Práctica final Tema 4

Descarga del proyecto

Descarga a tu PC el nuevo proyecto SuperBuscaMinas que está en el repositorio GitHub.

https://github.com/nachoinformatica/superbuscaminas00

Configura tu proyecto local para que apunte a un repositorio local git, y a un repositorio remoto **privado** en tu GitHub, que llamarás **spb_nombre_apellidos** (por ejemplo spb_nacho_lorenzo).

Asegúrate de que no apunta a mi repositorio original. Para ello visualiza los repositorios remotos de tu proyecto desde la consola git y pega pantallazo en un pdf.

Desde tu GitHub concede permisos a mi usuario **nachoinformatica** para que yo pueda acceder a tu nuevo repositorio.

Importante

Al finalizar la práctica es obligatoria la entrega de un documento PDF con vuestro nombre y apellidos.

Cuando se indique “hacer un pantallazo” se deberá pegar en el documento, con un pequeño texto explicativo e indicar la tarea a la que se refiere.

Cada vez que uséis el menú de refactorización haced pantallazo de la opción usada.

Cuando se indique hacer commit, deberá ser un commit y un push al repositorio remoto. Recordad que los tags no suben al remoto hasta que no se haga un Team, Remote, Push Tags.

Para simular una situación real, **lo primero que hay que hacer es documentar** cada uno de los métodos que no lo estén. Y, los que estén documentados, que cumplan el **estándar** para en el futuro generar un **javadoc** completo. Cada nuevo método que se vaya generando deberá tener su documentación.

Tarea 1

El método ***imprimeMatrizBotones*** está pendiente de hacer. Aprovecharemos para programarlo. Se trata de recorrer una matriz de ***BotonMina*** e ir imprimiendo por pantalla sus celdas.

Probad el método imprimiendo por la consola una matriz en algún momento del juego.

Pantallazo de la salida por consola.

Una vez probada la impresión, comentad esa línea de prueba (para que no se ejecute).

Haced el commit y cread un **tag** con la descripción "Método imprimeMatrizBotones creado y probado".

Tarea 2

Viendo el código, nos parece interesante crear una clase estática ***Utilidades***, y mediante refactorización mover el método ***imprimeMatrizBotones*** de ***Tablero*** a la nueva clase ***Utilidades***.

Pantallazo del tipo de refactorización y al pdf.

Haced el commit y crear un **tag** con la descripción "Método imprimeMatrizBotones se pasa a la nueva clase Utilidades".

Tarea 3

El método ***inicializaMatrizBotones*** tiene quizá demasiadas líneas. Hemos pensado que se puede pasar la creación de minas a un método nuevo llamado ***plantaMinas***, que tenga como parámetro el número de minas a generar.

Pantallazo del tipo de refactorización y al pdf.

Haced el commit con la descripción "Nuevo método plantaMinas que genera tantas minas como se indique".

Tarea 4

En el método ***isJuegoFinalizado*** se repite el código de detención del tiempo, el mensaje de salida, y la salida. Hay que crear un método llamado ***mensajeFinJuego*** que tenga como parámetro de entrada el texto mensaje que se muestra en el JOptionPane.

Podríais combinar varios tipos de refactorización, pero quiero que lo hagáis a mano y por qué en este caso es mucho más rápido no usar refactorización.

Poned comentarios en el pdf.

Haced el commit con la descripción "Nuevo método mensajeFinJuego para detener el tiempo y mostrar el mensaje correspondiente".

Tarea 5

En la clase SuperBuscaMinas, dentro del método ***initialize*** tenemos las coordenadas y tamaño del formulario

```
frmBuscaMinas.setBounds(0, 0, 500, 500);
```

El analista nos dice que esos 500 y 500 (ancho y alto del formulario) deben ser constantes, con los nombres WINDOW_HORIZONTAL_SIZE y WINDOW_VERTICAL_SIZE, respectivamente.

Pantallazo del tipo de refactorización y al pdf.

Haced el commit con la descripción "Nuevas constantes con ancho y alto de la ventana".

El analista se pasa de perfeccionista y piensa que en nuestro programa el formulario siempre va a contener un Tablero, así que decide que estas constantes deberían estar en la clase Tablero y llamarse de forma estática desde el método initialize de la clase SuperBuscaMinas.

Pantallazo del tipo de refactorización y al pdf.

Haced el commit con la descripción "Las nuevas constantes se pasan a la clase Tablero".

Tarea 6

Observamos que el método ***recursivoDestapaCeldasAdyacentes*** hace lo mismo según el caso (celda superior izquierda, celda superior, celda superior derecha…) así que vamos a intentar generalizar un método ***destapaBoton*** que se encargue de cambiar el aspecto, a número o a pulsado, y que se llame en cada caso (celda superior izquierda, celda superior, celda superior derecha…) con los parámetros correctos fila y columna.

En este caso podéis combinar algo de refactorización y algo de modificación manual. Indicar cómo lo habéis hecho en el pdf.

Pantallazo del tipo de refactorización y/o modificación manual y al pdf.

Aprovechad también (a mano) para que no sean dos if seguidos que siempre obliga a evaluar ambas condiciones. En el nuevo método quedaría perfecto un if else…

Si es valor NUMERO cambia aspecto a NUMERO

En otro caso Si es valor VACIO cambia aspecto a PULSADO

Haced el commit con la descripción "Nuevo método destapaBoton que se encarga de cambiar el aspecto del botón de una fila y columna concreta".

Tarea 7

Por último, vamos a crear una **nueva rama llamada Tarea 7**. En esa rama vais a realizar la última refactorización.

En la clase Tablero se desea convertir la clase anónima que se usa para el listener del ratón, a una clase interna llamada ***CeldaListener***. El código original es:

```
// Añado el listener de ratón
celda.addMouseListener(new MouseListener() { .  .  .
```

Y el nuevo código sería:

```
private final class CeldaListener implements MouseListener {
	private final BotonMina celda;

	private CeldaListener(BotonMina celda) {
		this.celda = celda;
	}

	@Override
	public void mouseReleased(MouseEvent e) {
		// Empieza el tiempo cuando haga click por primera vez
		if (!panelTiempo.isContadorIniciado()) {
			panelTiempo.contar();
		}
                    .    .    .    .    .    .

	// Añado el listener de ratón
	celda.addMouseListener(new CeldaListener(celda));
```

Pantallazo del tipo de refactorización y al pdf.

Haced el commit a la rama Tarea 7, y crear un **tag** con la descripción “Clase CeldaListener”.

Tarea Final. Genera la documentación Javadoc y súbela a GitHub. Rama master.