

Checklist de Inspeção Java

1. Defeitos de declaração de Variáveis, Atributos e Constantes (VC)

- Os nomes dos atributos, variáveis e constantes seguem a convenção estabelecida?
- Existem nomes confusos ou muitos similares entre si?
- Todos os atributos e variáveis têm o tipo correto?
- Todos os atributos e variáveis são inicializados apropriadamente?
- Tem alguma variável não-local que pode ser transformada em local?
- Todas as variáveis de laços “for” são declaradas no cabeçalho do laço?
- Existem literais que deveriam ser constantes?
- Existem variáveis ou atributos que deveriam ser constantes?
- Existem atributos que deveriam ser variáveis locais?
- Todos os atributos têm modificadores de acesso (private, protected, public) apropriados?
- Existem atributos de classe que deveriam ser de objetos e vice-versa?

2. Defeitos de Definição de Métodos (FD)

- Os nomes dos métodos seguem a convenção estabelecida?
- O valor de cada parâmetro do método é verificado antes de ser usado?
- Para cada método: Ele retorna o valor correto em cada ponto de retorno do método?
- Todos os métodos têm o modificador de acesso apropriado (private, protected, public)?
- Existem métodos de classe que deveriam ser de objeto e vice-versa?

3. Defeitos de definição de Classes (CD)

- Todas as classes têm construtores e destruidores apropriados?
- As subclasses têm coisas em comum que deveriam estar na superclasse?
- É possível simplificar a hierarquia de herança de classes?

4. Defeitos de referência a Dados (DR)

- Para toda referência a um vetor: O valor de indexação está dentro dos limites definidos?
- Para toda referência a um objeto: É certeza que seu valor não é nulo?

5. Defeitos numéricos e de computação (CN)

- Existe alguma computação com tipo de dados misturados?
- É possível que ocorra *overflow* ou *underflow* durante a computação?
- Para cada expressão que tiver mais de um operador: O que é assumido sobre a ordem de avaliação e precedência de operações está correto?
- Parênteses são usados sempre que possível para evitar ambigüidades?

6. Defeitos de Comparação e Relacionamento (CR)

- Para todo teste booleano: A condição correta é checada?
- Os operadores da operação estão corretos?
- As expressões booleanas foram simplificadas levando operadores de negação para sua parte mais interna?
- Toda expressão booleana está correta?
- A comparação pode ter efeitos colaterais impróprios?
- Será que "&" e "|" for inadvertidamente trocado por a "&&&" ou "||"?

7. Defeitos de Fluxo de Controle (CF)

- Para cada laço: Os construtores de laços mais adequados para aquela situação são usados?
- Todos os laços irão terminar?
- Quando existem múltiplos pontos de saída de um laço: Todas estas saídas são necessárias e tratadas adequadamente?
- Todas as declarações “switch” têm um caso default?
- Os comandos “break” que estão faltando nos comandos “switch” são propósitos, estão corretos e comentados adequadamente?
- Os comandos de “break” com “label” enviam o controle de execução para o lugar certo?
- Existem laços ou condicionais aninhados em grande profundidade? Isto é necessário?
- Será que os “ifs” aninhados não podem ser convertidos em declarações “switch”?
- Todas as estruturas de controle deixadas vazias estão corretas e limitadas por chaves ou comentário?
- Todas as exceções são tratadas apropriadamente?
- Todos os métodos terminam?

8. Defeitos de Entrada e Saída (IO)

- Todos os arquivos são abertos antes de serem usados?
- Os atributos dos objetos de entrada são consistentes com o uso do arquivo?
- Todos os arquivos são fechados após o seu uso?
- Existem erros de escrita em qualquer texto impresso ou mostrado na tela?
- Todas as exceções de I/O são tratadas de forma adequada?

9. Defeitos de Interface de Módulos (MI)

- O número, a ordem o tipo e os valores dos parâmetros passados a um método estão de acordo com a declaração do método?
- Os valores das escalas e unidades usadas estão compatíveis (e.g., centímetros versus metros)?
- Se um objeto ou vetor é passado como parâmetro. Ele é modificado quando não deveria ser? Se ele pode ser modificado, ele é modificado corretamente pelo método chamado?

10. Defeitos de Comentário (CM)

- Todo o método, classe, e arquivo têm um cabeçalho apropriado?
- Toda declaração de atributo, variável, e constante tem um comentário?
- O comportamento de cada método e classe é explicado em linguagem simples?
- Os comentários de cabeçalho de cada método e classe são consistentes com os seus comportamentos?
- Os comentários são consistentes com o código?
- Os comentários ajudam à compreensão do código?
- Existem comentários suficientes no código?
- Existem comentários demais no código?

11. Defeitos de Layout e Formatação (LP)

- A identação e formatação do código são consistentes?
- Para cada método: Ele tem mais do que 60 linhas de comprimento?
- Para cada módulo de compilação: Ele tem mais que 600 linhas de comprimento?

12. Defeitos de Modularidade (MO)

- Existe acoplamento de baixo nível entre os módulos (métodos e classes)?
- Existe coesão de alto nível dentro do módulo (métodos e classes)?
- Existe código repetido que poderia ser substituído por uma chamada a um método que implementasse este comportamento?
- As bibliotecas de classe Java são usadas quando e onde apropriado?

13. Defeitos de Uso de Armazenamento (SU)

- Os vetores são grandes o suficiente?
- As referências a vetores e objetos são marcadas como null quando deixam de ser necessárias?

14. Defeitos de Desempenho (PE)

- Melhores estruturas de dados ou melhores algoritmos podem ser utilizados?
- O custo de re-computar um valor pode ser reduzido através da computação do valor uma vez e posterior armazenamento do resultado?
- Todo valor computado e armazenado é realmente utilizado?
- Um procedimento computacional pode ser movido para fora de seu laço de repetição?
- Existem testes em laços de repetição que não precisam ser feitos?
- Um laço de repetição curto pode ser “decomposto” (desfeito)?
- Existem dois laços de repetição operando sobre os mesmos dados? Eles podem ser combinados em um único laço?

15. Defeitos Conceituais e de Domínio (CS)

- Os componentes do simulador estão completos em termos de métodos de execução e configuração?
- Configuração OK confere todos os aspectos do componente?
- O componente está implementando código desnecessário ou que poderia ser herdado de outra classe?
- O fluxo de simulação é adequadamente informado nas impressões?
- As distribuições estão sendo geradas e utilizadas adequadamente?
- A máquina de estados e a chamada de execução dos componentes estão implementadas de forma consistente?
- O desempenho da simulação pode ser melhorado?
- Exceções de simulação são pegas e tratadas adequadamente?