Exercício 1:

1. Pensando em desenvolvimento de uma aplicação, jogo, ferramenta, etc. logo nos vem a cabeça uma “soft skill” chamada de trabalho em equipe. E é muito incomum que um desenvolvedor realmente trabalhe do início ao fim em um projeto cobrindo todas as áreas necessárias, apesar de isso existir, obviamente. Mas como não estamos levando a exceção como regra, devemos nos atentar que por se tratar de um trabalho em equipe tem que existir além de uma boa comunicação entre os integrantes de um time, uma boa comunicação também em código. Isso colabora para que os outros integrantes do time consigam entender os seus códigos. A peça fundamental para tornar o seu código mais comunicável é uma nomenclatura adequada de classes, variáveis, métodos e funções. Quando exercemos essa prática de código limpo, deixamos o nosso código mais limpo e legível de forma que o desenvolvedor que irá sustentar aquele código futuramente não encontre dificuldades em entender o funcionamento do mesmo.

2. É necessário analisar bem o problema, para que quando formos dar manutenção não acabemos gerando outros problemas. Entendê-lo em sua raiz faz com que nossa manutenção seja mais efetiva e possa expurgar completamente o problema, sem empurrá-lo para outra pessoa, ou mesmo mascarar o problema e permitir que o mesmo ocorra até outra pessoa ir lá e tentar resolvê-lo.

3. Aqui basicamente é pegarmos um código e deixarmos ele mais limpo do que quando o pegamos. Isso é bom, uma vez que estamos pensando em grupo e não individualmente. Como supracitado não há desenvolvimento sem equipe/time e por isso, manter um código limpo serve para que não apenas nós futuramente não venhamos a ter nenhum infortúnio, mas que ninguém passe por problemas por culpa nossa. Ter exigência consigo mesmo de que sempre que pegar um código é necessário deixá-lo mais limpo possível.

Exercício 2

1. Esse método possui uma grande quantidade de parâmetros que nem sequer estão devidamente nomeados. Se olharmos para a, b, c, d, e, f o que realmente esses parâmetros pode significar. Particularmente eu perguntaria: o porquê o desenvolvedor anterior não os colorou como um vetor, array ou uma lista de Integer. Perderia muito tempo tentando entender o que cada um faz e isso pode de alguma forma atrasar a entrega.

2. A nomenclatura do método está horrível, e não é possível nem saber o que esse método faz ou não faz. Aqui o princípio ferido é a legibilidade.

3. O princípio aqui é o “Princípio da Menor Sobrecarga Cognitiva”, que basicamente o método está tendo mais de uma responsabilidade. Pelo nome é possível dizer que esse método checa o saldo e atualiza. Mas e se precisarmos apenas checar o saldo sem atualizá-lo. É preciso separar esse método em dois e então dividirmos as responsabilidades. Isso trás mais clareza ao código.