Avaliação Técnica

Respostas

Questão 1 –A

Dependendo do escopo do projeto usaremos as duas, ou uma ou outra. No caso do C# que não suporta heranças múltiplas, utilizamos as interfaces para implementá-las.

Usamos as classes abstratas no propósito de prover uma base de definições que indicam para as classes derivadas como irão trabalhar, permitindo assim os programadores implementar nas classes derivadas. A vantagem é que força a hierarquia para todas as subclasses.

Usamos as interfaces no propósito de fornecer um tipo de contrato, onde são especificados métodos, atributos e funções onde as classes que implementam as interfaces são obrigadas a implementar. Vantagem da interface é que uma classe pode implementar diversas interfaces.

Se todas as implementações compartilham a assinatura do método a interface atenderá melhor.

Já se as implementações são todas do tipo e compartilham um comportamento e status comum, a classe abstrata atenderá melhor.

Ligeira vantagem em utilizar classes abstratas por possuir maiores recurso.

Exemplos:

Classe Abstrata : public abstract class Forma  
    {  
        private double \_area;  
        private string \_cor;  
        private double \_perimetro;  
  
        public string Cor  
        {  
            get  
            {  
                return \_cor;  
            }  
            set  
            {  
                \_cor = value;  
            }  
        }  
  
        public double Area  
        {  
            get  
            {return \_area;}  
            set  
            {\_area = value;}  
  
        }  
          
        public double Perimetro  
        {  
            get  
            {return \_perimetro;}  
            set  
            { \_perimetro = value; }  
        }  
  
        public **abstract** void CalcularArea();  
        public **abstract** void CalcularPerimetro();  
        public string Descricao()  
        {  
            return "Sou a classe abstrata Forma.";  
        }  
    }

public class Quadrado : Forma  
    {  
        private double lado;  
        public double Lado  
        {  
            get  
            {return lado;}  
            set  
            {lado = value;}  
        }  
  
        public **override** void CalcularArea()  
        {  
            this.Area = lado \* lado;  
        }  
  
        public **override** void CalcularPerimetro()  
        {  
            this.Perimetro = 4 \* lado;  
        }  
    }

Exemplo Interface:

interface IEquatable<T>

{

bool Equals(T obj);

}

public class Car : IEquatable<Car>

{

public string Make {get; set;}

public string Model { get; set; }

public string Year { get; set; }

// Implementation of IEquatable<T> interface

public bool Equals(Car car)

{

return this.Make == car.Make &&

this.Model == car.Model &&

this.Year == car.Year;

}

}

Resposta 1 – B

A herança é um mecanismo que expressa similaridades entre classes. Simplificando a definição de classes similares torna atributos e serviços comuns a uma herança de classes.

EX. Uma subclasse “É-UM”; “É UM TIPO DE” – Vantagem captura o que é comum e isola o que é diferente.

Public class Pessoa()

{

public string nome{get; set;}

public string tipo{get; set;}

public string email {get; set;}

public string telephone {get; set;}

}

Public class pessoaJuridica : Pessoa

{

Public string cnpj {get; set;}

}

Public class pessoaFisica : Pessoa

{

Public string cpf {get; set;}

}

Todas herdam da classe Pessoa, porém podem mudar os papéis.

A delegação é um modo mais geral de estender uma classe, onde um objeto em vez de realizar uma de suas tarefas, delega para um objeto auxiliar associado.

Ex: A agregação de classe : “TEM-UM”; “É COMPOSTO POR”

Vantagem o comportamento pode ser escolhido em tempo de execução.

Public Class Pessoa

{

Public string nome {get; set;}

Public string endereço {get; set;}

Public void GetNome();

Public void SetNome();

Public void GetEnd();

Public void SetEnd();

}

Usa Usa

0..1

Public Class Agente

{

Public long idAgente{get; set;}

Public void GetId();

Public void SetId();

Public void GetNome();

Public void SetNome();

Public void GetEnd();

Public void SetEnd();

}

Public Class Passageiro

{

Public string numeroSmiles {get; set;}

Public void GetSmiles();

Public void SetSmiles();

Public void GetNome();

Public void SetNome();

Public void GetEnd();

Public void SetEnd();

}

Public Class Tripulacao

{

Public long idTripulacao {get; set;}

Public void GetId();

Public void SetId();

Public void GetNome();

Public void SetNome();

Public void GetEnd();

Public void SetEnd();

}

Delegação todas usam os métodos da classe Pessoa que são comuns.

Resposta 2 – Desenvolvida junto com a solução VS.

Resposta 3 – Desenvolvida junto com a solução VS.

Resposta 4 – A

Sim, pois o tratamento de exceções ajuda a aprimorar a tolerância a falhas de um sistema.

Caso 1 – Divisão por zero / Tratamento de Arquivos / Conexão com Banco de Dados / Conexão com a Rede;

Resposta 4 – B

Quando um código potencialmente poderá gerar uma exceção e o código pode se recuperar dessa exceção, mas que esta exceção seja difícil de prever.

Porque os blocos IF que trataria exceções esperadas não são suficientes para tratar algum código que pode lançar uma exceção não esperada,

Resposta 4 – C

Devemos lançar exceções quando o método não puder concluir sua funcionalidade; Quando é feita uma chamada inadequada a um objeto; Quando um argumento para um método causa exceção.

Ex 1: Quando o método não puder concluir sua funcionalidade definida:

Static void CopiarObjeto(ClasseEx ex)

{

If (ex==null)

{

Throw new system.argumentexception(“Parâmetro valor nulo”);

}

}

Ex 2: Quando é feita uma chamada inadequada a um objeto

Public Class Log

{

System.IO.filestream arqlog = null;

Void AbrirLog(System.Io.FileInfo nomeArq, System.Io.FileMode Mode);

Void EscreverLog()

{

If(!this.ArqLog.CanWrite)

{

Throw new System.InvalidOperationException(“ArqLog não pode ser somente leitura!”);

}}}

Ex 3 : Quando um argumento para um método causar exception;

Static int PegarValordoArray(int[] Array, int x)

{

Try

{ return array[x];}

Catch (System.IndexOutOfRangeException ex)

{System.ArgumentException ArgEx = new System.ArgumentException(“Indice está for a do Limite”, ex);}}

Resposta 5 – Ao analisar a solução poso perceber que se tivermos um problema no método Debite(Valor) por exemplo, e esta sem estar dentro de um bloco de controle de transação a conta onde o valor for debitado poderá ficar com saldo menor e a informação do saldo não ser atualizada.

Neste caso o ideal seria que as chamadas do serviço pelo cliente seja feita através de um TRANSACTIONSCOPE ou que o serviço tenha em suas configurações de binding a propagação de transações que permitirá ao cliente do serviço encaminhar uma transação existente, fazendo com que a execução da operação também faça parte dela, ao invés do WCF cria-la.

Resposta 6 –

create table Elemento\_Estoque

( id bigint IDENTITY(1,1) PRIMARY KEY NOT NULL,

  preco decimal(19,2),

  cnpj\_fabricante varchar(50),

  custo decimal(19,2)

)

create table Estoque

(id bigint IDENTITY(1,1) primary key NOT NULL,

Quantidade int,

id\_elemento\_estoque bigint foreign key (id\_elemento\_estoque) references dbo.Elemento\_Estoque (id) not null

)

create table Alimento

(id bigint IDENTITY(1,1) primary key NOT NULL,

data\_validade datetime,

nome varchar(200),

id\_elemento\_estoque bigint foreign key (id\_elemento\_estoque) references dbo.Elemento\_Estoque (id) not null,

peso decimal

)

create table Produto\_Limpeza

(id bigint IDENTITY(1,1) primary key NOT NULL,

data\_validade datetime,

volume decimal,

id\_elemento\_estoque bigint foreign key (id\_elemento\_estoque) references dbo.Elemento\_Estoque (id) not null,

nome varchar(200)

)

create table Pesquisa\_Mercado

(id bigint IDENTITY(1,1) primary key NOT NULL,

satisfacao int,

instituto\_pesquisa varchar(200),

volume decimal,

id\_produto\_limpeza bigint foreign key (id\_produto\_limpeza) references dbo.Produto\_Limpeza (id) not null

)

select \*

from [dbo].[Alimento]

select \*

from [dbo].[Elemento\_Estoque]

select \*

from [dbo].[Estoque]

select \*

from [dbo].[Pesquisa\_Mercado]

select \*

from [dbo].[Produto\_Limpeza]

Query:

select a.\*, c.\*, b.preco, b.custo, (b.preco\*15/100) as [Desconto\_15%]

from Produto\_Limpeza as a

left join Elemento\_Estoque as b

                on a.id\_elemento\_estoque = [b.id](http://b.id/)

left join [Pesquisa\_Mercado] as c

                on [a.id](http://a.id/) = c.id\_produto\_limpeza

where c.satisfacao > 70

and datediff(day, getdate(), data\_validade) < 5

Resposta 7 – Diagrama de Classes

Atendente

numAtendente

1 1

Pessoa

idPessoa

Nome

SobreNome

Cpf

Telefone

Celular

Endereços

Endereço

idEndereco

Rua

Cep

Estado

Cidade

Complemento

1..\*

Cliente

NumCliente

1

1

0..\*

Massagem

idMassagem

cliente

atendente

DataAgendamento

TipoMassagem

0..\*

Historico\_Massagem

idHistorico

cliente

Massagem

1

1..\*

0..1

TipoMassagem

idTipoMassagem Nome tempoMassagem