Clean Code

martes, 14 de mayo de 2024 11:19

Princípios Fundamentais do Clean Code

1. Nomenclatura Clara e Significativa:
A escolha dos nomes para variáveis e funções deve ser feita com cuidado. Um bom nome explica claramente o propósito da variável ou função, facilitando a compreensão do código.

```
//forma "não correta"
int c = 5;
for (int i=0;  i < c; i++) {

}


//forma "correta"
int totalDeChances = 5;
for (int i=0;  i < totalDeChances ; i++) {

}
```

2. Evite Números Mágicos:

Números mágicos são valores numéricos com significado não claro no código. Substitua-os por constantes nomeadas para tornar o código mais legível. Por exemplo, substitua 86400 (número de segundos em um dia) por SEGUNDOS_POR_DIA.

3. Reduza Comentários Desnecessários:

Comentários devem ser usados para documentar o código, não para explicar o que uma variável ou função faz. Um código bem escrito deve ser autoexplicativo. Se sentir necessidade de adicionar um comentário para explicar o que uma linha de código faz, considere refatorar essa linha para torná-la mais clara.

4. Funções Curtas e Focadas:

Cada função deve ter um único propósito e executá-lo bem. Evite funções longas e complexas, dividindo-as em funções menores se necessário. Isso não apenas torna o código mais limpo, mas também facilita o teste e a manutenção.

```
public static void notificarUsuarios(List<Integer> listaIds, double valorConta) {
    for (Integer id: listaIds) {
        Usuario usuario = Usuario.findById(id);

        if (usuario != null) {
            if(valorConta > usuario.saldo) {
                if(usuario.quantidadeNotificacoes == 4) {
                    Notifications notification = new Notifications(new Date(), usuario.id, "Voce foi notificado mai
                }
                Notifications notification = new Notifications(new Date(), usuario.id, "Sem saldo para realizar o p
            }

        }

    }
}
```

- Atividade: Refatore o código abaixo para seguir com os princípios do clean code citados até aqui.

```
package CleanCode;

import java.util.Date;
import java.util.List;

public class Exercicio1{}

class Usuario{
        public int id;
        public double saldo;
        public int quantidadeNotificacoes;
        Usuario(Integer id, double saldo, int quantidadeNotificacoes){}

        public static Usuario findById(Integer id){
        return new Usuario(1, 100, 2);
        }
}

class Notifications{
        Notifications(Date data, int usuarioId, String mensagem){}
        public static void create(Notifications notification){
}


public static void notificarUsuarios(List<Integer> listaIds, double valorConta) {
            for (Integer id: listaIds) {
                Usuario usuario = Usuario.findById(id);

                if (usuario != null) {
                        if(valorConta > usuario.saldo) {
                                if(usuario.quantidadeNotificacoes == 4) {
                                new Notifications(new Date(), usuario.id, "Voce foi notificado mais de 3
                         vezes. Sua assinatura foi cancelada");
                                }
                                else {
                                new Notifications(new Date(), usuario.id, "Sem saldo para realizar o
                         pagamento");
                                }

                        }
                }
        }
}
```

Referência: https://blog.rocketseat.com.br/clean-code-qualidade-do-desenvolvimento/