1) (A)

Quando pensar em usar classes abstractas e herança, O LSP este princípio diz que se uma classe Derivada deriva da classe Base, então qualquer código que use Base pode também usar uma instância de Derivada sem efeitos surpreendentes.

fator determinante para o uso de interfaces ou de classes abstratas é a imposição da programação dos métodos. Ora. se várias classes apresentam métodos com a mesma programação base, provavelmente vale a pena usar uma classe abstrata para diminuir a repetição de código (aliás, esta é uma das várias vantagens da orientação a objetos).

Interfaces, por outro lado, tem o objetivo de garantir que diferentes classes, que não têm nenhuma relação entre si, apresentem o mesmo conjunto básico de métodos. Uma abordagem não substitui a outra, muito pelo contrário – o que você mais encontra são classes do .NET (por exemplo) que herdam de outra classe e implementam uma ou mais interfaces

(B)

a herança deve ser pouco utilizada. a delegação é considerada muito superior à herança na maioria dos casos pois entre as suas principais vantagens a delegação permite mudar a associação entre classes em tempo de execução, os objetos podem assumir mais de um comportamento, os projetos são mais simples e reutilizáveis, além de descartar os problemas envolvendo a herança como o acoplamento entre as classes e outros problemas discutidos. A composição é utilizada em diversos Padrões de Projetos comprovando assim a sua grande utilidade e uso no desenvolvimento de projetos melhores.

para se fazer bom uso de herança, o desenvolvedor deve pensar em cada método que a classe filha herdou e sobreescreveu, e lembrar que as pré-condições não podem ser apertadas, e as pós-condições não podem serem afrouxadas.

Isso não é tão simples, e requer muito raciocínio do desenvolvedor, sempre que for reescrever um comportamento.

--------------------------------------------------------------------------------------------------------------------------------

4)

(A)

Sim, depende da interface da sua aplicação. Por exemplo, em uma aplicação Web, a melhor maneira de lidar com uma exceção é escalando-a, exibindo avisos em tela com a mensagem informativa da exceção ou usando uma lógica que lide com o erro internamente, disparando alguma lógica adicional ou reportando o problema em log.

(B)

Quando você precisar tratar exceções que foram lançadas. Determinadas situações podem gerar erros diferentes, o que exige tratamentos diferentes. Para cada tratamento, a tipificação da exceção é uma boa prática para prover à cada situação de erro o tratamento adequado.

(C)

Quando realmente a situação é excepcional para erros que não podem ser interceptados no fluxo normal do programa, como erros de abertura de arquivos, erros de E / S e assim por diante.

--------------------------------------------------------------------------------------------------------------------------------

5)

static IDictionary<Conta, object> LockContas = new ConcurrentDictionary<Conta, object>();

...

public void Creditar (long idConta, double valor ) {

using (var scope = new TransactionScope(TransactionScopeOption.Required)) {

Conta conta = contaDao.buscaConta(idConta );

var lockConta = LockContas.GetOrAdd(conta, new object());

lock (lockConta)

{

try{

conta.credite (valor);

contaDao.atualiza (conta);

}finally{

lockConta.TryRemove(conta, out \_);

}

}

scope.Complete();

}

}

public void Debitar (long idConta , double valor ) {

using (var scope = new TransactionScope(TransactionScopeOption.Required)) {

Conta conta = contaDao.buscaConta(idConta);

var lockConta = LockContas.GetOrAdd(conta, new object());

lock (lockConta)

{

try{

if (conta.podeDebitar(valor)){

conta.debite ( valor );

contaDao.atualiza(conta);

} else {

throw new SaldoInsuficienteException ();

}

}finally{

lockConta.TryRemove(conta, out \_);

}

}

scope.Complete();

}

}

----------------------------------------------------------------------------------------------------------------------------

link do github dos códigos fonte:

https://github.com/jaelsonrc/Crud-C-DotNet-Core