Praktikum Software Engineering 99: Code Inspection Checklist

**Java Code Inspection Checklist**

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

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

**No, linea 94, Entry p = null.**

2. Are there variables with confusingly similar names?

**Sí, linea 94 -> Entry p = null; linea 127 -> Parser p = new Parser();**

3. Is every variable properly initialized?

**No, linea 13 -> Entry info; linea 14 -> AgendaNode sig;**

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

**No, linea 23 -> private AgendaNode first;**

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

**No hay constantes**

6. Are there macros that should be constants?

**No hay macros**

7. Are there variables that should be constants?

**No**

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

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

**Si, linea 68 -> public boolean removeEntry(String name) {**

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

**No, linea 46 -> if (cur.info.getName() == p.getName()) {**

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

Si, linea 58 -> return false;

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

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

No, línea -> 22 public class Agenda implements AgendaInterface {, no tiene destructor

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

Si, linea 13 -> Entry info;

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

No hay clases derivadas

14. Can the class inheritance hierarchy be simplified?

No hay herencia

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

15. Is overflow or underflow possible during a computation?

No, linea 43 -> if (first != null) {

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

No hay ninguna expresión de asignación con más de 1 operador, y no hay expresiones aritméticas

17. Are parentheses used to avoid ambiguity?

No, linea 45 -> while (cur !=null && !found) {

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

18. Are the comparison operators correct?

Sí, linea 45 -> while (cur !=null && !found) {

19. Is each boolean expression correct?

Si, linea 41 -> boolean found = false;

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, linea 45 -> while (cur != null && !found) {

22. Will all loops terminate?

No, while (cur != null) {

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

No hay multiples salidas.

24. Does each switch statement have a default case?

No hay switch

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

No hay switch

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

No hay bucles anidados

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

No hay if anidados

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

No hay

29. Does every function terminate?

Si, linea 102 -> return p;

30. Are goto statements avoided?

No existen los goto

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

31. Have all files been opened before use?

Si, linea 129 -> FileWriter fichero = new FileWriter("agendofile.txt");

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

Si, linea 129 -> FileWriter fichero = new FileWriter("agendofile.txt");

33. Have all files been closed after use?

No, linea 142 -> **return** success;

no cierra el fichero antes

34. Is buffered data flushed?

No, no se hace

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

No, no existen errores

36. Are error conditions checked?

No, no se realiza comprobaciones en la funcion saveAgenda

**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?

Lo complen todas, linea 68 -> public boolean removeEntry(String name) {

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

Si, linea 28 -> numEntries = 0;

**9. Comment Defects (CM)**

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

La clase agenda no tiene comentario linea 21, pero sus funciones si

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

No, linea 32 -> private AgendaNode first;

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

Si, linea 123 -> public boolean saveAgenda() throws IOException {

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

or class?

La clase agenda no tiene comentario linea 21, pero sus funciones si

43. Do the comments and code agree?

Si, linea 145

44. Do the comments help in understanding the code?

Si, linea 145

45. Are there enough comments in the code

Si, todas las funciones están explicadas

46. Are there too many comments in the code?

No, existen las justas.

**10. Packaging Defects (LP)**

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

No, la clase agenda tiene 2 clases, Agenda y AgendaNode

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

No, estan todas dentro del limite

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

Praktikum Software Engineering 99: Code Inspection Checklist

Si, ninguna supera las 2000 lineas

**11. Modularity Defects (MO)**

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

No, existe una interfaz de agenda

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

Si, todas las funciones tienen cohesión con 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 codigo duplicado

53. Are framework classes used where and when appropriate?

No existen clases framework

**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?

slightly adapted from the C++ Inpsection Checklist of Christopher Fox, http://www.cs.jmu.edu/users/foxcj/cs555/StdDoc/CppChk.htm.

Copyright 1998 Christopher Fox