

<Iván López Medina>

<Ivan Rubi Lozano>

<Guerra de Barcos>

Funcionalitat: Insertar Barco

Localització: Tablero.java, setBarco()

Test: TestTablero.java. Para partición equivalente se crean los métodos test_ColocarBarcoOcupadoHorizontal(), test_ColocarBarcoOcupadoVertical(), test_ColocarBarcoVacioHorizontal(), test_ColocarBarcoVacioVertical(). Estos tests definen los criterios de valido e invalido. Con las funciones test_ColocarBarcoFueraVertical() y test_ColocarBarcoFueraHorizontal() utilizamos la técnica de valores límites. Todo esto son pruebas de caja negra.

Funcionalitat: Crear tablero

Localització: Tablero.java, tablero()

Test: TestTablero.java. Se realizan dos tests de particion equivalente de caja negra. Test_insertTableroValido() y test_insertTableroInvalido(). Estos tests definen los criterios de valido e invalido.

Funcionalitat: Crear barco

Localització: Barco.java, barco(), setTamano()

Test: TestBarco.java. test_instanciarBarco(). Testeamos que en función del nombre proporcionado podamos instanciar los 4 tipos de barcos. Test_hundirBarco() en este utilizamos los valores límite para comprobar que no Podemos restar vida negative a un barco.

Funcionalitat: Disparar

Localització: Jugador.java, disparar()

Test: TestJugador.java. test_disparo(). Función en la que hacemos uso de un mockobject que simula un teclado el cual tendrá un array de filas y columnas a las cuales ir disparando. Realizamos diferentes pruebas de caja negra con particiones equivalentes y valores límite.

Funcionalitat: Imprimir Tablero

Localització: Tablero.java, usado en main()

Test: JuegoTest.java. test_pintarTablero. Se crea un test el cual inicia una partida y con un mockobject que hace de teclado simulamos una partida para comprobar que imprima bien los valores del tablero (tocado, hundido, agua...)

Funcionalitat: Obtener barco

Localització: Tablero.java, getBarco()

Test: TableroTest.java. test_loopGetBarco. Se hace loop testing con 0 y 1 iteraciones.

Funcionalitat: Insercion aleatoria de barcos

Localització: Tablero.java, insertarBarcosAleatorios()

Test: test_insercionAleatoria(). Se generan aleatoriamente todos los barcos, no necesita mucho testing ya que se testea a la hora de testear todos los barcos e inserciones pero testeamos que se introduzcan todos de manera correcta. Introducimos 0, 1, 4 y 5. 4 Es el correcto.

Funcionalitat: Condición de victoria

Localització: Juego.java, insertarBarcosAleatorios()

Test: test_condicionVictoria(). Se testea la condición de victoria, que cuando se hundan los 4 barcos termina el juego. Se testea con diferente número de barcos, ya que es importante que termine.

Funcionalitat: Code Coverage function setBarco

Localització: Clase Tablero método setBarco

Test: Comprobamos que pasamos por todas las líneas de código, este es solo un ejemplo para tenerlo documentado, ya que seguimos el Statement Coverage en todo el proyecto

```
public boolean setBarco(Barco b, int x, int y, char orientacion) {
    result = true;
    b.setOrientacion(orientacion);
    if (orientacion == 'v') {
        if (comprobarPosicionBarco(b.getTamano(), x, y, orientacion)) {
            System.out.println(b.getTamano());
            int x_init = x;
            b.setPosicionBarco(x,y); //Barco guarda la posicion en la que ha sido colocado en el tablero
            setBarcos(b);

            int v = 0;
            while (x < x_init+b.getTamano()) {
                b.setPosicionVerticalBarco(v, x);
                System.out.println("Barco ok");
                matriz[x][y] = 4;
                x++;
                v++;
            }
        } else result = false;
    }

    if (orientacion == 'h') {
        if (comprobarPosicionBarco(b.getTamano(), x, y, orientacion)) {
            int y_init = y;
            b.setPosicionBarco(x,y); //Barco guarda la posicion en la que ha sido colocado en el tablero
            setBarcos(b);

            int h = 0;
            while (y < y_init+b.getTamano()) {
                b.setPosicionHorizontalBarco(h,y);
                System.out.println("Barco ok");
                matriz[x][y] = 4;
                y++;
                h++;
            }
        } else result = false;
    }
    return result;
}
```

El mismo método setBarco llama al método comprobarPosicionBarco, que revisa que no haya otro barco en esa posición. Como podemos ver hay líneas rojas, esas líneas son necesarias, ya que los barcos se colocan en el tablero de manera ALEATORIA. Pues sin ellas generaríamos un tablero incorrecto.

```
public boolean comprobarPosicionBarco(int tamano, int x, int y, int orientacion) {
    if (orientacion == 'h') {
        int y_init = y;
        if (y_init + tamano > 8) {
            System.out.println("Error insertando el barco. Te sales");
            return false;
        } else {
            while (y < y_init+tamano) {
                if (matriz[x][y] == 4) {
                    System.out.println("Error insertando el barco. Posicion ocupada");
                    result = false;
                }
                y++;
            }
        }
    }
















    if (orientacion == 'v') {
        int x_init = x;
        if (x_init + tamano > 8) {
            System.out.println("Error insertando el barco. Te sales");
            return false;
        } else {
            while (x < x_init+tamano) {
                if (matriz[x][y] == 4) {
                    System.out.println("Error insertando el barco. Posicion ocupada");
                    result = false;
                }
                x++;
            }
        }
    }
    return result;
}
```

Barco Acorazado en posicion h en X : 3Y:2
Error insertando el barco. Te sales
Error insertando el barco. Posicion ocupada
Error insertando el barco. Posicion ocupada
Error insertando el barco. Te sales
Error insertando el barco. Posicion ocupada
Barco ok

Funcionalitat: Statement coverage de las clases







Al haber métodos que se usan en tests no se puede llegar al 100% en esta ejecución ya que la realizamos sobre el main,

Localització: Clase Tablero

▼ Tablero.java		97.2 %	565	16	581
▼ Tablero		97.2 %	565	16	581
● setValorCasilla(int, int)		0.0 %	0	8	8
● getBarcos()		0.0 %	0	3	3
● getMatriz()		0.0 %	0	3	3
● barcoValido(String)		91.3 %	21	2	23
● Tablero()		100.0 %	51	0	51
● comprobarPosicionBarco(int, int...)		100.0 %	73	0	73
● getBarco(int, int)		100.0 %	78	0	78
● getMatrizValor(int, int)		100.0 %	7	0	7
● getTotalBarcos()		100.0 %	3	0	3
● imprimirTablero()		100.0 %	73	0	73
● insertarBarcosAleatorios()		100.0 %	118	0	118
● quitarBarco()		100.0 %	7	0	7
● setBarco(Barco, int, int, char)		100.0 %	115	0	115
● setBarcos(Barco)		100.0 %	11	0	11
● setMatrizValor(int, int, int)		100.0 %	8	0	8



















Funcionalitat: Statement coverage

Localització: Clase Juego

▼ Juego.java		81.5 %	53	12	65
▼ Juego		81.5 %	53	12	65
● entradaUsuario()		0.0 %	0	4	4
● setTeclado(Teclado)		0.0 %	0	4	4
● terminarPartida()		0.0 %	0	1	1
● main(String[])		100.0 %	53	0	53

Funcionalitat: Statement coverage

Localització: Clase Barco

▼ Barco.java		95.3 %	142	7	149
▼ Barco		95.3 %	142	7	149
● getVida()		0.0 %	0	3	3
● Barco(String)		100.0 %	37	0	37
● getHundido()		100.0 %	5	0	5
● getNombre()		100.0 %	3	0	3
● getOrientacion()		100.0 %	3	0	3
● getPosicionBarco()		100.0 %	3	0	3
● getPosicionHorizontalBarco()		100.0 %	3	0	3
● getPosicionVerticalBarco()		100.0 %	3	0	3
● getTamano()		100.0 %	3	0	3
● restVida()		100.0 %	13	0	13
● setHundido()		100.0 %	7	0	7
● setOrientacion(char)		100.0 %	4	0	4
● setPosicionBarco(int, int)		100.0 %	13	0	13
● setPosicionHorizontalBarco(int, int)		100.0 %	8	0	8
● setPosicionVerticalBarco(int, int)		100.0 %	8	0	8
● setTamano()		100.0 %	29	0	29

Funcionalitat: Statement coverage

Localització: Jugador

▼ Jugador.java	98.6 %	212	3	215
▼ Jugador	98.6 %	212	3	215
● getNombre()	0.0 %	0	3	3
● Jugador(String)	100.0 %	6	0	6
● disparar(Tablero, int, int)	100.0 %	194	0	194
● pideColumna(Scanner)	100.0 %	6	0	6
● pideFila(Scanner)	100.0 %	6	0	6

Funcionalitat: Condition coverage, convertido a decisión coverage ya que es simple

Localització: Disparar

Como el usuario puede disparar donde quiera, se deben tener en cuenta todas esas condiciones, que vienen agrupadas en el MockObject de disparar.

En la condición `if (x >= 0 && x < 9 && y >= 0 && y < 9)` tenemos 8 posibles casos. Hemos realizado una combinatoria para poder obtener las posibles combinaciones, que las probamos en un mockObject,

```
int[] valorFila = {0,1,0,7,6,5,8,8,10,-1,0,-1,-1,0,10,10};
int[] valorCol = {0,0,1,5,5,5,8,8,10,-1,-1,0,10,10,-1,0};

public int disparar(Tablero t, int x, int y){
    if(x>=0 && x<9 && y>=0 && y<9) { 8
        Barco barco = t.getBarco(x, y); //Me guardo el barco que esta en esa posicion disparada
        int ret = 0;
        if (t.getMatrizValor(x, y) == 4) {
            System.out.println("Position X: " + x + " Y : " + y);
            barco.restVida(); //restamos la vida del barco
            if(barco.getHundido()){
                System.out.println("BARCO HUNDIDO" + barco.getNombre());
                t.quitarBarco();
                int init_x = barco.getPosicionBarco()[0];
                int init_y = barco.getPosicionBarco()[1];
                char orientacion = barco.getOrientacion();
                System.out.println(init_x + "Y = " + init_y + orientacion);
                if (orientacion == 'h') {
                    for (int i = init_y; i < init_y + barco.getTamano(); i++) {
                        t.setMatrizValor(init_x, i, 2);
                    }
                }
                if (orientacion == 'v') {
                    for (int i = init_x; i < init_x + barco.getTamano(); i++) {
                        System.out.println(init_x+i);
                        t.setMatrizValor(i, init_y, 2);
                    }
                }
                t.setMatrizValor(x, y, 2);
                ret = 3; //Valor para los test
            } else {
                System.out.println("Tocado");
                t.setMatrizValor(x, y, 1);
                ret = 1;
            }
        } else {
            if ( t.getMatrizValor(x, y) != 3) {
                System.out.println("Casella ja seleccionada");
                ret = t.getMatrizValor(x, y);
            } else {
                t.setMatrizValor(x, y, 0);
                System.out.println(" x " + x + " y "+ y);
                ret = 0;
            }
        }
    }
    return ret; //Valor para el test
} else {
    System.out.println("Fuera de rango");
    return 2; //Valor para el test
}
}
```

Funcionalitat: Path coverage

Localització: setbarco()

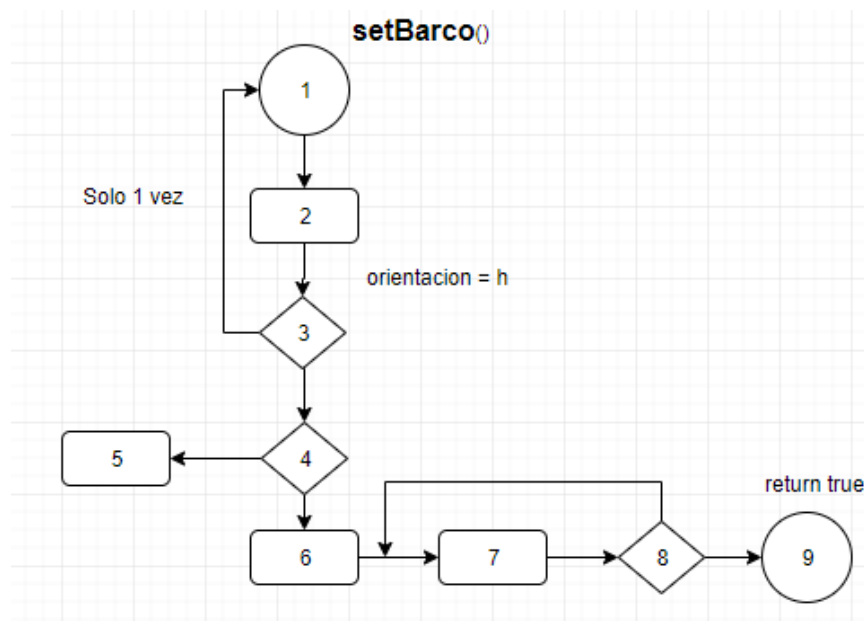
Path coverage del mètode setBarco, este mètode mira primero si es horizontal o vertical, el diagrama muestra solo 1, ya que el segundo es muy similar y se puede interpretar como repetir con orientación 'h' y 'v'

```
public boolean setBarco(Barco b, int x, int y, char orientacion) {
    result = true;
    b.setOrientacion(orientacion);
    if (orientacion == 'v') { A
        if (comprobarPosicionBarco(b.getTamano(), x, y, orientacion)) {
            System.out.println(b.getTamano());
            int x_init = x;
            B b.setPosicionBarco(x,y); //Barco guarda la posicion en la que ha sido colocado en el tablero
            setBarcos(b);

            int v = 0;
            while (x < x_init+b.getTamano()) {
                b.setPosicionVerticalBarco(v, x);
                C System.out.println("Barco ok");
                matriz[x][y] = 4;
                x++;
                v++;
            }
        } else result = false; D
    }

    if (orientacion == 'h') { A
        if (comprobarPosicionBarco(b.getTamano(), x, y, orientacion)) {
            int y_init = y;
            B b.setPosicionBarco(x,y); //Barco guarda la posicion en la que ha sido colocado en el tablero
            setBarcos(b);

            int h = 0;
            while (y < y_init+b.getTamano()) {
                b.setPosicionHorizontalBarco(h,y);
                C System.out.println("Barco ok");
                matriz[x][y] = 4;
                y++;
                h++;
            }
        } else result = false; D
    }
    return result;
}
```



$$V(G) = 9 - (8 - 2) = 3$$

Paths:

P1: 1,2,3,4,5,6,5,6,7 ; P2: 1,2,3,8; P3: 1,2,1,2,3,4,5,6,7