Boas práticas de programação (Clean Code)

1. Nomes significativos: Métodos, nomes de variáveis e etc. devem possuir nomes que significam alguma coisa em relação ao seu objetivo, exemplo:

Errado:

```
private Hora hI;  //representa horário inicial

private Hora hF; //representa horário final

//valida quantidade de horas

public boolean valida(hI,hF){

}
```

Correto:

```
private Hora horarioInicial;

private Hora horarioFinal;

public boolean calculaQuantidadeHoras(horarioIncial, horarioFinal){

}
```

2. Classes e métodos: Nome de classes devem ser substantivos e não conter verbos. Já nomes de métodos devem conter verbos pois eles indicam ações.

Considerar as seguintes métricas para codificação:

· Métodos <= 20 linhas;

· Linha <= 100 caracteres;

· Classe = 200 a 500 linhas.

Ademais, métodos devem realizar somente uma função, caso seja possível extrair em outros blocos partes do método, significa baixa coesão e é necessário refatorar o código.

3. Comentários nos códigos: Evitar comentários desnecessários que podem trazer mais desinformação que informação. Sempre projetar o código de forma que o uso de comentários seja mínimo. (Se é necessário olhar outros trechos do código para entender o comentário, não faz sentindo ter o comentário).

Exemplo de comentário desnecessário:

```
public boolean ultrapassouCargaHoraria(horarioIncial, horarioFinal){

return true; //retorna verdadeiro

}
```

4. Formatação: Respeitar endentação e manter bem estruturado visando sempre facilitar o entendimento do código e evitando códigos mal endentados que precisam ser decifrados. Exemplo:

Errado:

```
If(i<10){
        If(i<1){
                If(i<0){
                }
        else{
        ….
        }
        }
else if(i<2){
}
}
```

Correto:

```
If(i<10){
        If(i<1){
                If(i<0){
                }
                else{
                ….
                }
        }
        else if(i<2){
        }
}
```

5. Evite duplicação de Código: Evitar ambiguidade no código. Não deve possuir diferentes métodos que desempenham as mesmas funções. Se há duplicação de código, há baixa coesão.

6 . Tratamento de erros: Projeto o código de forma que garanta que mesmo quando acontecer algum erro, o código continuar fazendo o que precisa. Erros podem acontecer, e quando acontecerem é responsabilidade do programador garantir que o código não encerrara, sendo necessário prever erros esperados e inesperados.