Princípios do CleanCode

A muitos princípios no dito Clean Code, código limpo em portugues possui muitas convenções, boas práticas que fazem nosso código ser mais fácil de manter, ter mais durabilidade entre outros benefícios.

Regras gerais, Existem algumas regras gerais no clean code, são elas, KISS, regra do escoteiro, causa raiz, além de sempre seguir as convenções.
A regra que mais me identifiquei seria a KISS, uma sigla para manter isto estupidamente simples, por mais que tentemos, às vezes não conseguimos fazer que o nosso código seja tão simples de entendimento ou até mesmo o uso da aplicação que estamos fazendo.
Mas as outras regras acabam que complementando uma às outras, a regra do escoteiro por exemplo sempre nos fazendo deixar o nosso código o mais limpo que conseguimos, às vezes por falta de experiência, ou de realmente não saber dos métodos padrões fazemos coisas mais trabalhosas e “feias” que nem seriam realmente necessárias, por isso o estudo de padrões nos torna desenvolvedores muito mais eficazes.

Em geral devemos maneirar o uso de ifs e fazer a troca para a polimorfia, um dos conceitos mais usados na orientação a objeto, e um dos que fazem o paradigma ser tão utilizado.
Um ponto muito bom, uma das regras de lei de design, é evitar o uso de configurações desnecessárias, criar coisas que nem precisam ser usado só por costume não deve ser um bom hábito, por exemplo, criar métodos getter and setter em qualquer classe que você cria, às vezes nem precisamos usar esses métodos mas por um padrão estabelecido quando começamos a estudar queremos criar sempre.

Ser consistente é uma das regras de entendimento de codigo, seguir sempre o mesmo padrão ao criar algo, como uma classe ou método, na ideia de manter seu código similar ao que você está fazendo, se você criou uma classe de um jeito, ao criar outra não tentar criar algo totalmente diferente daquilo que você já fez, Um das coisas que dá dor de cabeça a qualquer desenvolvedor é a criação de nome de variáveis, precisamos criar um padrão de ser totalmente explicativo ao criar um nome de variável, não apenas algo como total, saldo, etc.
criar algo como totalDeProdutos ou saldoTotalRestante, nomes o mais explicativos possíveis, que por mais que seja apenas você naquele projeto, você sempre vai poder bater o olhar e saber exatamente sobre oque você está falando, até porque um método que tem uma variável

como parâmetro, ou em seu corpo bem explicativa como totalDeProdutos, estará fazendo alguma alteração no total de produtos.
Gostei bastante do tópico sobre obsessão primitiva, algo que realmente eu faço bastante, e vejo agora que é uma má prática com certeza.
Criar Classes usando tipos primitivos, é normal e qualquer desenvolvedor faz, mas com certeza podemos criar objetos que supririam nosso desejo por ficar contidos nesses tipos primitivos, até por que podemos manipular mais facilmente tipos objeto, evitando usar excessivamente Strings, que podem nos causar certa confusão as vezes, evitando também pontos, acentos, caracteres especiais, seja no nome da variável, seja no conteudo da variavel.
Evitar códigos comentados é algo que eu preciso por em prática rapidamente, sempre faço pequenos testes nos códigos, e acabo deixando resquícios desses testes por meio dos comentários e acabo deixando o código totalmente "manchado", ficava com um certo medo de precisar daquele bloco de código novamente e ter excluído, por mais que já achava uma forma muito errada de codificar.
Precisamos separar os conceitos verticalmente, mantendo uma estrutura de pastas de forma organizada, por exemplo para classes sua própria pasta, enuns se for usar, interfaces, classes utilitárias entre outras coisas.
Esconder estruturas internas sempre é algo a se fazer, atributos, métodos como getters e setters devem ser encapsulados sempre, privar essas propriedades sempre vai trazer mais pontos positivos que negativos, obviamente existem casos que precisamos não "esconder" essas propriedades mas são casos bem raros, Classes pais não devem saber dos detalhes dos seus filhos, formas de métodos ou atributos. Sempre é melhor fazer Sobrecarga do que criar uma tomada de decisão do sistema, Sobrecarga para mim é uma ótima forma de criar sistemas, obviamente não devemos criar um método que tenha parâmetros diferentes só por fazer, mas é uma forma muito boa de sanar alguns problemas.

No cenário de teste temos algo bastante diferente, utilizamos um assert, Que é uma forma de "garantir" que o teste que você fez foi assertivo. método da classe Assert que possui muita sobrecarga, e apenas devemos ter em mente um Assert, não fazendo muitos para não nos confundirmos, oque vem o ponto de termos uma complexidade que é desnecessária, nos nossos testes ou desenvolvimento, é algo que devemos tomar cuidado, Isso também pode ser levado para muitos segmentos, não apenas na codificação, mas nas formas do design do sistema, da interface com o usuário e tantas outras áreas da tecnologia.