barco fuera de los limites, intenta de
           nuevo.");
       continue;
   }
   tableroIpotetico = ubicarBarcos();
```

```
    if (barcoSobreBarco(tableroIpotetico)) {
        System.out.println("Error: Barco sobre barco, intenta de nuevo.");
        continue;
    }
    mostrarTablero(tableroIpotetico);
    break;
}
```

Sentencia *do...while*

La sentencia *do...while* ejecutará un bloque de código mientras al evaluar una expresión lógica esta sea verdadera. pero ejecutará el bloque de código por lo menos una vez. La usamos repetir menús.

```
static void menuPrincipal() {
    String[] opcionesPrincipales = { "Reiniciar juego", "Atacar barco",
        "Cambiar barco", "Ver tablero original", "Salir" };
    int seleccion = 0;
    do {
        seleccion = crearMenu(opcionesPrincipales);
        System.out.println("Opcin " + (seleccion + 1) + ": " +
            opcionesPrincipales[seleccion]);
        seccion(opcionesPrincipales[seleccion]);
        if (gano()) {
            seleccion = 4;
            System.out.println("Ya se han atacado todos los barcos.");
            System.out.println("Cantidad total de movimientos: " +
                puntaje[MOVIMIENTOS] + ".");
            System.out.println("Fallos " + puntaje[FALLOS] + ".");
        }
        if (puntaje[FALLOS] >= 20) {
            seleccion = 4;
            System.out.println("Lo sentimos, excediste los intentos.");
        }
    } while (opcionesPrincipales[seleccion] != "Salir");
}
```

Sentencia *for*

La sentencia *for* ejecutará un bloque de código mientras al evaluar una expresión lógica esta sea verdadera con respecto a la variable y el incremento. La usamos recorre las estructuras de datos.

```
static int[][] ubicarBarcos() {
    int[][] tableroConBarcos = new int[6][6];
    int x = 0, y = 0, longitud = 0, orientacion = 0;
    for (int i = 0; i < barcos.length; i++) {
        x = barcos[i][X];
        y = barcos[i][Y];
        longitud = barcos[i][LOGITUD];
        orientacion = barcos[i][ORIENTACION];
        if (orientacion == VERTICAL) {
            for (int celda = 0; celda < longitud; celda++) {
                tableroConBarcos[y + celda][x] += 1;
            }
        }
        if (orientacion == HORIZONTAL) {
            for (int celda = 0; celda < longitud; celda++) {
                tableroConBarcos[y][x + celda] += 1;
            }
        }
    }
    return tableroConBarcos;
}
```