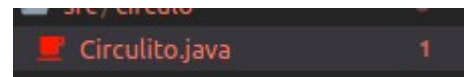


Ejercicio 1

cambiar clase circulo → circulito

```
public class Circulo {  
    private int x;  
    private int y;  
    private double radio;  
    public Circulo() {  
        // ...  
    }  
}
```



```
public static void main(String[] args) {  
    Circulito circulo = new Circulito(37,43,2.5);  
    String salida =  
        // ...  
}
```

(yo solo pulsé F2 para cambiar la clase, todo lo demás se hizo automáticamente)

cambiar obtenerarea → obtenerareacirculo

```
public double obtenerArea() {  
    return Mat.obtenerAreaCirculo(  
        // ...  
    );  
}
```

```
salida+="\nEl área es "+dosDigitos.format(circulo.obtenerAreaCirculo());  
System.out.println(salida);
```

Al igual que antes, los cambios en el main se ven reflejados automáticamente

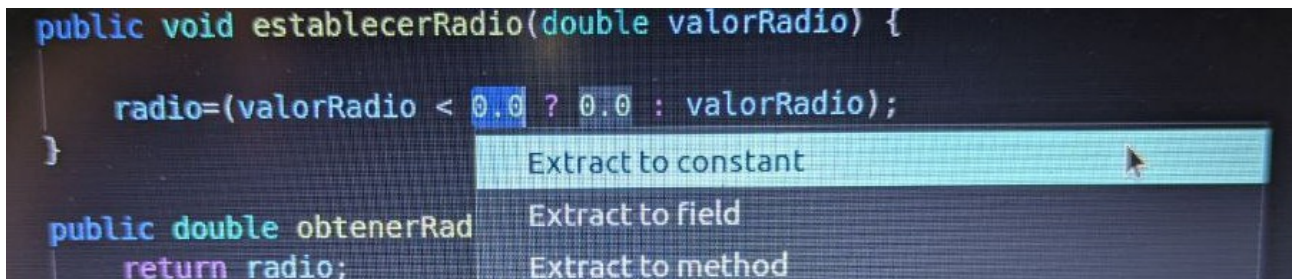
cambiar x e y por coordenada x y coordenada y

```
private int x;  
private int y;  
private double radio;  
    // ...  
}
```

```
private double coordenadaX;  
private double coordenadaY;  
    // ...  
}
```

refactorizar 0.0 a una constante

```
public void establecerRadio(double valorRadio) {  
    radio=(valorRadio < 0.0 ? 0.0 : valorRadio);  
}  
  
public double obtenerRadio()  
    return radio;
```



```
private static final double LIMITERADIO = 0.0;  
private int coordenadaX;
```

Cambiar establecerX y obtenerX por setX y getX (lo mismo para Y)

```
coordenada setY|  
}  
Enter to Rename, Shift+Enter to Preview
```

```
return getX|  
}  
Enter to Rename, Shift+Enter to Preview
```

He hecho el proceso con los dos obtener y los dos establecer, pero no pongo capturas para no llenar el documento, el proceso es el mismo, en el archivo main se modifican todos automaticamente.

```
circulo.setX(35);  
circulo.setY(20);  
circulo.establecerRadio(10);
```

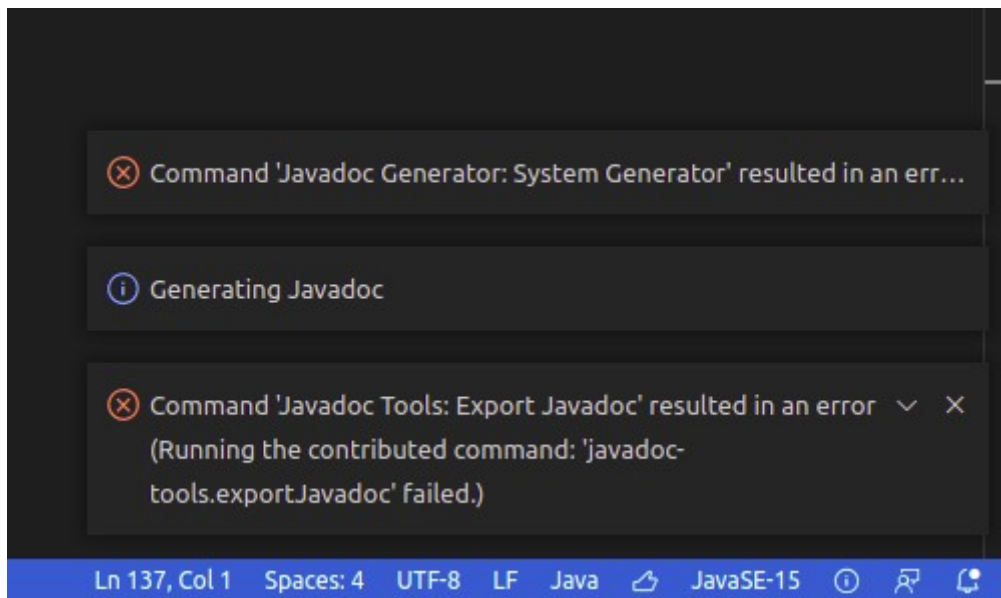
```
"La coordenada X es "+circulo.getX()+  
"\nLa coordenada Y es "+circulo.getY()+  
"\nEl radio es "+circulo.obtenerRadio();
```

Dados mis nulos conocimientos en java, se me hace bastante complejo ponerme a pensar en encapsular, de todos modos, creo que se trata de hacer que ciertas variables sean “ocultadas”, de manera que solo sean accesibles desde métodos internos del objeto

En este caso creo que ya están en private, por lo que no se si se requiere hacer algo mas.

Ejercicio 2

Tanto con el javadoc tools como con el otro me salta error



De hecho a usted le pasa lo mismo en el video, desconozco el motivo por el cual no lo genera, una pena, me gustaría saber como queda lo que he comentado en el código.

Me sucede lo mismo con mi test de la unidad 3 (necesario para el ejercicio 4)

Ejercicio 3

Aquí he tenido bastantes problemas, la máquina virtual consume demasiados recursos, por lo que me es imposible moverla, así que opté por usar el instalador de sonarqube en su web, así como la versión de maven.

Al ejecutarlo en VSC hay una retalla de errores, debidos a que un plugin no quiere funcionar

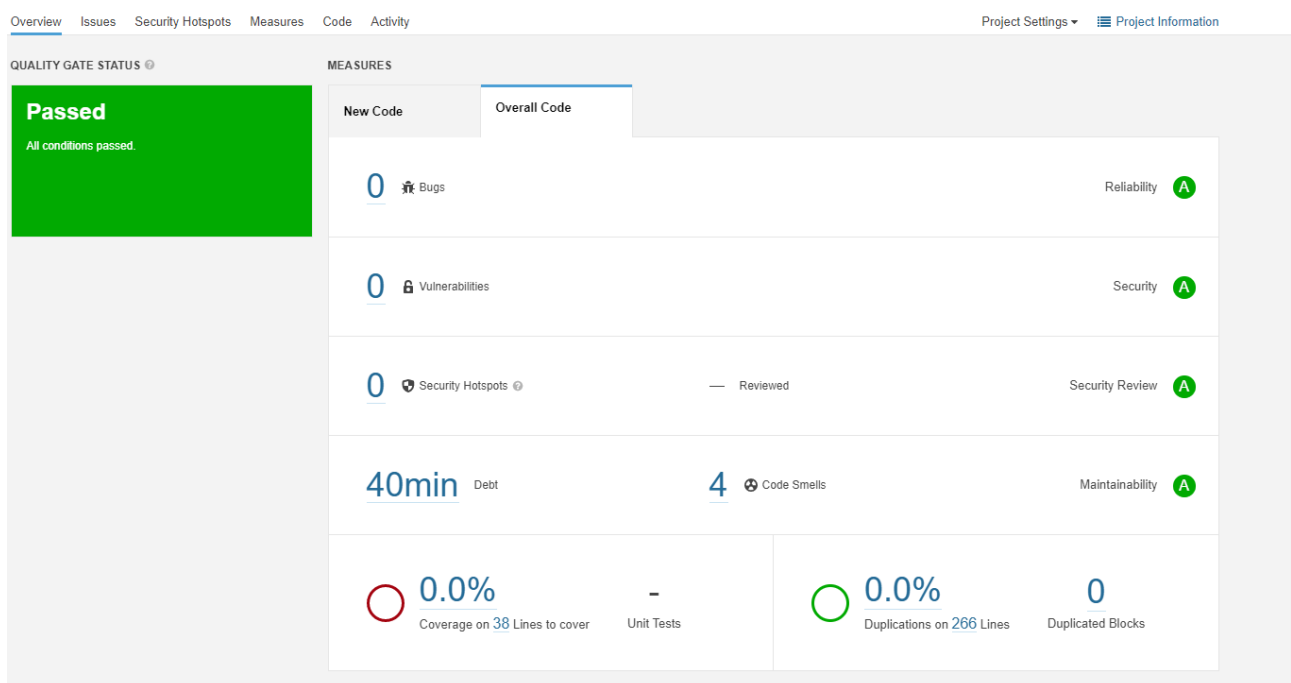
```
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 4.203 s
[INFO] Finished at: 2021-03-07T16:10:06+01:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-checkstyle-plugin:3.1.0:check (checkstyle) on project pruebaSonar: You have 185 Checkstyle violations. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://wiki.apache.org/confluence/display/MAVEN/MojFailureException
PS C:\Users\Pablo\Desktop\pruebaSonar>
```

Como se puede ver, hay 105 errores, muchos de ellos como estos:

```
[ERROR] src/main/java/pruebaSonar/Circulito.java:[1,9] (naming) PackageName: El nombre 'pruebaSonar' debe coincidir con el patrón '^([a-z]+(\\.[a-z][a-z0-9]*)*)$'.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[3] (javadoc) JavadocStyle: La primera frase debería finalizar con un punto.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[16] (javadoc) JavadocStyle: La primera frase debería finalizar con un punto.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[17] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 124).
[ERROR] src/main/java/pruebaSonar/Circulito.java:[22] (whitespace) EmptyLineSeparator: 'CTOR_DEF' debe ser separado de la declaración anterior.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[28] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[29] (javadoc) JavadocStyle: La primera frase debería finalizar con un punto.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[29] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[29,5] (whitespace) EmptyLineSeparator: '/*' cuenta con más de 1 líneas vacías antes.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[34,20] (whitespace) WhitespaceAround: '=' no está precedido de espacio en blanco.
[ERROR] src/main/java/pruebaSonar/Circulito.java:[34,20] (whitespace) WhitespaceAround: '=' no está seguido de espacio en blanco.
```

Acto seguido desactive el chequeo tal y como decían en stackoverflow y aún así, tampoco pasó, tuve que ir desde la consola de CMD, y allí si que funcionó el mvn clean package (algún problema había con VSC), tras lo cual pude poner el comando de sonarqube, pero claro al quitar el chequeo, pocos errores salen, ya que la inmensa mayoría son errores de estilo, espacios que debían quitarse, líneas en blanco, signos de “+=” que quedan mejor separados etc.

Aún así, este es el aspecto de sonarqube



src/main/java/pruebaSonar/Circulito.java

☐ Rename this package name to match the regular expression `^[a-z_]+(\\.[a-z][a-z0-9_]*)*$`. Why is this an issue? 5 minutes ago L1 convention

src/main/java/pruebaSonar/Main.java

☐ Rename this package name to match the regular expression `^[a-z_]+(\\.[a-z][a-z0-9_]*)*$`. Why is this an issue? 5 minutes ago L1 convention

☐ Replace this use of System.out or System.err by a logger. Why is this an issue? 5 minutes ago L26 bad-practice, cert, owasp-a3

☐ Replace this use of System.out or System.err by a logger. Why is this an issue? 5 minutes ago L28 bad-practice, cert, owasp-a3

4 of 4 shown

Los errores son en la nomenclatura del paquete (que usé mayúsculas igual que en las clases) y luego la recomendación de cambiar los `system.out.println` por `log.err` para que cumpla el compliance.

En cuanto a la lista de errores mostrada por mvn clean package, la inmensa mayoría son como puse, de estilo, poner puntos al final de línea, estructurar el código y formatearlo básicamente

Tras formatear, me he entretenido a “arreglar” algunos fallos, como el nombre del paquete, los puntos, etc. Curiosamente el formateo de VSC mete * en los comentarios javadoc que maven detecta como “espacios en blanco”

Ejercicio 4

El problema con javadoc, ha sido el mismo, puedo añadir comentarios, formatear el código etc pero a la hora de generar la web me da el mismo error que en el ejercicio2

En cuanto a sonarqube, ¾ de lo mismo, mvn clean package anuncia decenas de fallos

```
[ERROR] src/main/java/pruebatest/TestCases.java:[1] (misc) NewLineAtEndOfFile: El fichero no termina con un retorno de carro.
[ERROR] src/main/java/pruebatest/TestCases.java:[1] (coding) PackageDeclaration: Falta la declaración de paquete.
[ERROR] src/main/java/pruebatest/TestCases.java:[1] (imports) AvoidStarImport: Usar la importación con '*' debería evitarse - org.openqa.selenium.*.
[ERROR] src/main/java/pruebatest/TestCases.java:[7] (imports) ImportOrder: Orden incorrecto para el import 'org.junit.After'.
[ERROR] src/main/java/pruebatest/TestCases.java:[11] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[19] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[25] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 133).
[ERROR] src/main/java/pruebatest/TestCases.java:[28] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 193).
[ERROR] src/main/java/pruebatest/TestCases.java:[28,56] (coding) MagicNumber: '1366' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[28,62] (coding) MagicNumber: '900' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[31] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[34] (whitespace) EmptyLineSeparator: 'METHOD_DEF' debe ser separado de la declaración anterior.
[ERROR] src/main/java/pruebatest/TestCases.java:[36] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[39] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[39,77] (whitespace) FileTabCharacter: Archivo contiene caracteres de tabulación (este es el primer ejemplo
[ERROR] src/main/java/pruebatest/TestCases.java:[49] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 148).
[ERROR] src/main/java/pruebatest/TestCases.java:[56] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[62] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[65] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 200).
[ERROR] src/main/java/pruebatest/TestCases.java:[65] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[68,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[75] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[81] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[83] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 192).
[ERROR] src/main/java/pruebatest/TestCases.java:[83] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[84,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[90] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[96] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 141).
[ERROR] src/main/java/pruebatest/TestCases.java:[96] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[101,22] (coding) MagicNumber: '5000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[105,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[108] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[119,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[120] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 144).
[ERROR] src/main/java/pruebatest/TestCases.java:[122,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[123] (sizes) LineLength: La línea es mayor de 120 caracteres (encontrado 149).
[ERROR] src/main/java/pruebatest/TestCases.java:[125] (regexp) RegexpSingleline: Line has trailing spaces.
[ERROR] src/main/java/pruebatest/TestCases.java:[126,22] (coding) MagicNumber: '2000' es un número mágico.
[ERROR] src/main/java/pruebatest/TestCases.java:[130] (javadoc) JavadocStyle: La primera frase debería finalizar con un punto.
[ERROR] src/main/java/pruebatest/TestCases.java:[130] (regexp) RegexpSingleline: Line has trailing spaces.
[INFO] -----
```

muchos de ellos son “trailing spaces” que los añade el propio VSC al formatear, es decir en una línea javadoc tal que

```
/*
 * comentario.
 * @return valor
 */
```

El formateador separa con un * ambas líneas, cosa que a maven no le gusta (ni a sonarqube según tengo leído)

También informa de que usar el asterisco a la hora de importar no es buena idea, cosa que es cierta, y luego algunos errores que no se que significan, sobre unos números mágicos.

No he desactivado el chequeo, para poder usar sonarcube, porque el reporte que daría sería muy pobre, ya que los errores que hay son todos de estilo, si lo deshabilito sonarqube dirá que pasa con 0 errores, cosa que no es cierta.

Subiré este PDF y 2 carpetas, una llamada círculo, que contiene el proyecto original con los comentarios javadoc y los cambios del ejercicio 1, otra llamada preubasonar que contiene círculo.java y main.java con algunos errores del maven corregidos (se que esto no era necesario)

Y además el archivo testcases.java que es el archivo con los tests que hice en la unidad anterior junto con este PDF

Así mismo pondré el pull request en github

Espero que esté todo mas o menos bien.

Saludos.