**Java Code Inspection Checklist**

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

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

Clase Agenda: si, por ejemplo, la variable numEntries (línea 24)

Agenda Interface: no hay variables.

Clase Entry: si, por ejemplo, la variable surname (línea 28)

Clase Parser: si, por ejemplo, la variable PRO (línea 22)

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

Clase Agenda: no, cada una tiene un nombre claramente autodescriptivo, por ejemplo, en la línea 13 la variable info.

Agenda Interface: NO existen variables

Clase Entry: no, cada una tiene un nombre claramente autodescriptivo, por ejemplo, en la línea 16 la variable name.

Clase Parser: no, cada una tiene un nombre claramente autodescriptivo, por ejemplo, en la línea 60 la variable telefono.

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

Clase Agenda: si, por ejemplo, numEntries= 0 en la línea 28

Agenda Interface: no hay variables

Clase Entry: si, por ejemplo, country= “ ” en la línea 31

Clase Parser: si, por ejemplo, line = null en la línea 31

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

Clase Agenda: no, tanto first como nunEntries aparecen en varios métodos

Agenda Interface: no hay variables

Clase Entry: no, ya que solo hace sets y gets con todas las variables

Clase Parser: no, todas las variables no locales son correctas

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

Clase Agenda: no, no hay ninguna constante literal

Agenda Interface: no hay constantes literales

Clase Entry: no, no hay constantes literales

Clase Parser: no, un ejemplo correcto serían los -1 de la línea 76

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

Clase Agenda: no, no existen macros

Agenda Interface: no, no existen macros

Clase Entry: no, no existen macros

Clase Parser: no, no existen macros

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

Clase Agenda: no, no hay ninguna que no deba modificar su valor, por ejemplo: línea 82 la variable numEntries

Agenda Interface: no hay variables

Clase Entry: no, no hay ninguna que no deba modificar su valor, por ejemplo: línea 59 la variable name

Clase Parser: no, no hay ninguna que no deba modificar su valor, por ejemplo: línea 81 la variable provincia

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

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

Clase Agenda: si, por ejemplo, línea 39 el método addEntry

Agenda Interface: si, por ejemplo, línea 11 el método isEmpty

Clase Entry: si, por ejemplo, línea 131 el método getZip

Clase Parser: si, por ejemplo, línea 53 el método a createEntry

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

Clase Agenda: no, no checkea ninguno

Agenda Interface: no, no checkea ninguno

Clase Entry: no, no checkea ninguno

Clase Parser: no, no checkea ninguno

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

Clase Agenda: si, por ejemplo, la función boolean loadAgenda devuelve un booleano en la línea 153

Agenda Interface: no hay ningún return

Clase Entry: si, por ejemplo, la función voidTelephone no devuelve nada en la línea 87

Clase Parser: si, por ejemplo, la función getLine devuelve un string en la línea 49

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

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

Clase Agenda: sí, aunque no hace uso de destructores se puede ver el constructor de la clase Agenda en la línea 26 y el constructor de la clase AgendaNode en la 16.

AgendaInterface: no tiene constructor ni destructor, al ser una interfaz que ha de ser implementada desde la clase Agenda.

Clase Entry: en la línea 25 podemos encontrar el constructor de la clase Entry.

Clase Parser: sí, en la línea 30 encontramos el constructor de la clase Parser.

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

Clase Agenda: no, ya que las declaraciones de las variables de la clase son private (líneas 23 y 24).

AgendaInterface: al tratarse de la interfaz no posee miembros propios.

Clase Entry: no, todas las variables están declaradas como private en la líneas 16 a 23.

Clase Parser: no, en las líneas 27 y 28 se pueden ver las que las declaraciones de las variables son private.

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

No hay clases derivadas para ninguna de las clases.

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

No ya que no hay herencia de clases.

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

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

No encontramos la posibilidad de que se produzca overflor o underflow.

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

Clase Agenda: sí, en la línea 45 podemos ver un ejemplo de ello.

AgendaInterface: esta clase no tiene expresiones que evaluar.

Clase Entry: sí, como por ejemplo en la línea 45.

Clase Parser: sí, en la línea 76 encontramos un claro ejemplo.

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

Clase Agenda: no, aunque no son necesarios, como por ejemplo en la línea 74.

AgendaInterface: esta clase no tiene expresiones en las que usar paréntesis.

Clase Entry: no, en la única expresión del código (línea 45) no se usan (no necesarios).

Clase Parser: sí, en la línea 76 se puede observar su uso de manera correcta.

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

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

Clase Agenda: sí, como por ejemplo en la línea 46.

AgendaInterface: al ser una interfaz no posee operadores que evaluar.

Clase Entry: no se usan operadores de comparación.

Clase Parser: sí, se puede ver el ejemplo en la línea 76.

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

Clase Agenda: sí, por ejemplo en la línea 46 se utiliza el boolean found que se inicializa antes a false.

AgendaInterface: la interfaz no tiene booleans.

Clase Entry: no tiene expresiones booleanas.

Clase Parser: sí, como en la línea 88 que utiliza el boolean auxb que ha sido inicializado con anterioridad.

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

No, en las comparaciones encontradas en las clases no hemos podido encontrar ninguna en la que se puedan producir efectos secundarios.

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

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

Clase Agenda: si, por ejemplo, el while de la línea 133

Agenda Interface: no hay bucles

Clase Entry: no hay bucles

Clase Parser: no hay bucles

**22. Will all loops terminate?**

Clase Agenda: no, en el bucle while de la línea 133 no termina, ya que no pasa al siguiente de cur.

Agenda Interface: no hay bucles

Clase Entry: no hay bucles

Clase Parser: no hay bucles

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

Clase Agenda: si, por ejemplo, el bucle while de la línea 74, tiene dos salida y controla ambas de forma correcta

Agenda Interface: no hay bucles

Clase Entry: no hay bucles

Clase Parser: no hay bucles

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

Clase Agenda: no hay ninguna sentencia switch

Agenda Interface: no hay ninguna sentencia switch

Clase Entry: no hay ninguna sentencia switch

Clase Parser: no hay ninguna sentencia switch

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

Clase Agenda: no hay ninguna sentencia switch

Agenda Interface: no hay ninguna sentencia switch

Clase Entry: no hay ninguna sentencia switch

Clase Parser: no hay ninguna sentencia switch

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

Clase Agenda: no hay ningún anidamiento

Agenda Interface: no hay bucles

Clase Entry: no hay bucles

Clase Parser: no hay bucles

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

Clase Agenda: no hay ningún anidamiento

Agenda Interface: no hay bucles

Clase Entry: no hay bucles

Clase Parser: no hay bucles

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

Clase Agenda: no hay ninguna estrcutura de control

Agenda Interface: no hay ninguna estrcutura de control nula

Clase Entry: no hay ninguna estrcutura de control

Clase Parser: no hay ninguna estrcutura de control

**29. Does every function terminate?**

Clase Agenda: si, por ejemplo, la función nEntries de la línea 105

Agenda Interface: si, ya que solo son las declaraciones que luego se implementan en clase agenda

Clase Entry: si, por ejemplo, setAdress de la línea 67

Clase Parser: si, por ejemplo, createEntry de la línea 53

**30. Are goto statements avoided?**

Clase Agenda: no existe ninguna sentencia goto

Agenda Interface: no existe ninguna sentencia goto

Clase Entry: no existe ninguna sentencia goto

Clase Parser: no existe ninguna sentencia goto

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

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

Clase Agenda: si, por ejemplo, en la línea 129

Agenda Interface: no hay ficheros

Clase Entry: no hay ficheros

Clase Parser: no hay ficheros

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

Clase Agenda: si, por ejemplo, el uso del fichero en la función saveAgenda de la línea 123

Agenda Interface: no hay ficheros

Clase Entry: no hay ficheros

Clase Parser: no hay ficheros

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

Clase Agenda: no, en la línea 129 se abre un ficheor que nunca llega a cerrar

Agenda Interface: no hay ficheros

Clase Entry: no hay ficheros

Clase Parser: no hay ficheros

**34. Is buffered data flushed?**

Clase Agenda: no limpia en buffer en ningún momento

Agenda Interface: no hay ficheros

Clase Entry: no hay ficheros

Clase Parser: no hay ficheros

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

Clase Agenda: no hay nada escrito para imprimir solo le llegan strings ya formadas

Agenda Interface: no hay nada para imprimir

Clase Entry: no hay errores como en la asignación de la línea 112

Clase Parser: no hay errores como en la asignación de la línea 113

**36. Are error conditions checked?**

Clase Agenda: si, en la línea 123 se invoca una excepción en caso de error

Agenda Interface: no hay opción a que haya un error

Clase Entry: no hay ningún control de error

Clase Parser: no hay ningún control de error

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

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

Clase Agenda: si, por ejemplo, el getName() de la línea 74

Agenda Interface: no hay llamadas a métodos

Clase Entry: si, por ejemplo, la función equals de la línea 45

Clase Parser: si, por ejemplo, la función createEntry de la línea 42

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

Clase Agenda: no existen valores que puedan tener distintas unidades

Agenda Interface: no hay ningún valor

Clase Entry: no existen valores que puedan tener distintas unidades

Clase Parser: no existen valores que puedan tener distintas unidades

**9. Comment Defects (CM)**

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

Clase Agenda: No, esta clase no tiene ningún tipo de comentario además de que algunas de sus funciones como nEntries() o isEmpty(), también carecen de comentario explicativo. Por otro lado, todos los comentarios que aparecen, son apropiados a las funcionalidades.

Clase Entry: Tanto la clase como todas sus funciones tienen sus respectivos comentarios explicativos.

Clase Parser: La clase Parser tiene comentada su funcionalidad, pero sus funciones, ningún tiene un comentario explicativo.

AgendaInterface: El archivo contenedor de la interfaz tiene un comentario explicativo, sus funciones no están comentadas, pero se explica su funcionalidad en la clase Agenda, pero esto no pasa con las funciones nEntries() y isEmpty().

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

Clase Agenda: Ninguna variable tiene comentario

Clase Entry: Ninguna variable tiene comentario

Clase Parser: Ninguna variable tiene comentario

AgendaInterface: No existen ni constantes ni variables

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

Si, por ejemplo, todos los nombres de las funciones son descriptivos y adecuados a la función que representa. Lo mismo ocurre con las variables.

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

Todas las funciones y clases comentadas tienen un comentario que corresponde adecuadamente a la definición y a su funcionalidad.

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

Sí, no encontramos ninguna inconsistencia entre los comentarios y el código, como por ejemplo en las líneas 38 a 42 de la clase Entry se explica el método hasData() y a continuación se desarrolla el código de manera correcta.

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

La ayuda de los comentarios es muy útil para entender el código de cada clase y sus métodos.

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

Sí, hay suficientes comentarios como para poder entender bien el código.

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

No, en todas las clases podemos encontrar comentarios, pero los consideramos necesarios para el correcto entendimiento del código.

**10. Packaging Defects (LP)**

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

No, en el archivo Agenda.java, hay contenidas dos clases la Clase AgendaNode (línea 12) y en la clase Agenda (linea 22)

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

No hay ninguna función con más de 60 líneas. La función más larga es de 23 líneas, que corresponde a loadAgenda(), de la clase Agenda

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

No hay ninguna clase que tenga más de 2000 lineas. La clase más larga de nuestro programa Agenda tiene 129 líneas (clase Entry).

**11. Modularity Defects (MO)**

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

Existe un bajo acoplamiento entre clases. En este caso solo hay una dependencia de datos, entre la clase AgendaNode y la clase Agenda, que implementa un objeto de la clase anterior. La clase Agenda no funcionaría sin el objeto AgendaNode, pero AgendaNode funcionaría tanto si hay cambios en Agenda como si no se viera declarada en esa clase. Lineas de la 12 a la 23 de la Clase Agenda.

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

Existe una alta cohesión, pues cada módulo (clase) tiene una funcionalidad clara y definida y su alcance abarca la implementación de la clase.

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

No, no existe en ninguna clase, ninguna línea o función de código duplicado o que haga funciones similares.

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

No existen clases framework en este caso.

**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?**