**OBS: Perguntas em que realizei pesquisas para responder o conceito básico, pois não tinha conhecimento relevante sobre, estão sinalizadas NA COR VERMELHA.**

**Programação:**

**#Utilizei C# para as respostas.**

1. **O que é variável? Dê exemplo (mais de um);**

Variável é um componente que você cria para armazenar valores, podendo ser eles numéricos (int, double, float) ou caracteres (string por exemplo). No caso da variável, o valor armazenado pode ser alterado na memória.

Exemplo(C#):

variavel1 int = 0;

variavel2 string;

1. **O que é constante? Dê exemplos (mais de um);**

As constantes ficam armazenadas em um espaço na memória, mas diferentemente das variáveis, ela não pode ser alterada ao longo do código/tempo.

1. **O que é declaração de variável/constante? Dê exemplo (mais de um);**

A declaração das variáveis/ constantes determina-se no caso do C# em locais pré-definidos, onde estipula-se o nome, o tipo e no caso das variáveis numéricas atribui-se o valor inicial delas.

Exemplo:

double var1 = 0;

1. **O que é loop? Dê exemplos (mais de um);**

Loop é um laço de repetição/estrutura de repetição que efetua os comandos dentro do laço até que a condição do loop seja satisfeita. Exemplos:

|  |  |
| --- | --- |
| **While**  while (condição)  {  // Sentença para ser executada;  } | **Do... while**  do  {  //Sentença para ser executada;  }  while(condição); |
| **For**  for (inicialização; condição; incremento)  {  // Sentença para ser executada;  } | **\* Foreach**  foreach(<tipo de dado> <nome> in <lista>)  {  //Sentença para ser executada;  } |

**Observações:**

* **Do... while** – Garante a inicialização, execução da sentença pelo menos uma vez antes de verificar a condição do while. No caso da utilização apenas do while, dependendo do caso o código pode nunca ser executado.
* **While** – Tratando-se de uma estrutura booleana, em que sempre será retornado verdadeiro ou falso, a sentença do código só será executada enquanto a condição retornar verdadeiro. A incrementação é necessária na sentença, caso contrário a condição nunca mudará de valor, nunca será falsa, então o programa entrará em loop infinito.
* **For /While** – A diferença se dá principalmente na estrutura, visto que a inicialização não ocorre no while apenas no for, e na incrementação, realizada para interromper o loop, que acontece na própria syntax no caso do for e no caso do while, que possui uma syntax mais simples, esta deverá ser implementada na sentença, dentro das chaves, para interromper o loop.

1. **O que é estrutura de decisão? Dê exemplos (mais de um);**

Estrutura de decisão pode ser representada quando o código deverá tomar a decisão com base na expressão condicional. Exemplo: if

**If:**

if (condição)

{

//Sentença para ser executada caso satisfaça a condição do if (Quando condição = true);

}

else

{

//Sentença para ser executada caso não satisfaça a condição do if (Quando condição=false);

}

Exemplo de código com if e else em que é verificado se a média(método da classe Média) é calculada e for menor que 5, o aluno será reprovado e informará a nota da rec(txtN3.Visible true;) para ser calculada no método MedObj.calcrec . E Se a media não for menor que 5, sera calculada normalmente no método MedObj.calc.

if (MedObj.media < 5)

{

txtmed.Text = Convert.ToString(MedObj.calcrec(float.Parse(txtN1.Text), float.Parse(txtN2.Text), float.Parse(txtN3.Text), float.Parse(txtmed.Text)));

lblResult.Text = "Reprovado.";

txtN3.Visible = true;

}

else

{

txtmed.Text = Convert.ToString(MedObj.calc(float.Parse(txtN1.Text), float.Parse(txtN2.Text), float.Parse(txtmed.Text)));

lblResult.Text = "Aprovado." }

**\*Swift:** Executa seletivamente os comandos conforme o case satisfaz o valor da expressão. \*Mais utilizado em numérico\*. Exemplo da estrutura:

swith(exp)

{

case condicao1:

comando;

break;

case condicao2:

comando;

break;

default;

break

}

1. **O que chamamos de exceção? Dê exemplo;**

O Tratamento de exceção é utilizado para lidar com possíveis erros ou situações inesperadas no algoritmo que o sistema deve estar preparado para receber. Exemplo: Try Catch, onde o catch trabalha o caso inesperado, e pode haver mais de um catch para tratar mais de uma exceção.

**Try Catch:**

|  |  |
| --- | --- |
| **Estrutura:**  try  {  //Código onde pode ocorrer o erro.  }  Catch  {  //Código onde será tratado o erro.  } | **Exemplo:**  try  {  int i = 0;  Console.WriteLine(10 / i);  }  catch(DivideByZeroException ex)  {  Console.WriteLine(ex.Message);  } |

1. **Qual a diferença entre linguagem de programação orientada á objetos e não orientada á objetos? Dê 3 exemplos de cada tipo;**

A linguagem de programação orientada a objetos possui como principal característica a organização do código sendo utilizada em grande escala nos dias de hoje por conseguir solucionar os problemas e trazê-los de maneira mais próxima ao Software do que a programação estruturada. A POO, possui diversas características que as classificam, como por exemplo o uso de classes (onde encapsulam-se os objetos que devem obedecer às características da classe), reaproveitamento de códigos (com a criação de diversos método torna-se mais fácil reutiliza-los em outros softwares e garantir cada vez mais a melhora dos programas), utilização de camadas de programação, entre outras. Exemplo: C#, Java, Python

A programação não orientada a objetos ou programação estruturada (PE), porém é diferente, ela já não depende dos conceitos da POO, não é organizada em camadas, não possui classes, todos os códigos no mesmo arquivo, sendo possível utilizar dados de variáveis em quaisquer partes do código, com a programação reduzida em 3 estruturas: sequência, decisão e iteração. Exemplos: COBOL, C, Perl

1. **\*Versionamento de código, sabe o que é e para quê é usado? Dê exemplos de ferramentas usadas atualmente;**

Versionamento de código trata-se de um software que possui como principal função permitir o gerenciamento das versões de um programa, tendo em vista que um programa é realizado por partes e que mais de uma pessoa terá acesso a ele, é imprescindível a utilização de um software para o controle das versões de um código para que não haja perda de nenhuma parte de um código que geraria um bug ocasionando diversos problemas. A ferramenta mais utilizada para o versionamento atualmente é o Git, que permite a criação de versões do sistema, voltar a versões anteriores, analisar de maneira geral o histórico do código.

**Orientação a objetos:**

1. **O que é herança? Dê exemplo;**

Herança é um conceito fundamental a programação orientada a objetos, haja vista que ela permite compartilhar métodos, argumentos de uma classe pai (superclass) para uma classe filho (uma subclasse (inherits – classe filho herda de uma classe pai) ou uma classe estendida (extends – classe filho estende uma classe pai)). Exemplo:

public class A()

{

objx;

}

public class B:A

1. **\*O que é interface? Dê exemplo;**

Interface tem como objetivo reduzir código complexos, permitindo fazer códigos mais flexíveis, definindo um contrato/modelo para uma classe puxando membros definidos na implementação da Interface.

interface ISampleInterface

{

void SampleMethod();

}

class ImplementationClass : ISampleInterface

{

// Explicit interface member implementation:

void ISampleInterface.SampleMethod()

{

// Method implementation.

}

static void Main()

{

// Declare an interface instance.

ISampleInterface obj = new ImplementationClass();

// Call the member.

obj.SampleMethod();

}

}

1. **O que é uma classe? Dê exemplo;**

Uma classe é uma estrutura fundamental que atua como “template” combinando ações, métodos, funções relacionadas ao conjunto de objetos instâncias que as pertence, que tenham características e comportamento similares. As classes tem as features de seus objetos definidas ou seja, o conjunto formado pelo estado e comportamento dos objetos.

|  |  |
| --- | --- |
| **Estrutura:**  //[modificador do nível de acesso/visibilidade] - [classe] - [identificador]  public class Customizada  {      // Campos, propriedades, métodos e eventos vão aqui ...  } | **Exermplo**:  public class Point3D: Point  {  public int z;  public Point3D(int x, int y, int z) :  base(x, y)  {  this.z = z;  }} |

1. **O que é um objeto? Dê exemplo;**

Objetos, chamados também de instâncias, são blocos alocados na memória que possuem um estado e comportamento definidos. Ele pode ser armazenado em variáveis com tipo da variável, valor, etc. Exemplo:

Point p1 = new Point(0, 0);

//criação do objeto p1

1. **\*O que é propriedade da classe? Dê exemplo;**

Propriedades são extensões dos campos, que nomeiam membros com tipos associados, com sintaxe semelhante ao do campo, porém a propriedade não tem locais de armazenamento.

Ex: a classe MyList<T> declara duas propriedades, Count e Capacity, que somente realizam função de leitura e gravação

1. **O que é método da classe? Dê exemplo;**

O método é utilizado para implementar uma ação, um cálculo, que pode ser repetido mais vezes chamando o método novamente reaproveitando o código. O método pode ser executado por uma classe ou objeto chamando-o no código quando o for útil. Exemplo:

class Média

{

//variavéis e declaração de variáveis

public float nota1, nota2, rec, media;

//métodos

public Média(float n1, float n2, float r, float m)

{

this.nota1 = n1;

this.nota2 = n2;

this.rec = r;

this.media = m;

}

//No exemplo o método da classe media é criado.

1. **Um método que retorna um valor geralmente é chamado de \_função­\_. Dê exemplo;**

Exemplo de uma função num método retornando o valor da média(m) que realiza a soma de duas notas (n1,n2)

public float calc(float n1, float n2, float m)

{

m = (n1 + n2) /2 ;

return m;

}

1. **O que é visibilidade de uma classe, método, propriedade ? Dê exemplo;**

Visibilidade da classe ou modificadores de acesso são palavras reservadas para modificar o acesso a membros, classes, métodos, propriedades, variáveis. Exemplos:

Public, protected, internal, private.

public Média(float n1, float n2, float r, float m)

{

this.nota1 = n1;

this.nota2 = n2;

this.rec = r;

this.media = m;

}

Exemplo declarando o método public, para que outras classes possam ter acesso a ele.

1. **O que são tipos/estruturas de dados? Dê exemplo;**

Estruturas de dados é um conceito para organizar os dados de forma racional. Alguns exemplos são pilhas, filas, ordenação, busca, árvore binária. Exemplo:

Stack<string> nomeDaVariavel = new Stack<string>();

1. **O que é referência de objeto? Dê exemplo;**

Com as referências duas variáveis podem fazer referência ao mesmo objeto, porém as ações em um objeto põem influenciar o outro. Exemplo: nome = new String("POO/Java"); criando-se um objeto com uma referência dada.

1. **Qual a diferença entre valor por referência e valor por tipo? Dê exemplo;**

Primitive Type seriam as variáveis de valor por tipo, ou seja quando cria-se uma variável do tipo int ou string é uma primitive type. No entanto, quando cria-se uma variável com a palavra reservada new um objeto ou classe, você cria uma Class Type, que são as variáveis de valor por referência.

1. **O que é polimorfismo? Dê exemplo;**

Polimorfismo pode ser definido como quando classes deriavadas/subclasses de uma mesma superclasse para invocar métodos da classe derivada, onde os objetos da calsse derivada podem ser tratados como os da classe base, com matrizes, referências, etc.

Exemplo de classe base e derivada retornando valor.

public class BaseClass

{

public virtual void DoWork() { }

public virtual int WorkProperty

{

get { return 0; }

}

}

public class DerivedClass : BaseClass

{

public override void DoWork() { }

public override int WorkProperty

{

get { return 0; }

}

}

1. **O que é implementação? Dê exemplo;**

Implementação pode ser utilizada com duas interfaces, quando por exemplo, as duas contém um membro com um mesmo nome, as duas usarão-o na implementação.

public interface IControl

{

void Paint();

}

public interface ISurface

{

void Paint();

}

public class SampleClass : IControl, ISurface

{

// Both ISurface.Paint and IControl.Paint call this method.

public void Paint()

{

Console.WriteLine("Paint method in SampleClass");

}

}

1. **O que é override? Dê exemplo?;**

Overload pode ser definido como um método que subscreve ou substitui o método da classe pai para a classe filho. Como exemplo citado na questão acima, onde a classe derivada substitui um membro da outra.

1. **O que é overload? Dê exemplo;**

Overload pode ser definido como sobrecarga de métodos, métodos com mesmo nome e assinatura (nome e parâmetros de com o método foi declarado). Exemplo:

public float calc(float n1, float n2, float r, float m)

{

m = (n1 + n2) /2 ;

return m;

}

public float calc(float n1, float n2, float r, float m)

{

m = (n1 + n2) / 2;

int mf = m;

m = (mf + r)/2

return m;

}

Exemplo: dois métodos com mesmo nome e parametros que são metodos diferentes e retornam valores diferentes. No primeiro exemplo a media da nota final é dada apenas com a media das 3 notas. No Segundo método é salvo as primeiras notas(n1 e n2) que é alocado emu ma nova variável antes da recuperação(r) e depois realizado a media da nota final com a rec.

1. **O que são classes abstratas? Dê exemplo;**

São classes em que não é permitido instanciar objetos, implementam alguma funcionalidade, porem deixam as outras funcionalidades para sub-classes, são classes modelo. Exemplo:

abstract public class Exemplo

{

public int a;

protected int b;

private int c;

public String toString() { return (“Valores” + a +”;” + b +”;” + c); }

abstract void alerta();

}

1. **O que são classes concretas? Dê exemplo;**

São as classes que possuem métodos, atributos, pode ser instanciada e permite a criação de objetos. No caso referenciando a classe abstrata, as concretas atuam como classes derivadas/subclasses das classes abstratas, que subscrevem os métodos para implementa-los, realizando as outras funcionalidades. Exemplo:

public class Aluno {

private String nome;

private String ra;

public Aluno() { }

public Aluno(String nome, String ra)

{

this.nome = nome;

this.ra = ra;

}

public String Nome

{

get { return this.nome; }

set { this.nome = value; }

}

1. **O que são classes/propriedades/métodos estáticos? Dê exemplo;**

Uma classe estática não pode ser instanciada e não permite métodos, campos, propriedades ou eventos estáticos. Métodos estáticos podem ser sobrecarregados, mas não substituídos, porque pertencem à classe e não a qualquer instância da classe. Por exemplo não é permitido usar new para criar nova classe ou variável.

public class Automobile

{

public static int NumberOfWheels = 4;

public static int SizeOfGasTank

{

get

{

return 15;

}

}

public static void Drive() { }

public static event EventType RunOutOfGas;

}

No exemplo, uma classe não estatica adotando membros estáticos, o que é mais comum. Utiliza-se a palavra reservada static para declarer um membro como estatico.

Todos os exemplos acima deverão estar representados em alguma linguagem orientada á objetos (typescript ou c#, criando classes, etc);