

Buscaminas de Sergio Gallo y Raul Estrada

Nuestro fork: <https://github.com/1494116/Buscaminas>

Angel Garcia Calleja - 1490917

Daniel Calvo Ramos - 1494116

Test cases a partir de casos de uso

Para poder realizar los test cases, nosotros nos hemos basado en los documentos de teoría:

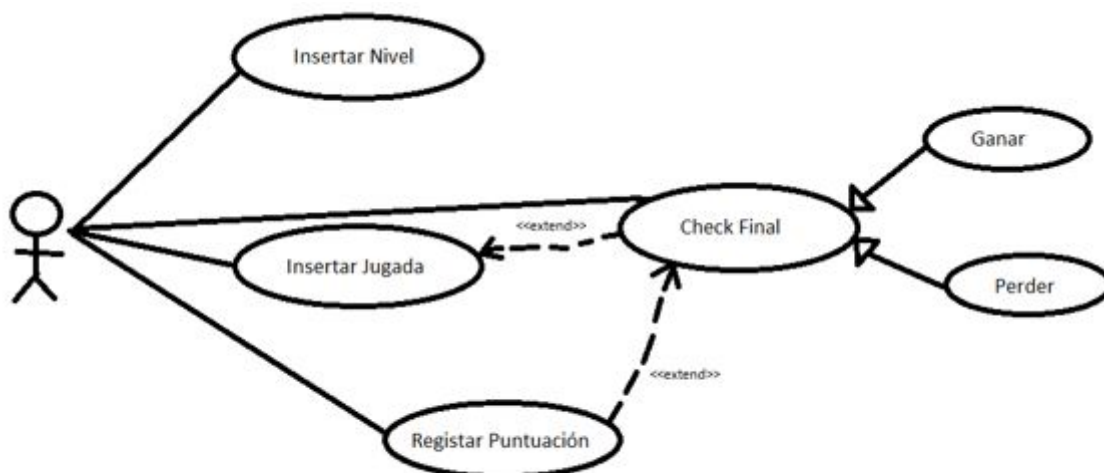
1. **Identificar los casos de uso:** Primero de todo hemos realizado un diagrama de casos de uso para identificar de forma precisa los casos de uso de nuestro juego.
2. **Identificar escenarios a partir de casos de uso:** Se cubre todo el conjunto de escenarios para cada caso de uso.
3. **Identificar test cases a partir de un escenario:** Identificar los test cases y las condiciones que permiten que se ejecute.

Nosotros hemos realizado los test cases siguiendo estos patrones, ya que son los explicados en teoría y de la forma en la que mejor lo hemos entendido.

El patrón que seguirá este apartado será que para cada caso de uso sacaremos todos los flujos de ejecución posibles (escenarios) y partir de estos escenarios se probarán diferentes test data, lo que resultará en los test cases.

También mencionar que a la hora de realizar las tablas, no hemos seguido al pie de la letra los templates provistos en el campus virtual. Las tablas que nosotros hemos creado siguen el formato explicado en el párrafo anterior.

Realizamos el siguiente diagrama de casos de uso para ver mejor de donde se podían sacar escenarios.



Los casos de uso de los que analizaremos escenarios son Insertar nivel, Insertar jugada y Registrar puntuación.

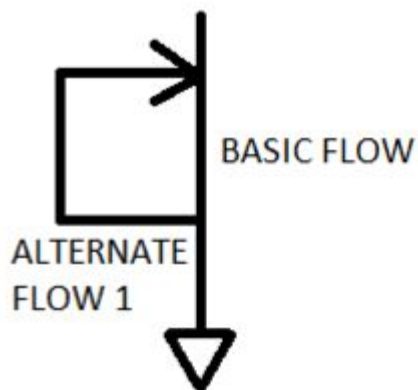
Para los escenarios de estos crearemos un diagrama de flujo de cada caso y miraremos cómo dependiendo de los inputs que introducimos el programa actúa de la manera esperada o no.

Gracias a realizar estos casos hemos encontrado resultados que se desvían de los que nosotros consideramos correctos.

Por ejemplo si pones una bandera en una casilla donde hay una mina y después la abres, la mina explota. En nuestra opinión si has marcado una casilla no la deberías de poder abrir, la tendrías que desmarcar primero.

Caso de uso Insertar nivel: nos basamos en el procedimiento que tiene el programa de comenzar la partida. Primero te pide que introduzcas un nivel de 4 disponibles, después comprueba que ese nivel es válido y finalmente comienza la partida. En el caso que el nivel sea incorrecto, el programa te vuelve a pedir que introduzcas el nivel de nuevo.

Test Case ID	Insertar Nivel		
Descripción	Al iniciar el juego el sistema pide al usuario que introduzca el nivel en el que desea jugar.		
Módulo	Buscaminas		
Hecho por	Daniel y Angel	Fecha	12/12/2020



Basic Flow: introducir nivel correctamente

Alternate Flow 1: introducir nivel incorrectamente (te lo vuelve a pedir)

Escenario ID 01		Basic Flow		
TC ID	Test Steps	Test Data (Nivel)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel	1	Se inicia una partida de nivel Fácil.	Se inicia una partida de nivel Fácil.
02	1. Iniciar juego 2. Introducir Nivel	2	Se inicia una partida de nivel Medio.	Se inicia una partida de nivel Medio.
03	1. Iniciar juego 2. Introducir Nivel	3	Se inicia una partida de nivel Difícil.	Se inicia una partida de nivel Difícil.
04	1. Iniciar juego 2. Introducir Nivel	4	Se inicia una partida de nivel Muy Difícil.	Se inicia una partida de nivel Muy Difícil.
05	1. Iniciar juego 2. Introducir Nivel	5	Te pregunta si quieres salir.	Te pregunta si quieres salir.

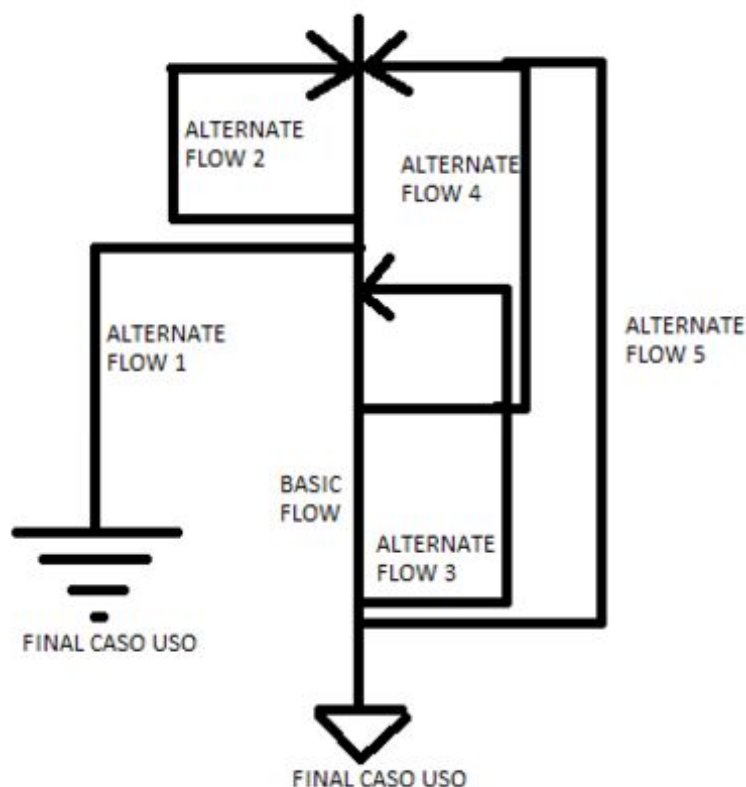
Escenario ID 02		Basic Flow + Alternate Flow 1		
TC ID	Test Steps	Test Data (Nivel)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel	0	Mensaje de fuera de rango.	Te pide introducir el nivel de nuevo.
02	1. Iniciar juego 2. Introducir Nivel	6	Mensaje de fuera de rango.	Te pide introducir el nivel de nuevo.
03	1. Iniciar juego 2. Introducir Nivel	-5	Mensaje de fuera de rango.	Te pide introducir el nivel de nuevo.
04	1. Iniciar juego 2. Introducir Nivel	300	Mensaje de fuera de rango.	Te pide introducir el nivel de nuevo.
05	1. Iniciar juego 2. Introducir Nivel	3.0	Mensaje de caracter inválido.	Mensaje de caracter inválido. Te pide introducir el nivel de nuevo.
06	1. Iniciar juego 2. Introducir Nivel	a3	Mensaje de caracter inválido.	Mensaje de caracter inválido. Te pide introducir el nivel de nuevo.
07	1. Iniciar juego 2. Introducir Nivel	Tres	Mensaje de caracter inválido.	Mensaje de caracter inválido. Te pide introducir el nivel de nuevo.

Caso de uso Insertar Jugada: agrupamos en un mismo caso de uso 3 inputs diferentes: opción, fila y columna ya que son las que confeccionan la jugada en sí. Comprobando los valores de este caso de uso nos dimos cuenta que el comportamiento del programa era diferente dependiendo de que input se introducía.

La opción funciona como el nivel: introduces un int dentro del rango y este se guarda en la opción, si lo pones un int fuera del rango te pide introducirlo otra vez y si no introduces un int y metes un string o un float te muestra un mensaje de carácter inválido y te pide introducirlo de nuevo.

Las filas y columnas funcionan de igual manera: introduces un int como fila/columna dentro del rango y se realiza la jugada. Si no introduces un valor dentro del rango (tablero) te volverá a preguntar por ellas pero en cambio si introduces un string o un float te mostrará un mensaje de carácter inválido y tendrás que volver a introducir la jugada desde el principio. Esto causa que haya diferentes flows alternativos para estos casos.

Test Case ID	Insertar Jugada		
Descripción	Una vez iniciada la partida el sistema nos pide introducir la jugada que deseamos realizar (opción + fila + columna).		
Módulo	Buscaminas Nivel 1		
Hecho por	Daniel y Angel	Fecha	12/12/2020



Basic flow: jugada correcta

Alternate flow 1: opción Salir

Alternate flow 2: opción incorrecta

Alternate flow 3: fila/columna incorrecta (fuera de rango)

Alternate flow 4: fila mal (caracter invalido)

Alternate flow 5: columna mal (caracter invalido)

La diferencia entre los alternate flow 3 y 4-5 es que en el Alternate flow 3 si pones una fila o una columna incorrecta, por ejemplo fila 0, una vez introduzcas la columna te volverá a pedir la fila y la columna, guardándote la opción que habías seleccionado. En cambio en los Alternate flow 4 y 5 si pones un carácter inválido de fila o columna inmediatamente vuelve a pedirte la opción deseada.

Ejemplo: si pones opción marcar (1), después fila 0, el sistema te preguntará por la columna y una vez introducida te volverá a pedir la fila ya que fila 0 no existe.

Si pones opción marcar (1) y después fila A, el sistema te “expulsa” y te vuelve a pedir la opción, no avanza y te pregunta por la columna.

Escenario ID 03		Basic Flow				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	1	Casilla [1][1] marcada	Casilla [1][1] marcada + aviso bomba detectada
02	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	5	Casilla [1][5] marcada	Casilla [1][5] no marcada + aviso no hay bomba
03	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	5	1	Casilla [5][1] marcada	Casilla [5][1] no marcada + aviso no hay bomba
04	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	5	5	Casilla [5][5] marcada	Casilla [5][5] marcada + aviso bomba detectada
05	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	1	1	Casilla [1][1] desmarcada	Mensaje no desmarcar que hay una bomba
06	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	1	5	Casilla [1][5] desmarcada	Casilla [1][5] desmarcada
07	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	5	1	Casilla [5][1] desmarcada	Casilla [5][1] desmarcada
08	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	5	5	Casilla [5][5] desmarcada	Mensaje no desmarcar que hay una bomba
09	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	1	1	Casilla [1][1] abierta	Casilla [1][1] abierta

10	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	1	5	Casilla [1][5] abierta	Casilla [1][5] abierta
11	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	5	1	Casilla [5][1] abierta	Casilla [5][1] abierta
12	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	5	5	Casilla [5][5] abierta	Casilla [5][5] abierta

Ahora haremos más comprobaciones según el estado de la casilla que queremos abrir, por ejemplo de la casilla[1][1]

Escenario ID 03_01		Basic Flow				
Caso concreto		Casilla cerrada y sin bomba				
TC ID	Test Steps	Test Data (Op)	Estado	Mina	Resultado esperado	Resultado obtenido
13	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	Cerrado	No	Casilla marcada	Casilla no marcada + Mensaje No hay mina
14	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	Cerrado	No	Casilla desmarcada	Casilla desmarcada
15	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	Cerrado	No	Casilla abierta	Casilla abierta

Escenario ID 03_02		Basic Flow				
Caso concreto		Casilla cerrada y con bomba				
TC ID	Test Steps	Test Data (Op)	Estado	Mina	Resultado esperado	Resultado obtenido
16	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	Cerrado	Si	Casilla marcada	Casilla marcada + Mensaje Mina Detectada
17	1. Iniciar juego 2. Introducir Nivel	2	Cerrado	Si	Casilla desmarcada	Casilla desmarcada + Mensaje No la

	3. Introducir Jugada					desmarques, es una Mina
18	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	Cerrado	Si	Casilla abierta, por lo tanto pierdes	Casilla abierta + Mensaje Has Perdido

Escenario ID 03_03		Basic Flow				
Caso concreto		Casilla marcada (únicamente puedes marcar si hay bombas)				
TC ID	Test Steps	Test Data (Op)	Estado	Mina	Resultado esperado	Resultado obtenido
19	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	Marcado	Si	Casilla marcada	Casilla marcada + Mensaje Mina Detectada
20	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	Marcado	Si	Casilla desmarcada	Casilla marcada + Mensaje No la desmarques, es una Mina
21	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	Marcado	Si	Casilla marcada	Casilla abierta + Mensaje Has perdido

Escenario ID 03_04		Basic Flow				
Caso concreto		Casilla abierta (únicamente puede quedar abierta y seguir jugando si no hay bomba)				
TC ID	Test Steps	Test Data (Op)	Estado	Mina	Resultado esperado	Resultado obtenido
22	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	Abierto	No	Casilla abierta + Mensaje no se puede marcar	Casilla abierta + Mensaje Incorrecto
23	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	Abierto	No	Casilla abierta + Mensaje no se puede desmarcar	Casilla abierta + Mensaje Incorrecte
24	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	Abierto	No	Casilla abierta + Mensaje Casilla ya abierta	Casilla abierta + Mensaje Casilla ya destapada

Escenario ID 04		Basic Flow + Alternate flow 1				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	4	-	-	Preguntar por confirmación	Preguntar por confirmación

Escenario ID 05		Basic Flow + Alternate flow 2				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	0	-	-	Pedir opción de nuevo	Te pide opción de nuevo
02	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	5	-	-	Pedir opción de nuevo	Te pide opción de nuevo
03	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	-1	-	-	Pedir opción de nuevo	Te pide opción de nuevo
04	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	100	-	-	Pedir opción de nuevo	Te pide opción de nuevo
05	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3.0	-	-	Pedir opción de nuevo	Mensaje de caracter inválido + te pide opción de nuevo
06	1. Iniciar juego 2. Introducir Nivel 3. Marcar [1][1] 4. Introducir Jugada	tres	-	-	Pedir opción de nuevo	Mensaje de caracter inválido + te pide opción de nuevo

Escenario ID 06		Basic Flow + Alternate flow 3				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	0	1	Mensaje fila incorrecta	Mensaje números incorrectos + pide fila de nuevo
02	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	6	1	Mensaje fila incorrecta	Mensaje números incorrectos + pide fila de nuevo
03	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	0	Mensaje columna incorrecta	Mensaje números incorrectos + pide fila de nuevo
04	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	6	Mensaje columna incorrecta	Mensaje números incorrectos + pide fila de nuevo
05	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	0	0	Mensaje fila y columna incorrectas	Mensaje números incorrectos + pide fila de nuevo
06	1. Iniciar juego 2. Introducir Nivel 3. Marcar [1][1] 4. Introducir Jugada	2	6	6	Mensaje fila y columna incorrectas	Mensaje números incorrectos + pide fila de nuevo

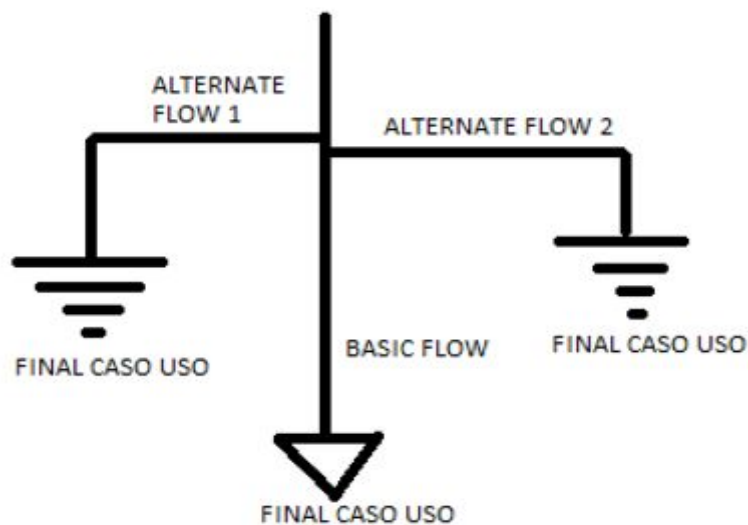
Escenario ID 07		Basic Flow + Alternate flow 4				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	3.0	-	Mensaje fila inválida	Mensaje error caracter inválido + pide opción de nuevo
02	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	Tres	-	Mensaje fila inválida	Mensaje error caracter inválido + pide opción de nuevo
03	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	_3	-	Mensaje fila inválida	Mensaje error caracter inválido + pide opción de nuevo
04	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	∞ (INF)	-	Mensaje fila inválida	Mensaje error caracter inválido + pide opción de nuevo

05	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	a3	-	Mensaje fila inválida	Mensaje error caracter inválido + pide opción de nuevo
06	1. Iniciar juego 2. Introducir Nivel 3. Marcar [1][1] 4. Introducir Jugada	2	a 3	-	Mensaje fila inválida	Mensaje error caracter inválido + pide fila de nuevo

Escenario ID 08		Basic Flow + Alternate flow 5				
TC ID	Test Steps	Test Data (Op)	Test Data (Fila)	Test Data (Col)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	1	3.0	Mensaje columna inválida	Mensaje error caracter inválido + pide opción de nuevo
02	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	3	1	Tres	Mensaje columna inválida	Mensaje error caracter inválido + pide opción de nuevo
03	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	_3	Mensaje columna inválida	Mensaje error caracter inválido + pide opción de nuevo
04	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	1	1	∞ (INF)	Mensaje columna inválida	Mensaje error caracter inválido + pide opción de nuevo
05	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	1	a3	Mensaje columna inválida	Mensaje error caracter inválido + pide opción de nuevo
06	1. Iniciar juego 2. Introducir Nivel 3. Introducir Jugada	2	1	a 3	Mensaje columna inválida	Mensaje error caracter inválido + pide fila de nuevo

Caso de uso Registrar puntuación: hemos introducido diferentes strings para comprobar si se apuntaban bien en el fichero Puntuaciones.txt.

Test Case ID	Registrar Puntuación		
Descripción	Al finalizar una partida el sistema pregunta por tu nombre para registrar la puntuación en Puntuaciones.txt		
Módulo	Buscaminas		
Hecho por	Daniel y Angel	Fecha	12/12/2020



Basic flow: nombre y puntuación registrados correctamente.

Alternate flow 1: nombre registrado incorrectamente.

Alternate flow 2: error al guardar.

Foto 4 Diagrama de flujo de Registrar Puntuación

Escenario ID 09		Basic Flow		
TC ID	Test Steps	Test Data (Nombre)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	Prueba	Se guarda "Prueba" correctamente	Se guarda "Prueba" correctamente
02	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	111	.Se guarda "111" correctamente	.Se guarda "111" correctamente
03	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	111Prueba	Se guarda "111Prueba" correctamente	Se guarda "111Prueba" correctamente

04	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	3.0	Se guarda "3.0" correctamente	Se guarda "3.0" correctamente
05	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	\$\$\$\$	Se guarda "\$\$\$\$" correctamente	Se guarda "\$\$\$\$" correctamente
06	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	:('	Se guarda ":(correctamente	Se guarda ":(correctamente

Escenario ID 10		Basic Flow + Alternate Flow 1		
TC ID	Test Steps	Test Data (Nombre)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	🔥	Se guarda "🔥" correctamente	Se guarda "?"
02	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	A ngel	.Se guarda " A ngel" correctamente	.Se guarda "??ngel"
03	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	ᄁ	Se guarda "ᄁ" correctamente	Se guarda "?"

Escenario ID 11		Basic Flow + Alternate Flow 2		
TC ID	Test Steps	Test Data (Nombre)	Resultado esperado	Resultado obtenido
01	1. Iniciar juego 2. Introducir Nivel 3. Insertar jugada 4. Perder partida	∞ (INF)	Se guarda "∞" correctamente	Se guarda " ". *

Cuando escribimos ∞ nos referimos a un número muy muy grande. En este caso en el fichero desde el eclipse se ve una línea en blanco la cual puedes borrar, pero si abres el fichero con el bloc de notas descubres que no se ha borrado y el número gigante sigue ahí.

Automatización

Para realizar automated testing hemos creado un script en el cual hemos copiado el main original para que el flujo de ejecución del programa sea lo más parecido al original, y le hemos añadido mock objects para simular los inputs del usuario y poder realizar las partidas.

Para poder simular las entradas por teclado, hemos tenido que crear unos cuantos mock objects, de los cuales hablaremos posteriormente.

En el código que hemos testado, los mock objects estaban mal creados y mal definidos, por lo que no hemos podido aprovechar nada de estos, así que hemos creado una interfaz llamada MainAux, sobre la cual hemos definido los diferentes mock objects. Posteriormente hemos creado una clase MainAuxMock sobre la cual hemos definido los valores que deberán pasar los mock objects. Por último y no menos importante, hemos tenido que crear una pequeña función en la clase main original llamada setMainMock(MainAux m) para poder crear los mockObjects sobre la clase main.

Después de definir e implementar los mock Objects, hemos tenido que crear una función repartirBombasManual, (función que ya implementamos en nuestra práctica 1) para poder realizar testing sobre los resultados obtenidos por las partidas jugadas. Para poder empezar a colocar las bombas utilizando esta función, nos hemos tenido que ir al constructor de Tablero y comentar la llamada a la función crearMinas, (que pone las bombas de forma aleatoria), y hemos incluido una llamada a la función repartirBombasManual.

Lo último que hemos tenido que crear ha sido una sobrecarga del constructor de la clase Tablero para poder tener un orden de las partidas que se están realizando para el posterior testing, y el método setNivel de la clase Tablero.

Los mock objects creados son los siguientes:

- **pasarNivel(int partida):** Esta función devuelve un nivel dependiendo de la partida que se pasa como parámetro. Lo utilizamos para simular la selección del nivel de la partida seleccionada.
- **pasarJugada(int partida):** Esta función devuelve un array de arrays de 3 posiciones. Está formada por un gran switch donde dependiendo de la partida en la que estemos, se asignan unas jugadas u otras. Cada uno de estos arrays de 3 posiciones simulan la jugada (abrir casilla, marcar casilla...), la fila seleccionada y la columna seleccionada sobre las que se van a aplicar dicha jugada.
- **pasar Bombas(int partida):** Esta función devuelve un array de arrays de 2 posiciones. Está formado por un gran switch donde dependiendo de la partida en la que estemos, se asignan las bombas en unas coordenadas u otras. Estas posiciones serán las que se pasen a la función repartirBombasManual.
- **pasarNombre(int partida):** Esta función devuelve un strings, dependiendo del número de partida pasada como parámetro. Los strings que contiene el array de strings simulan los diferentes nombres que se escribirán en el fichero de puntuaciones al finalizar la partida.
- **pasarPartida(int partidaID):** Esta función devuelve el número de partida actual para que las demás funciones sepan en qué partida estamos en cada momento;

Para poder realizar las diferentes partidas hemos creado un nuevo fichero testMain.java (el cual se encuentra en <https://github.com/1494116/Buscaminas> en la ruta src/view/testMain.java) donde se ejecutarán todos seguidos. Para cada partida hemos copiado el main del programa y le hemos hecho algunos cambios.

Las partidas se crean según un atributo de ese “main” llamado num_partida. Este atributo es el que le pasamos a las funciones que hemos explicado arriba. De esta manera, las partidas tienen niveles diferentes, se introducen jugadas diferentes, las bombas están en casillas diferentes y guardamos nombres diferentes en el fichero Puntuación.txt.

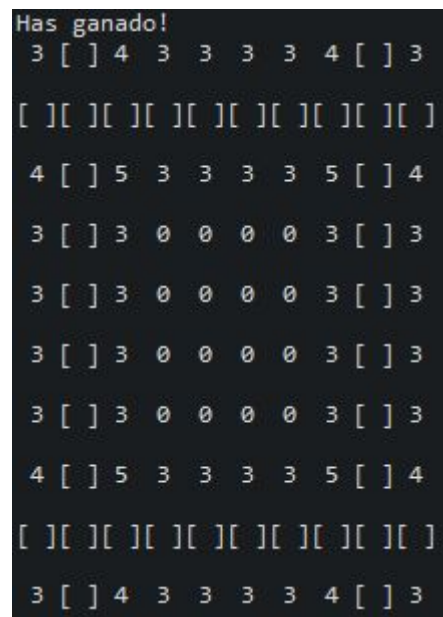
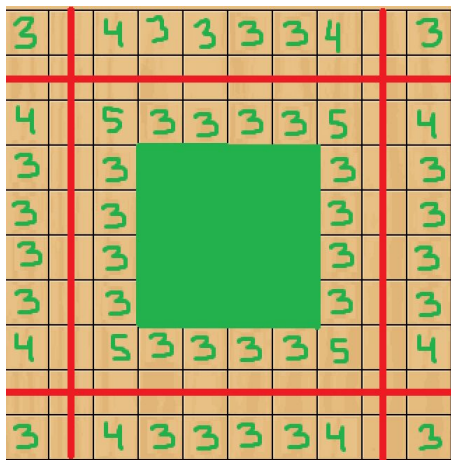
Dentro de ese “main” se encuentran todos los assertEquals() con los que hacemos las comprobaciones de cada partida. Comprobamos que el estado de una casilla después de introducir una jugada determinada sea el esperado y que el nivel de la partida coincida con el nivel introducido.

Gracias a estas comprobaciones podemos mostrar cómo el programa no funciona de la manera que nosotros consideramos correcta (siguiendo el funcionamiento del Buscaminas de Google), tal y como se ha mencionado, antes podemos abrir casillas con bandera cuando esa acción no se debería de poder realizar.

Para realizar la automatización hemos realizado un total de 14 partidas donde en cada una se comprueban diferentes aspectos:

- **Partida 0: Perder al principio tras input erróneo nivel 2:** El objetivo de esta partida es introducir un valor de una fila fuera de rango y posteriormente realizar una jugada correcta. De esta forma se comprueba si si podemos perder tras realizar la primera jugada correcta y si el flujo del programa sigue siendo correcto tras realizar un movimiento erróneo.
- **Partida 1: Partida perdida nivel 3:** Esta partida es muy parecida a la anterior, pero en este caso comprobamos que el juego funciona al utilizar otro nivel de dificultad diferente, y si el perder partida funciona tras unas cuantas jugadas.
- **Partida 2: Partida completa ganada nivel 2:** En esta partida intentamos hacer una simulación real de una partida. De hecho para realizar esta partida, uno de los dos miembros del grupo puso las bombas sin que el otro integrante lo supiera y posteriormente, este iba jugando para intentar ganar la partida desconociendo donde se encuentran las bombas. Paralelamente, se iba registrando sus jugadas para poder hacerlas de manera automática posteriormente. Mediante esta partida se comprueba que el desarrollo normal del programa es el correcto, así como las funciones de marcar una casilla.
- **Partida 3: Salir sin empezar:** En este caso comprobamos que la opción de salir del juego perteneciente al menú principal funciona correctamente. Cabe destacar que hemos tenido que comentar una parte del código del “main” para que se pudiesen ejecutar el resto de tests.
- **Partida 4: Salir a mitad de partida:** En esta partida el objetivo es empezar a jugar de forma casual y en un momento determinado de la partida, seleccionar la opción de salir en el input de seleccionar jugada. De ese modo comprobamos que el flujo alternativo de salir en mitad de la partida funciona correctamente: tanto la propia opción como su posterior confirmación.

- **Partida 5: Partida perdida con selecciones erráticas nivel 1:** El objetivo de esta partida es forzar a que aparezcan los mensajes de error del juego. Esto se logra ya sea introduciendo niveles de dificultad inexistentes, filas/columnas fuera de rango etc. Posteriormente intentamos jugar de forma normal para comprobar si el hecho de habernos equivocado afecta a la posterior ejecución.
- **Partida 6: Nivel 4 ganada:** Partida normal donde lo que se busca es comprobar la funcionalidad del nivel 4.
- **Partida 7: Partida con todo bombas:** El objetivo de poner todas las casillas con bombas es ver si aparece algún tipo de excepción o problema en el desarrollo del programa.
- **Partida 8: Ganar partida con un click:** Se rellena todo el campo de bombas salvo 1 casilla, y posteriormente se abre esa casilla. El objetivo de esta partida es localizar algún tipo de error en el flujo normal del programa debido a que es una situación algo inusual.
- **Partida 9: Partida ganada con bombas en ‘cuadrado’ para comprobar situaciones conflictivas:** En esta partida las bombas están colocadas de forma estratégica para comprobar que la función de calcular los vecinos funciona correctamente en todos los bordes (fotos). Posteriormente se gana la partida.



- **Partida 10: Partida comprobación caso especial cerradoNoBomba(M(*),D,A), cerradoSiBomba(M,D):** Aunque el nombre es poco claro, es muy sencillo de entender: En esta partida comprobamos dos casos ‘especiales’ que no siempre están controlados. Estas comprobaciones las haremos de forma individual, es decir, marcamos una casilla que no tiene bomba, desmarcamos otra casilla diferente que no tiene bomba y así todas. No significa que primero marquemos, luego desmarcamos y finalmente abramos la misma casilla. Para todas las siguientes partidas usaremos la misma distribución de bombas, una diagonal de punta a punta del tablero({1,1}, {2,2}, etc..). En el primer caso, se intenta marcar(M), desmarcar(D), y abrir(A) casillas cerradas que no contienen bomba. Aquí se produce un error conceptual en el caso de marcar. No se puede

marcar una casilla que no tenga una bomba. Esto se puede utilizar para “hackear” la partida ya que podemos marcar todas las casillas del tablero y en las que no salga un mensaje de error y se marque sabremos que hay una mina.

En el segundo caso, intentamos marcar y desmarcar casillas cerradas las cuales contienen bombas, no ponemos el de abrir ya que este caso lo comprobamos más adelante.

El funcionamiento es correcto salvo en Marcar (foto).

```
Jugada: 312
B 2 [ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]

Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
1

Elige una fila:
2

Elige una columna:
1

No hay mina!
```

- **Partida 11: Partida comprobación caso Marcado(M,D(*)):** En esta partida comprobamos que si una casilla ya ha sido marcada (por lo tanto tendrá bomba) podemos marcarla de nuevo y desmarcarla. El funcionamiento es correcto para marcar pero no para desmarcar. No se puede desmarcar una casilla marcada ya que, como tiene una mina, aparece un mensaje diciendo “No la desmarques, es un MINA” (foto).

```
Jugada: 111
B [ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]

Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
2

Elige una fila:
1

Elige una columna:
1

No la desmarques, es una MINA!
```


- **Partida 12: Partida comprobación caso Abierto(M,D,A) CerradoSibomba(A):** En esta partida comprobamos que si una casilla ya ha sido abierta, podemos marcarla, desmarcarla y volverla a abrir. El funcionamiento es todo correcto, para finalizar la partida haremos la comprobación que quedaba pendiente de antes, abrir una casilla cerrada que tiene una bomba.
- **Partida 13: Partida comprobación caso Marcado(A(*)):** En esta partida comprobamos el caso que quedaba pendiente, abrir una casilla ya marcada. Como ya se ha mencionado a lo largo de este informe, si una casilla está marcada no se debería de poder abrir, habría que desmarcarla primero.

```
Jugada: 111
B [ ][ ][ ][ ]

[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]

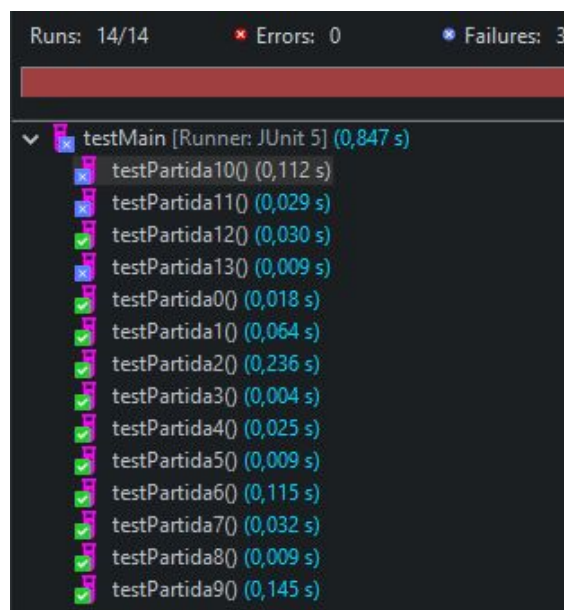
Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
3

Elige una fila:
1

Elige una columna:
1

MINA!
```

Tras realizar todas estas partidas y poner sus asserts correspondientes, hemos ejecutado nuestro test y nos hemos encontrado con 3 failures.



A continuación analizaremos el motivo de los failures en las partidas 10,11 y 13.

- **Partida 10:** En esta partida el fallo se ha producido al comprobar que si una casilla que no contenía una bomba la intentamos marcar, el valor del atributo 'marcado' seguía siendo false, por tanto no quedaba marcada.
- **Partida 11:** En esta partida se ha producido un fallo al comprobar si podíamos desmarcar una casilla que está marcada. El resultado esperado es que el atributo 'marcado' tenga un valor false, sin embargo el resultado obtenido es que sigue siendo true, por tanto la casilla no queda desmarcada.
- **Partida 13:** En esta partida aparece el ya mencionado fallo conceptual donde se puede abrir una casilla ya marcada. Nosotros lo consideramos fallo porque afecta a la jugabilidad ya que puedes abrir sin querer una casilla donde nosotros sabemos que hay una bomba porque la hemos marcado.

Exploratory testing

Ataque 1: insertar inputs para que todos los mensajes de error se ejecuten 1 vez.

Hay 6 inputs diferentes, para cada uno probaremos ciertos valores para ver si los errores están controlados.

Nivel. Int entre 1 y 5. Hemos probado ints menores que 1 y mayores que 5, no muestra mensaje de error, simplemente te pide el nivel de nuevo. Si introduces un float (3.0), un string o caracteres especiales se muestra mensaje de carácter no válido.

Jugada: Int entre 1 y 4. Hemos probado ints menores que 1 y mayores que 4, no muestra mensaje de error, simplemente te pide la jugada de nuevo. Si introduces un float (3.0), un string o caracteres especiales se muestra mensaje de carácter no válido.

Fila/Columna: Int entre 1 y 20, dependiendo del nivel. Hemos probado ints menores que 1 y mayores que 20, al estar fuera de límites se muestra un mensaje de "Números incorrectos", pero no te dice si el número incorrecto es la fila o la columna. Si introduces un float (3.0), un string o caracteres especiales se muestra mensaje de carácter no válido y te devuelve a la selección de jugada.

Salir: String S,s o N,n. Hemos probado S y s, sale del juego correctamente. También N y n, el juego continúa correctamente. Si introduces un int, float o un string diferente a S o s, el juego continúa.

Nombre. String. Hemos probado palabras normales (Prueba) y se guarda correctamente. Ints y floats normales también funcionan. Caracteres especiales como \n también se guardan. Si pones 2 strings (palabra1 palabra2), solo se guarda la primera palabra.

Tras analizar un poco más a fondo el código del juego, hemos encontrado la razón por la cual el ataque no encuentra ningún error: se utiliza la función NextInt(), la cual 'rechaza' todos los tipos de datos que no sean de tipo entero lanzando una excepción y volviendo a pedir el dato hasta que sea del tipo correcto.

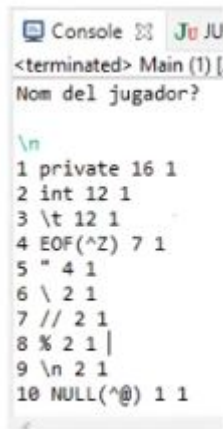
Ataque 2: insertar como input conjuntos de caracteres especiales en campos string.

En el programa, hay 2 inputs de tipo string: Cuando hemos acabado la partida y tenemos que insertar el nombre del jugador, y el segundo se produce cuando queremos salir del juego, que se nos pedirá confirmación para salir.

Como el juego que estamos testeando está realizado en lenguaje Java, hemos de probar a introducir los caracteres especiales del lenguaje Java.

Para realizar este ataque hemos probado con palabras reservadas de Java y con caracteres especiales. A continuación listamos todos los que hemos probado:

\, ", EOF(^Z), NULL(^@), //, \n, int, private, break, %, private, \t.



```
Console JU JU
<terminated> Main (1) [
Nom del jugador?

\n
1 private 16 1
2 int 12 1
3 \t 12 1
4 EOF(^Z) 7 1
5 \" 4 1
6 \ 2 1
7 // 2 1
8 % 2 1 |
9 \n 2 1
10 NULL(^@) 1 1
```

Cómo se puede apreciar en los resultados obtenidos, tanto si insertamos como nombre un carácter reservado como un carácter especial, el resultado esperado es el correcto y el programa funciona correctamente.

Esto se debe a que convierten todo lo que les sea escrito por pantalla en un string, ya sea un int, un float etc. De esta forma todo lo que se introduce va a ser convertido siempre en string, lo que implica que estos caracteres especiales no provoquen comportamientos no deseados en el programa.

Cabe destacar que si se introducen caracteres especiales como emoticonos **A** el resultado se guarda como ?, pero no provoca un error.

Ataque 3: desbordar el buffer de input en campos string o parámetros.

En el programa hay 6 inputs, 2 strings y 4 ints.

Para este ataque simplemente hemos introducido un número de muchísima longitud y hemos visto cómo reacciona el juego.

Nombre: te deja guardar el número, aunque la siguiente vez que juegues y tenga que leer el fichero no mostrará el número (habrá una fila vacía), en cambio, si abres el fichero de Puntuaciones sí que podrás ver el número. También, si intentas borrar esa línea vacía en el Eclipse no se borrará el número, seguirá apareciendo si lo abres con el bloc de notas.

Para el resto de inputs, si introduces ese número gigante simplemente salta un mensaje de error diciendo que introduzcas otro número.

Ataque 4: repetir el mismo input o serie de inputs muchas veces.

Para este error hemos seleccionado los inputs de seleccionar jugada, fila y columna, ya que son los que se usan más.

Tal y como está hecho el programa podemos meter en 1 línea de la consola todos los inputs que queramos hacer a lo largo de la ejecución del juego y estos se introducirán por orden, por ejemplo si al inicio del juego introduces 1 3 2 4 se seleccionará el nivel 1, jugada 3, fila 2 y columna 4.

Hemos usado esta forma para repetir el input de estos 3 campos.

Hemos probado estas combinaciones: 1 5 1, 2 5 1, 3 1 1, 4 n.

Ataques propuestos por el equipo

Ataque 1: Interrumpir el programa (CTRL+Z) en mitad de la ejecución.

Para realizar este ataque, ejecutaremos el programa de forma habitual y en un cierto momento cuando se nos pida insertar algo por pantalla, interrumpiremos la ejecución del programa.

Hemos realizado este ataque en todos los lugares donde hay que introducir algo por pantalla: (insertar nivel, insertar jugada, insertar fila/columna, insertar nombre).

Al comprobar este caso hemos visto que sucede algo extraño, y es que si nosotros hacemos click en algo que no sea la consola y posteriormente volvemos a clicar en la consola para escribir, vemos que aparece un error de elemento inexistente. Este error varía ligeramente si se produce en el momento en el que el programa espera un int o un string.

```
Selecciona un nivel:
Pulsa 1: Nivel Facil
Pulsa 2: Nivel Medio
Pulsa 3: Nivel Dificil
Pulsa 4: Nivel Muy Dificil
Pulsa 5: Salir
Exception in thread "main" java.util.NoSuchElementException
    at java.base/java.util.Scanner.throwFor(Scanner.java:937)
    at java.base/java.util.Scanner.next(Scanner.java:1594)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
    at mvc.view.Main.main(Main.java:37)
```

Seguro que quieres salir? (S/N)

```
Exception in thread "main" java.util.NoSuchElementException
    at java.base/java.util.Scanner.throwFor(Scanner.java:937)
    at java.base/java.util.Scanner.next(Scanner.java:1478)
    at mvc.view.Main.main(Main.java:131)
```

Ataque 2: Mostrar el resultado de variables privadas a través del input.

Este ataque es un tipo de *code injection* mediante el cual se intenta ejecutar instrucciones en los inputs para intentar mostrar el contenido de variables privadas.

La finalidad de este ataque es que el sistema nos proporcione información sobre variables internas del código, que en este caso se traduciría en ventaja para poder ganar la partida de forma más fácil.

Hemos realizado este ataque en los 6 inputs que hay en el programa, y en ninguno de ellos el programa ha mostrado el resultado que estábamos intentando obtener al realizar el ataque.

En nuestro caso hemos probado 3 casos: intentar que el programa nos muestre si en una casilla determinada existe o no una bomba, que nos muestre el fichero de puntuaciones sin haber jugado la partida, y que nos indique el número de bombas que hay en el tablero. Debido a que todo lo que se introduzca por teclado se convierte a string directamente (en el caso de los input string) o se espera únicamente un int (en el caso de los input int), es imposible realizar este tipo de ataque en el código.

```

Elige una fila:
System.out.println(partia.getBombas());
Error: Introduce un caracter valido

Que quieres hacer? :
Pulsa 1: Marcar casilla
- - - - -
Nom del jugador?|

System.out.println(puntuacion.mostrar_Puntuaciones(partida.getNivel()));
1 System.out.println("partida.getcasilla(2,3).getMina()); 79 2
2 System.out.println(puntuacion.mostrar_Puntuaciones(partida.getNivel())); 70 2
3 System.out.println(partia.getBombas()); 1 2

```

Ataque 3: Modificar el comportamiento del código.

Este ataque es un tipo de *code injection* cuyo objetivo es utilizar los inputs para intentar modificar el comportamiento del programa y provocar errores.

La finalidad de este ataque es intentar ganar al juego haciendo trampas y de una forma deshonesta.

Hemos realizado este ataque en los 6 inputs que hay en el programa, y en ninguno de ellos el programa ha mostrado el resultado que estábamos intentando obtener al realizar el ataque.

Para realizar este ataque hemos probado los siguientes escenarios: eliminar una bomba de una casilla determinada, abrir una casilla sin haberla abierto mediante inputs y modificar las puntuaciones del fichero de puntuaciones.

Este ataque no ha alterado el funcionamiento del código, ya que los inputs están bien protegidos y, en el caso de que el input esperado sea de tipo int se nos muestra un mensaje de error, y en caso de inputs de tipo string, simplemente nos convierte la instrucción a texto así que este no puede causar ningún tipo de problema en el programa.

```

Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
partida.getCasilla(0,0).setMina(false);
Error: Introduce un caracter valido

```

```

Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
partida.getCasilla(0,0).setAbierto(true);
Error: Introduce un caracter valido

```

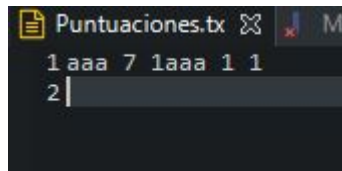
```

Que quieres hacer? :
Pulsa 1: Marcar casilla
Pulsa 2: Desmarcar casilla
Pulsa 3: Destapar casilla
Pulsa 4: Salir
puntuacion.escribitPuntuaciones(1000,partida.getNivel(),tramposo);
Error: Introduce un caracter valido

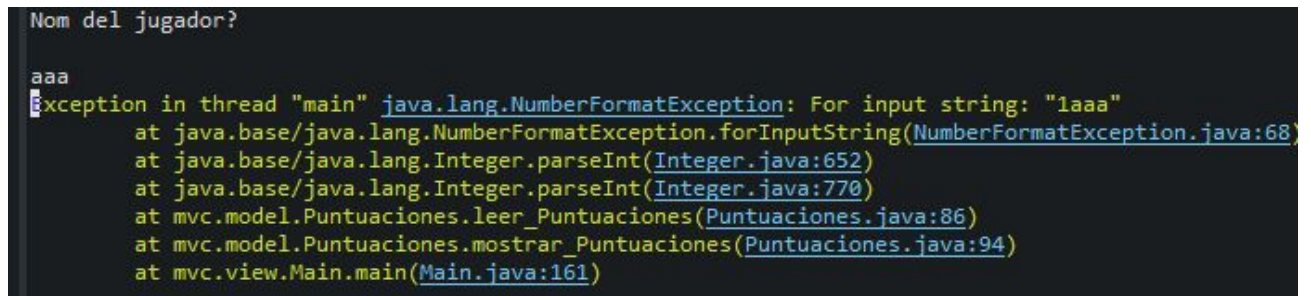
```

Problemas extra

Extra 1: nos hemos dado cuenta que si al borrar las puntuaciones manualmente no lo dejas preparado en una nueva línea vacía salta una excepción.

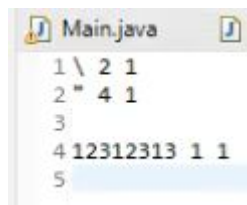


```
Puntuaciones.tx
1 aaa 7 1aaa 1 1
2 |
```

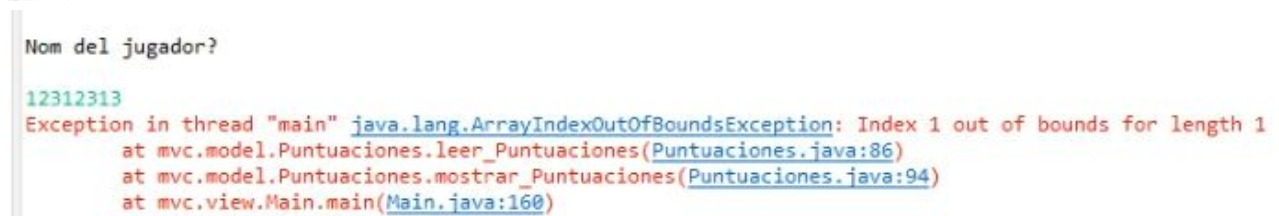


```
Nom del jugador?
aaa
Exception in thread "main" java.lang.NumberFormatException: For input string: "1aaa"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:68)
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
    at java.base/java.lang.Integer.parseInt(Integer.java:770)
    at mvc.model.Puntuaciones.leer_Puntuaciones(Puntuaciones.java:86)
    at mvc.model.Puntuaciones.mostrar_Puntuaciones(Puntuaciones.java:94)
    at mvc.view.Main.main(Main.java:161)
```

Extra 2: si introduces una línea en blanco en algún punto del fichero de Puntuaciones que no sea la última, salta una excepción.



```
Main.java
1 \ 2 1
2 " 4 1
3
4 12312313 1 1
5
```



```
Nom del jugador?
12312313
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1
    at mvc.model.Puntuaciones.leer_Puntuaciones(Puntuaciones.java:86)
    at mvc.model.Puntuaciones.mostrar_Puntuaciones(Puntuaciones.java:94)
    at mvc.view.Main.main(Main.java:160)
```

RTF

RTF Propia

Para realizar la RTF propia, realizamos los 3 documentos provistos: inspection_log, inspection_issue_log y typo_list.

En el inspection_log anotamos los datos de toda la gente que participó en nuestro rtf.

Para hacer la inspection_issue_log, nos ayudamos de las dos checklist provistas en el campus virtual así como de los reportes que hacían nuestros compañeros en la reunión. En este documento aparecen todos los defectos que hemos encontrado que NO son de tipo checkstyle.

En la typo_list hemos apuntado todos los defectos de tipo checkstyle los cuales no afectan al funcionamiento del programa.

Inspection Summary Report

Inspection Identification:

Project: Buscaminas
Inspection ID: <https://github.com/sergioux284/Buscaminas>
Meeting Date: 13/12/20

Work Product Description:

	<u>Inspectors</u>	<u>Signature</u>	<u>Preparation Time</u>
Author:	Sergio Gallo Cárdenas	1360181	2 hours
Moderator:	Raul Estrada Herrerias	1363699	2 hours
Reader:	Angel García Calleja	1490917	2,5 hours
Recorder:	Daniel Calvo Ramos	1494116	2,5 hours
Inspector:	Gerard Noguer Pagès	1459535	1 hours
Inspector:	Daniel Calvo Ramos	1494116	2,5 hours
Inspector:	Youssef Assbaghi Asbahi	1493477	1 hours
Inspector:	Angel García Calleja	1490917	2,5 hours

Inspection Data

Pages or Lines of Code:		Meeting Time:	1:47 hours
Planned for Inspection:	tot el codi	Total Planning Effort:	1 labor hours
Actually Inspected:	tot el codi	Total Overview Effort:	1 labor hours
		Total Preparation Effort:	2:30 labor hours
		Actual Rework Effort:	4:17 labor hours

Product Appraisal

ACCEPTED

☐ as is
☒ conditionally upon verification

NOT ACCEPTED

☐ reinspect following rework
☐ inspection not completed

Verifiers: Angel García Calleja & Daniel Calvo Ramos

Projected Rework Completion Date: 16/12/2020

Inspection Issue Log

Project: Buscaminas	Origin:	Requirements, Design, Construction, Testing
Inspection ID: https://github.com/sergioux284/Buscaminas	Type:	Missing, Wrong, Extra, Usability, Performance, Style, Clarity, Question
Meeting Date: 13/12/2020	Severity:	Major, minor
Recorder: Daniel Calvo Ramos	Priority:	

Defects Found: 33 Major 8, minor 25.



#	Origin	Type	Severity	Location	Description
1	D	M	m	Todo el código	Escasez de comentarios
2	D	W	m	Casilla Tablero	Atributos privados definidos como protected
3	D	W	M	Tablero Partida	Atributos privados definidos como public
4	D	M	M	Puntuaciones	Atributos privados no estan definidos (ni public/protected/private)
5	C	C	m	Puntuaciones	Nombres de variables en diferentes idiomas
6	C	M	m	Partida Casilla Puntuaciones	Variables no inicializadas
7	D	M	m	Casilla	Faltan paréntesis
8	C	M	m	Todo el código	No se usa el <i>this</i> para acceder a variables privadas
9	C	W	M		Los ficheros de test están guardados en la misma carpeta que el test desarrollado
10	C	E	m	Main	Se duplica código innecesariamente
11	R	M	M		Falta en archivo travis.yml
12	D	W	m	Tablero	Constantes no definidas como tal

13	C	W	M	Tablero	Las coordenadas (fila y columna) de bombas y casilla no coinciden (dada una casilla fila X, para acceder a la misma fila cuando estamos trabajando con bombas es fila X-1)
14	D	E	m	Tablero	Hay métodos que superan las 60 líneas de código
15	C	M	m	Puntuaciones	Los arrays no se fijan a null al acabar su ejecución
16	D	C	m	Tablero	Mensajes de error creados en diferentes idiomas.
17	D	C	m	Partida	Método mostrar() no tiene un nombre suficientemente descriptivo.
18	D	S	m	Puntuaciones	Variable de control del for no inicializada en el header
19	D	C	m	Puntuaciones	Variables puntuacion y Puntuaciones tienen nombres similares que causan confusión
20	C	S	m	Tablero	No se sigue el mismo estilo a la hora de comprobar valores booleanos en los condicionales.
21	D	E	m	Tablero	Condiciones del if redundantes debido al diseño del programa
22	D	E	m	Tablero	Funciones redundantes
23	C	S	m	Todo el código	Se utilizan prints en lugar de console.log
24	C	W	m	Main	Variables se crean e inicializan a la vez (misma línea)
25	D	C	m	Todo el código	Uso de magic numbers
26	C	M	m	Main	Defaults no retornan nada
27	C	M	M	Main Tablero	Falta de defaults en algunos switch-case
28	D	C	m	Todo el código	Nombres de parámetros y atributos idéntico que puede llegar a causar confusión

29	C	M	M	Tablero	El método crearMinas no comprueba que en la primera casilla que abres haya una bomba (perder al primer click)
30	C	P	M	Tablero	Se crean <u>variables</u> dentro del bucle
31	C	M	m	Tablero	En la función abrir casilla no se comprueba que si hay una casilla marcada
32	D	C	m	Tablero	Función crearMinas poco cohesiva
33	D	U	m	Todo el código	Código difícil de mantener para futuras versiones.

Typo List

Record any typographical errors that you find during your inspection preparation on this list, including spelling, grammatical, formatting, and style errors. These should be corrected but need not be discussed at the inspection meeting. They will not be counted as defects.

Inspector: Angel García Calleja & Daniel Calvo Ramos

Scheduled Inspection Meeting Date: 15/12/2020

Work Product Description: <https://github.com/sergioux284/Buscaminas>



Page Number	Line Number or Section	Description of Typo
Puntuaciones	13, 19, 23, 34, 38, 82, 86, 95, 97	Variable Puntuaciones no es suficientemente descriptiva
Partida	64	Método mostrar() no tiene un nombre suficientemente descriptivo.
Casilla	5, 6	Cada declaración de variable debe estar en su línea.
Casilla	22, 40, 75, 80	Las { no sigue el Kernighan and Ritchie style.
Casilla	23	Faltan paréntesis para separar las dos condiciones.
Casilla	39, 42, 43, 44, 54, 55	El nombre de la variable no sigue lowerCamelCase.
Casilla	40, 42, 43, 44	Falta un espacio después del if.
Casilla	40	Falta un espacio antes y después del ==.
Casilla	42	Faltan paréntesis para separar entre las condiciones.
Casilla	43	Falta un espacio antes y después del >.
Casilla	44, 45, 48	Falta un espacio antes y después del =.
Casilla	44	Falta un espacio antes y después del +.
Casilla	44	El if debe llevar {}.
Casilla	45	El else debe llevar {}.

Puntuación	12	Falta espacio después de >.
Puntuación	13,56	El nombre de la variable no sigue lowerCamelCase.
Puntuación	22, 34, 85, 96	Falta un espacio antes y después del =.
Puntuación	32, 36, 39, 91, 110	Se debe separar de la declaración anterior.
Puntuación	56, 86	Falta espacio después de la ,.
Puntuación	57, 105	Falta espacio antes y/o después de >.
Puntuación	60, 68, 100	Falta un espacio antes y después del +.
Puntuación	78, 98	Las { no sigue el Kernighan and Ritchie style (no va en la misma línea y está puesto debajo).
Puntuación	84	Falta espacio después del while.
Puntuación	84	Falta un espacio antes y después del !=.
Puntuación	97	Falta un espacio antes y después de :.
Puntuación	99	Falta un espacio antes y después del ==.
Puntuación	99, 105	Falta un espacio después del if.
Partida	32, 34, 68, 71,72	Falta un espacio después del if.
Partida	34, 36, 77	Falta espacio después de }.
Partida	34	El else debe llevar {}.
Partida	34, 66, 67	Falta espacio antes y después de <.
Partida	66, 67	Falta espacio después de for.
Partida	66, 67,	Falta un espacio antes y después del =.
Partida	66, 67	Falta un espacio antes y después del ;.

Partida	68, 91	Faltan paréntesis para separar entre las condiciones.
Partida	66, 67, 68, 71, 77	Falta espacio antes de {.
Partida	69, 78	Falta un espacio antes y después del +.
Partida	84, 87	Se debe separar de la declaración anterior.
Main	12, 13	Cada declaración de variable debe estar en su línea.
Main	19, 21, 22, 66, 87, 100	Falta espacio después del while.
Main	22, 63, 136	Falta espacio antes de {.
Main	32, 72	Falta espacio después de switch.
Main	45, 71, 80, 93, 106, 111, 124	Falta un espacio después del if.
Main	50	Falta un espacio antes del else.
Main	48, 126, 147	Falta un espacio antes y después del =.
Main	71	Falta un espacio antes y después del <.
Main	71, 145	Falta un espacio antes y después del >.
Main	79, 84, 92, 97, 105, 110	Falta un espacio antes y después del -.
Main	105	{ debe estar en la misma línea que el else.
Main	136, 157	Falta espacio después del catch.
Main	152	Falta espacio después de la ,.
Main	136, 157	Falta espacio antes de }.
Tablero	10, 11, 12, 13, 259	El nombre de la variable no sigue lowerCamelCase.
Tablero	11, 19	Falta un espacio antes y después del =.

Tablero	28, 36, 38, 51, 54, 58, 69, 70, 73, 76, 80, 81, 87, 88, 94, 95, 101, 102, 108, 109, 115, 116, 122, 123, 129, 130, 153, 164, 165, 170, 171, 176, 177, 182, 183, 188, 189, 194, 195, 200, 201, 206, 207, 228, 231, 239, 241, 250, 252	Falta un espacio después del if.
Tablero	30, 36, 38, 51, 54, 58, 60, 76, 77, 78, 80, 81, 87, 88, 94, 95, 101, 102, 108, 109, 115, 116, 122, 123, 129, 130, 147, 153, 161, 162, 164, 170, 171, 176, 177, 182, 183, 188, 189, 194, 195, 200, 201, 206, 207	Falta espacio antes de {.
Tablero	36, 51, 70, 76, 81, 88, 95, 102, 109, 116, 123, 130, 153, 165, 171, 177, 183, 189, 195, 201, 207	Falta un espacio antes y después del ==.
Tablero	42, 45, 60, 231, 233, 241, 243, 252, 254	Falta espacio después de }.
Tablero	51, 69, 76, 80, 81, 87, 88, 94, 95, 101, 102, 108, 109, 115, 116, 122, 123, 129, 130, 164, 170, 176, 182, 188, 194, 200, 206	Faltan paréntesis para separar entre las condiciones
Tablero	51, 76, 80, 81, 87, 88, 94, 95, 101, 102, 108, 109, 115, 116, 122, 123, 129, 130, 164, 170, 176, 182, 188, 194, 200, 206	Falta un espacio antes y después del &&.
Tablero	68	Cada declaración de variable debe estar en su línea.
Tablero	69, 80, 87, 97, 101, 108, 115, 122, 129, 164, 170, 176, 182, 188, 194, 200, 206	Falta un espacio antes y después del >=.
Tablero	77, 161, 220, 221	Falta espacio después de for.
Tablero	77, 150, 161, 220, 221	Falta un espacio antes y después del =.
Tablero	77, 101, 108, 115, 122, 129, 147, 161, 188, 194, 200, 206, 220, 221, 231, 241, 252	Falta un espacio antes y después del <.
Tablero	77, 161	Falta un espacio antes y después del ;.
Tablero	78, 162	Falta espacio después de switch.

Tablero	80, 81, 82, 87, 88, 89, 94, 95, 101, 108, 115, 122, 123, 124, 129, 130, 131, 164, 165, 166, 170, 171, 172, 176, 177, 182, 188, 194, 200, 206	Falta un espacio antes y después del -.
Tablero	81, 102, 123, 130	La línea supera los 100 caracteres.
Tablero	82, 89, 103, 110, 117, 124, 131,	Falta un espacio después de ,.
Tablero	101, 102, 103, 108, 109, 110, 115, 116, 117, 122, 123, 124, 129, 130, 131, 182, 183, 188, 189, 194, 195, 200, 201, 206, 207, 259	Falta un espacio antes y después del +.
Tablero	147	Falta espacio después del while.
Tablero	149, 150	Falta un espacio antes y después del *.
Tablero	165, 171, 177, 183, 189, 195, 201, 107	El if debe llevar {}.
Tablero	228, 259, 262, 281	Se debe separar de la declaración anterior.