**Java Code Inspection Checklist**

**1. Variable and Constant Declaration Defects (VC)**

**1. Are descriptive variable and constant names used in accord with naming conventions?**

Afirmativo, todos los nombres de constants y variables cumplen con los estándares acordados.

Ejemplo: líneas de la 11 a la 19, todas correctamente nombradas.

**2. Are there variables with confusingly similar names?**

No, todas las variables tienen un nombre fácilmente identificable y sin opción a equívocos.

Ejemplo: en la línea 108 se declara una variable NewDistance cerca de la variable distance (línea 112), que ambas se refieren a distancias, pero está claro cuál es la nueva y cual no.

**3. Is every variable properly initialized?**

Si, por ejemplo, de la línea 52 a la 54 donde hay 3 variables bien inicializadas.

**4. Could any non-local variables be made local?**

No, todas las variables no locales aparecen en más de un método.

Por ejemplo: la variable dijkstraExec aparece en la línea 134 y en la 154, líneas que pertenecen a dos métodos diferentes.

**5. Are there literal constants that should be named constants?**

No, no hay ningún caso de constante literal que se repita tanto como para que sea necesario crear una constante. Ejemplo: línea 112, se utiliza la constante literal 0.0 pero no aparece en ningún otro momento.

**6. Are there macros that should be constants?**

No, no existe ningún macro.

**7. Are there variables that should be constants?**

No, todas las variables son variables bien creadas, ninguna tiene que ser constante.

Por ejemplo: nVertices (línea 16) obtiene su valor en los parámetros del constructor (línea 41)

**2. Function Definition Defects (FD)**

**8. Are descriptive function names used in accord with naming conventions?**

Si, todas tienen un nombre acorde a los estándares.

Por ejemplo: en la línea 73 la función nextCur se adapta a los estándares.

**9. Is every function parameter value checked before being used?**

No. Por ejemplo: en la línea 107 en parámetro ini de la función es utilizado en la línea 111 sin haberlo chekeado antes.

**10. For every function: Does it return the correct value at every function return point?**

Si, por ejemplo, en la línea 179 es una función que devuelve un booleano como bien hace en la línea 180.

**3. Class Definition Defects (CD)**

**11. Does each class have an appropriate constructor and destructor?**

Sí, aunque java no hace uso de destructores, sí podemos encontrar un constructor para la clase Dijkstra que comienza en la línea 41.

**12. For each member of every class: Could access to the member be further restricted?**

No, ya que todas las declaraciones de variables de la clase (líneas 13 a 19) hacen el acceso restringido.

**13. Do any derived classes have common members that should be in the base class?**

En este código no hay clases derivadas.

**14. Can the class inheritance hierarchy be simplified?**

No, más simple no podría ser porque solo hay una clase.

**4. Computation/Numeric Defects (CN)**

**15. Is overflow or underflow possible during a computation?**

No contemplamos que se pueda producir overflow ni underflow.

**16. For each expressions with more than one operator: Are the assumptions about order of evalu-ation and precedence correct?**

Sí, en la línea 118 se puede ver un claro ejemplo.

**17. Are parentheses used to avoid ambiguity?**

No se utilizan paréntesis en ninguna expresión para evitar la ambigüedad, pero también es cierto que no son necesarios, como en la línea 78.

**5. Comparison/Relational Defects (CR)**

**18. Are the comparison operators correct?**

Sí, todos los comparadores están usados correctamente. Un ejemplo del uso de estos se puede ver en la línea 118.

**19. Is each boolean expression correct?**

Sí, por ejemplo en la línea 78 se utiliza el boolean visited[i] inicializado a false con anterioridad de forma correcta.

**20. Are there improper and unnoticed side-effects of a comparison?**

No

**6. Control Flow Defects (CF)**

**21. For each loop: Is the best choice of looping constructs used?**

Si, por ejemplo, en la línea 57 el bucle for usado está correctamente elegido ya que inicializa un array pasando así por todos los valores.

**22. Will all loops terminate?**

Si. Por ejemplo: en la línea 113 funciona correctamente porque va cambiando el valor de false a true en la línea 130, hasta que visited[end]= true y sale del bucle.

**23. When there are multiple exits from a loop, is each exit necessary and handled properly?**

No existe ninguna salida multiple en ningún bucle.

**24. Does each switch statement have a default case?**

No existe ningún switch.

**25. Are missing switch case break statements correct and marked with a comment?**

No existe ningún switch.

**26. Is the nesting of loops and branches too deep, and is it correct?**

No, el nivel máximo de anidamiento en es la línea 113 con un while que anida con el for de la línea 115, que además es correcto ya que no sería simplificable.

**27. Can any nested if statements be converted into a switch statement?**

No, en ningún caso sería adecuado. Por ejemplo: en la línea 154 solo hay un if por lo que no sería necesario crear un switch.

**28. Are null bodied control structures correct and marked with braces or comments?**

No existe ninguna estrcutura de control con un cuerpo null en el código.

**29. Does every function terminate?**

Si, por ejemplo, la función de la línea 73, todos sus bucles acaban y además devuelve el valor del tipo correcto.

**30. Are goto statements avoided?**

No existe ninguna sentencia goto.

**7. Input-Output Defects (IO)**

**31. Have all files been opened before use?**

No se utiliza ningún archivo para guardar o leer datos.

**32. Are the attributes of the open statement consistent with the use of the file?**

No se utiliza ningún archivo para guardar o leer datos.

**33. Have all files been closed after use?**

No se utiliza ningún archivo para guardar o leer datos.

**34. Is buffered data flushed?**

No está implícito en el código.

**35. Are there spelling or grammatical errors in any text printed or displayed?**

No se imprime ningún texto por pantalla.

**36. Are error conditions checked?**

No, por ejemplo, en la línea 78 no verifica el valor de las variables usadas.

**8. Module Interface Defects (MI)**

**37. Are the number, order, types, and values of parameters in every function call in agreement with the called function’s declaration?**

Si, los parámetros en cada llamada de cada función están conforme a los parámetros que aparecen en la declaración de la función llamada. Por ejemplo: línea 131 aparece una llamada a la función nextCur con los parámetros adecuados, en este caso ninguno.

**38. Do the values in units agree (e.g., inches versus yards)?**

Si. Por ejemplo, la unidades de distancia son siempre double, es decir, si fuesen metros todos serian metros y si fuesen kilómetros, todos serían kilómetros.

**9. Comment Defects (CM)**

**39. Does every function, class, and file have an appropriate header comment?**

Si. Por ejemplo, de la línea 21 a la 40 es un comentario describiendo el constructor.

**40. Does every variable or constant declaration have a comment?**

No, por ejemplo, la variable next de la línea 74 no está contada.

**41. Is the underlying behavior of each function and class expressed in plain language?**

Si, todos los comentarios explican de forma clara el comportamiento subyacente de las funciones y la clase.

Por ejemplo: de la línea 3 a la 9 explica la utilización de la clase implementada.

**42. Is the header comment for each function and class consistent with the behavior of the functionor class?**

Si, todas las descripciones de funciones corresponden con lo que hace el código. Por ejemplo: de la línea 47 a la 48 dice que lo hace la función de inicializar los datos y la función hace lo que dice el comentario.

**43. Do the comments and code agree?**

Sí, no existe ninguna inconsistencia entre los comentarios y el código que le sigue. Por ejemplo: línea 51 dice que crea las estructuras de datos y en las líneas 52,53 y 54 lo hace.

**44. Do the comments help in understanding the code?**

Si, se entiende perfectamente. Por ejemplo: línea 111, existe un comentario explicativo en la misma línea sobre la utilidad de esa línea de código.

**45. Are there enough comments in the code**

Si, está explicado prácticamente línea por línea, solo las líneas más básicas no tienen explicación debido a su obviedad, por lo que existen comentarios suficientes.

**46. Are there too many comments in the code?**

No, porque los comentarios no son suficientes como para ensuciar el código y ayudan a una comprensión total del código desarrollado.

**10. Packaging Defects (LP)**

**47. For each file: Does it contain only one class?**

Si, solo hay un fichero java con una única clase llamada Dijkstra.

**48. For each function: Is it no more than about 60 lines long?**

No, hay ninguna función con más de 60 líneas, la más extensa es computerShortestPath que va de la línea 107 a la 136 (29 líneas).

**49. For each class: Is no more than 2000 lines long (Sun Coding Standard) ?**

**Praktikum Software Engineering 99: Code Inspection Checklist**

Solo hay una clase y tiene 182 líneas, por lo que no supera las 2000.

**11. Modularity Defects (MO)**

**50. Is there a low level of coupling between packages (classes)?**

No hay acoplamiento, ya que solo hay una clase.

**51. Is there a high level of cohesion within each package?**

Si, ya que solo hay una clase donde todos los métodos tienen una temática común. Por ejemplo, dentro de la función computeShortestPath se llama a la función initializeDataStructures.

**52. Is there duplicate code that could be replaced by a call to a function that provides the behaviorof the duplicate code?**

No, no existe código duplicado.

**53. Are framework classes used where and when appropriate?**

No, no hay uso de frameworks.

**12. Performance Defects (PE) [Optional]**

**54. Can better data structures or more efficient algorithms be used?**

**55. Are logical tests arranged such that the often successful and inexpensive tests precede the**

**more expensive and less frequently successful tests?**

**56. Can the cost of recomputing a value be reduced by computing it once and storing the results?**

**57. Is every result that is computed and stored actually used?**

**58. Can a computation be moved outside a loop?**

**59. Are there tests within a loop that do not need to be done?**

**60. Can a short loop be unrolled?**

**61. Are there two loops operating on the same data that can be combined into one?**