(1)

(a) Usar uma interface ou uma classe abstrata? Por que? Cite exemplos.

(b) Usar herança ou delegação a outros objetos? Por que? Cite exemplos.

*(A resposta serve para as duas perguntas)*

R: A resposta pode ser variável com o problema a ser resolvido.

Os dois estão corretos em ser usados na POO, então cabe ao desenvolvedor escolher o caminho de acordo com o problema.

Exemplo:

Você tem N classes que fazem as seguintes tarefas:

- Salva informação no banco;

- Faz algum tipo de cálculo;

- Exibe uma mensagem na tela.

Neste caso, cabe tanto a implementação de uma interface ou de uma classe abstrata. Poderíamos criar as N classes implementando a interface e escrevendo o código dos métodos em comum *(Salva,Calculo e Exibe),* ou poderíamos criar uma classe abstrata com os métodos em comum sendo métodos virtuais e em cada classe N esse código ser reescrito (ou não, podendo aproveitar o código base já escrito na classe abstrata).

Então por fim, ambas as abordagens estão corretas, dependendo do método adotado, pattern adotado, entre outros fatores que forem adotados para resolver algum problema.

(4) Esta questão aborda o tratamento de erros orientado a objetos.

(a) E boa pratica definir um tipo específico de exceção que estende da classe ́ Exception? Se sim, em quais casos?

R: Sim, quando queremos que o tratamento do erro seja especifico, ou voltando algum tipo de mensagem personalizada, ou mesmo gravando informações do erro, entre outros casos como esse.

(b) Quando você capturaria uma exceção através de clausulas try e catch?

Por que?

(c) Em quais situações você lançaria uma exceção? Cite exemplos.

R: (B) Casos de chamada para outros serviços (Qualquer consumo externo do software, banco, api, dll’s e etc). Esses casos são de extrema importância já que não temos domínio do código do terceiro e o retorno nem sempre pode ser o que esperamos (por N motivos, seja de conexão, seja de retorno nulo ou algum outro erro). (C) Situações como descritas anteriormente, e outras situações que as condições estabelecidas não foram atendidas.

(5) R: O código aparenta estar correto, os métodos debite e credite também não vão conflitar já que os dois métodos acabam alterando lugares diferentes, o débito alteraria a conta corrente e o credite alteraria seu limite de crédito. O que pode gerar concorrência em transações ou em chamadas é o buscaConta(id), já que não é mostrado como o contaDao é implementado nem como está sendo feito a parte de acesso e o contaDao.atualiza() não deveria estar nessa parte do código e sim dentro do código, provavelmente em algum código finally ou algo do gênero, assim evitando atualizações ao mesmo tempo.

(6)