1A - Abstrata - Encapsulando somente funcionalidades comuns as classes herdadas, não pode ser instanciada, é usada apenas para herança.

Interfaces - Deixa a implementação para suas classes herdadas, não possui implementação, apenas a assinatura do método, como é firmado um contrato com as classes herdadas, Para podermos usar uma interface, devemos criar uma classe e herdar da interface. Com isso, é obrigatório implementar todos os métodos da interface.

1B - Herança - Atributos das classes bases ajuda os programas orientados a objetos que cresçam com menos complexidade e com reutilização de código proporcionando um desenvolvimento conciso.

Delegação - Você pode chamar o método(referenciado) sem ter que saber em tempo de compilação qual método será chamado e a flexibilidade para implementar qualquer funcionalidade em tempo de execução.

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

(a) É boa prática definir um tipo específico de exceção que estende da classe Exception? Se sim, em quais casos? Objeto null , não encontrado por exemplo pode gerar Exception mas não seria necessariamente um erro pode ser que o item não está mais no estoque.

(b) Quando você capturaria uma exceção através de clausulas try e catch? Por que? Conexões e Persistências com Bando de Dados, Files , uploads e webservice

(c) Em quais situações você lançaria uma exceção? Cite exemplos! Devem ser usadas para relatar e tratar condições de erro.As exceções não devem ser retornadas como um valor retornado ou um parâmetro em vez de serem lançadas.

5 Trabalharia com metados em async e adicionava no creditar um verificar creditar prevendo uma situação de coneccao aberta.  
public async Task debitar(long idConta, double valor)

{

Conta conta = contaDao.buscaConta(idConta);

using (var scope = new TransactionScope(TransactionScopeAsyncFlowOption.Enabled))

{

if (conta.podeDebitar(valor))

{

conta.debite(valor);

contaDao.atualiza(conta);

await contaDao.SaveChangesAsync();

}

else

{

throw new SaldoInsuficienteException();

}

scope.Complete();

}

}

public async Task creditar(long idConta, double valor)

{

Conta conta = contaDao.buscaConta(idConta);

using (var scope = new TransactionScope(TransactionScopeAsyncFlowOption.Enabled))

{

if (conta.podeCreditar(valor))

{

conta.credite(valor);

contaDao.atualiza(conta);

await contaDao.SaveChangesAsync();

}

else

{

throw new ProblemaConectividadeException();

}

scope.Complete();

}

}

6 with kits as

(select

l.nome as ProdudoLimpeza,

a.nome as Alimento,

a.data\_validade as ValidadeAlimento,

(select preco from elemento\_estoque where id = a.id\_elemento\_estoque) as a\_preco,

(select custo from elemento\_estoque where id = a.id\_elemento\_estoque) as a\_custo,

l.data\_validade as l\_data\_validade,

(select preco from elemento\_estoque where id = l.id\_elemento\_estoque) as l\_preco,

(select custo from elemento\_estoque where id = l.id\_elemento\_estoque) as l\_custo,

(select avg(satisfacao) from pesquisa\_mercado where id\_produto\_limpeza = l.id group by id\_produto\_limpeza) as Satisfacao

from alimento a

join produto\_limpeza l

on 1 = 1)

select ProdudoLimpeza, Alimento,

CONVERT(DECIMAL(16,2),((a\_preco + l\_preco)) - (a\_preco + l\_preco \* 15)/100) as PrecoComDesconto,

CONVERT(DECIMAL(16,2),((a\_preco + l\_preco) - (a\_custo + l\_custo))) as LucroKit,

CONVERT(VARCHAR(10),ValidadeAlimento, 103) AS ValidadeKit

from kits k

where datediff(day, getdate(), ValidadeAlimento) < 5

and Satisfacao > 70

order by LucroKit desc