Rúbrica o CheckList a tener en cuenta

1. Conocimientos básicos de Java

- No poner todos los atributos privados.
- No usar interfaces en declaraciones (List <StudentExam> l = new ArrayList <> ())
- El código no compila.
- El código tiene avisos, o malos avisos silenciosos (p. e. importar paquetes que ya no se usan).
- Comparar cadenas por igualdad de estado con ==

2. Convenciones de código Java

- Malos nombres para clases o métodos.
- Nombres de las clases en plural.
- Mala capitalización de nombres (p. e. saveclients en vez de saveClients)
- Formato de código incorrecto (alineación vertical, posición de corchetes, etc.)
- Líneas de código demasiado grandes.

3. Polimorfismo

- Mal uso de herencia y clases abstractas.
- No se utiliza para la generalización.
- Uso incorrecto de instanceof (no usarlo con subclases)
- Mala redefinición de métodos

4. Diseño

- Clases con métodos no relacionados.
- El diseño general impide que el programa consiga buenos resultados.
- Hay métodos muy largos (con muchas líneas) (falta de cohesión)
- Existen métodos con muchos parámetros (falta de cohesión).
- Hay clases con muchos métodos y / o atributos (falta de cohesión)
- Hay setters públicos en las clases.
- Los getters para listas devuelven la lista privada sin copiar (devolver nueva ArrayList ...)

5. Pruebas

- No se incluyen suficientes casos de uso (o escenarios) para los métodos a probar.
- Las pruebas no están en una carpeta de pruebas /test folder
- Las pruebas no están bien organizadas (un método para cada caso de uso)

- Métodos de prueba sin un comentario explicativo.
- No hay llamadas al método assertXxxx (...) o no están al final del método de prueba

6. Excepciones

- No hay manejadores por defecto.
- No hay un manejador por defecto para errores de sistema / programación
- No hay un manejador por defecto para errores de aplicación / lógica / usuario
- Las clases de excepción creadas no redefinen constructores de la superclase.
- Los errores de sistema y de programación no se manejan con excepciones no comprobadas.
- Hay excepciones lanzadas y recogidas en el mismo método.
- No se validan los parámetros de los métodos (incluidos constructores) y no se lanza la excepción apropiada (precondiciones: IllegalArgumentException, IllegalStateException)
- Las clases que interactúan con el usuario no comprueban la validez de los datos proporcionados por el usuario.

7. IO / Streams

- try-finally no se usa para cerrar flujos
- La cadena de procesamiento no está correctamente ensamblada.
- Las IOExceptions no se convierten en excepciones no comprobadas
- La clase de utilidad para leer/escribir ficheros no propaga la excepción FileNotFoundException

8. Ordenación

- No hay implementación Comparator/Comparable
- Los comparadores no implementan la interfaz de Java Comparador <T>
- Las clases comparables no implementan la interfaz de Java Comparable <T>
- El Comparator/Comparable no ordena con criterios secundarios (o terciarios) si se indica en el enunciado

9. Parsing

- No se detectan todos los errores durante el parseado
- Las líneas incorrectas no se manejan como se esperaba (ignoradas, etc.)
- Las líneas en blanco no son ignoradas

10. Ejecución

- La ejecución no produce los resultados esperados.
- La ejecución rompe mostrando una excepción y la traza de la pila en la consola.
- La ejecución no admite entradas incorrectas (datos proporcionados por el usuario, nombre de fichero, contenido del fichero, etc.)

11. UML

- No están representadas todas las asociaciones en el diagrama.
- Usar asociación (linea continua) en lugar de dependencia (línea discontínua) (o viceversa)
- Asociaciones con multiplicidad errónea.
- Asociaciones con navegabilidad incorrecta (cabezas de flecha)
- Usar el símbolo equivocado para representar la herencia.
- El diagrama de clases no se corresponde con el código (las clases representadas en el UML no existen en el código)

12. Bugs

- Hay errores graves en la implementación.
- Hay pequeños errores en la implementación.