1 - a)Depende do contexto pra qual a classe se destina, uma classe abstrata pode conter ou não conter código de lógica no corpo dos métodos, mas pre ser considerada abstrata de fato, deve conter pelo menos um método sem corpo. Enquanto que a interface possui apenas assinaturas dos métodos, que uma outra classe que implementa a interface deve definir.

Exemplo, se eu tiver uma classe chamada mamífero, posso a declarar como abstrada, criando um método abstrato chamado locomoção, onde as classes que herdarem de mamífero deverão sobrescrever o método locomoção para implementar.

b)Depende do contexto empregado para a solução, mas no geral é indicado que se use composição me relação a herança, pois o comportamento pode ser escolhido em tempo de execução em vez de estar amarrado em tempo de compilação, como acontece com herança.

Exemplo, se eu posso criar uma instancia de uma classe e utilizar um comportamento dela, ao invés de herdar dela todos os comportamentos e caracteristicas da classe mãe.

4 - a) Para que o tratamento dos erros fique totalmente isolado da lógica de negócios. Utilizando exceções, podemos nos preocupar apenas com a lógica de negócio do sistema e só depois com o tratamento de erros.

b) Todas as vezes que seja necessário um tratamento caso haja um erro durante um processo na regra de negócio, para informar ao usuário ou par ter controle em log.

c) Quando no form possível prosseguir a execução a partir do ponto que originou a exceção.

5 - Deveria haver validação pra saber se a conta existe antes de efetuar as operações. Outro ponto é que os métodos deveriam ser sincronos, usando tasks e wait, para resolver o problema da concorrência