Dar nomes às coisas é uma atividade constante na vida de um desenvolvedor. Damos nomes para variáveis, métodos, classes, parâmetros, atributos, etc. E a má escolha desses pode impactar e muito no tempo de manutenção.

Veja, por exemplo, o código abaixo, e tente dizer rapidamente o que ele faz:

```
public class Trem {

    private List<Vagao> vagoes;
    private int capacidade;

    public boolean disp(int reservas) {

        int r = 0;
        for(Vagao v : vagoes) {
            r += v.reservados();
        }

        r = capacidade - r;
        return r >= reservas;
    }

}
```

Esse código tem péssimos nomes. As variáveis estão com péssimos nomes; estamos reutilizando a mesma variável para coisas diferentes, etc. Esse código conta a quantidade de lugares já reservados nos vagões do trem, e no fim retorna se ainda há lugares sobrando para a quantidade de pessoas determinada (representado pelo parâmetro `reservas`).

Vamos começar a dar nomes melhores à eles. A variável `r` representa na verdade a quantidade de assentos já reservados. Vamos representar isso no código:

```
    public boolean disp(int reservas) {

        int lugaresJaReservados = 0;
        for(Vagao v : vagoes) {
            lugaresJaReservados += v.reservados();
        }

        lugaresJaReservados = capacidade - lugaresJaReservados;
        return lugaresJaReservados >= reservas;
    }
```

A variável `v` também pode ter um nome melhor. A variável `reservas` também pode indicar que na verdade ela se refere a lugares a serem reservados.

Além disso, uma péssima prática é reutilizar uma mesma variável para "duas coisas diferentes". Veja a variável `lugaresJaReservados`: usamos ela para contar a quantidade de lugares já reservados, mas depois usamos ela para guardar a quantidade de lugares livres. Vamos melhorar isso:

```
    public boolean disp(int lugaresAReservar) {

        int lugaresJaReservados = 0;
        for(Vagao vagao : vagoes) {
            lugaresJaReservados += vagao.reservados();
        }

        return capacidade - lugaresJaReservados >= lugaresAReservar;
    }
```

Já está melhor. Agora nosso `return` contém uma expressão que é difícil de entender. Veja que `capacidade - lugaresReservados` indica a quantidade de lugares livres. Podemos então introduzir uma variável que explica isso melhor. Veja:

```
    public boolean disp(int lugaresAReservar) {

        int lugaresJaReservados = 0;
        for(Vagao vagao : vagoes) {
            lugaresJaReservados += vagao.reservados();
        }

        int lugaresLivres = capacidade - lugaresJaReservados;
        return lugaresLivres >= lugaresAReservar;
    }
```

Agora podemos melhorar o nome do método, e deixar claro que ele nos diz se é possível reservar aquela quantidade de lugares no trem:

```
public boolean podeReservar(int lugaresAReservar);
```

Ótimo. Veja agora como ler o método está muito mais fácil. As variáveis se auto-explicam. O texto parece fluente. Repare também que o tempo todo usamos camel case, pois essa é a convenção do Java. Lembre-se de usar as convenções da sua linguagem.

Sempre que vir algo com um nome não claro, lembre-se de refatorar.