OAT 01

Respostas

01-Qual a diferença entre Polimorfismo, Herança, Encapsulamento e Abstração?

Polimorfismo significa “várias formas”, que é um termo designado a objetos de classes distintas, que podem reagir de uma maneira diferente ao fazerem o uso de um mesmo método já a herança é um termo que se refere a uma
classe nova, que pode ser criada a partir de outra existente. Ela pode “herdar atributos e comportamentos da classe a ser estendida já o encapsulamento é um mecanismo que trata de dar segurança aos objetos, mantendo-os controlados em relação ao seu nível de acesso e por ultimo Abstração nada mais é que a habilidade de modelar
características do mundo real, de um denominado problema em questão que o programador esteja tentando resolver, e determinar como estas características irão atuar ao receber informações.

02-O que é Associação, Agregação, Composição e Generalização?

Associação – A Associação representa a relação existente entre objetos. Ela descreve o vínculo entre duas classes e geralmente determina que as instâncias de uma classe estão de alguma forma ligadas às instâncias da outra classe.
Agregação – Representa uma relação todo-parte entre o agregado e suas partes. Demonstra que as informações de um objeto precisam ser complementadas por um objeto de outras classes. Nessa relação, a perda do todo não resulta, obrigatoriamente, na perda das partes.
Composição – Também representa uma relação todo-parte entre o agregado e suas partes. Representa um vínculo mais forte entre objetos-todo e objetos-parte. Objetos-parte têm que pertencer ao objeto-todo. A composição representa um forte relacionamento, em que as partes não sobrevivem sem o todo.
Generalização / Especialização – representa a relação existente entre classes. Este relacionamento também é conhecido na orientação a objetos como herança. É identificar super-classe (geral) e sub-classes (especializadas). É quando existe a necessidade de criar uma classe que herde as propriedades de outra classe, isto é, os atributos, métodos e também relacionamentos (associações) da classe superior.

03- Crie uma classe de nome Carro e atribua a ela todas as propriedades que você acredita que um carro possua. Instancie a classe Carro e preencha 3 objetos distintos.
Imprima na tela do usuário todos os atributos dos três carros.

```
class Carro:
    def __init__(self, modelo, ano, cor):
        self.modelo = modelo
        self.ano = ano
        self.cor = cor

carro1 = Carro("Nissan Altima", 1992, "Preto")
```

```
carro2 = Carro("Mercedes-Benz E-Class", 1993, "Branco")

carro3 = Carro("Hyundai Tucson", 2004, "Vermelho")]

print("Carro 1:")
print(f"Modelo: {carro1.modelo}")
print(f"Ano: {carro1.ano}")
print(f"Cor: {carro1.cor}")

print("Carro 2:")
print(f"Modelo: {carro2.modelo}")
print(f"Ano: {carro2.ano}")
print(f"Cor: {carro2.cor}")

print("Carro 3:")
print(f"Modelo: {carro3.modelo}")
print(f"Ano: {carro3.ano}")
print(f"Cor: {carro3.cor}")
```

04- Pesquise como um criar um método específico dentro de uma classe em C#.
Crie dois métodos para a classe Carro criada na questão anterior. Um método chamado Acelerar(), que retorna a string “O carro está em movimento” como resposta e outro método Freiar() que também retorna uma string “O carro está parado” como resposta.

```
class Carro
{
        //
        public string Modelo { get; set; }
        public int Ano { get; set; }
        public string Cor { get; set; }
        public int Quilometragem { get; set; }
        public double Preco { get; set; }
        // Método construtor
        public Carro(string modelo, int ano, string cor, int quilometragem,
double preco)
{
        Modelo = modelo;
        Ano = ano;
        Cor = cor;
}
        // Método para acelerar
public string Acelerar()
{
        return "O carro está em movimento";
}
// Método para frear
```

```
public string Freiar()
{
        return "O carro está parado";
}
}
class Program
{
   static void Main()
{
        // Instanciando três carros
        Carro carro1 = new Carro("Nissan Altima", 1992, "Preto");
        Carro carro2 = new Carro("Mercedes-Benz E-Class", 1993, "Branco");
        Carro carro3 = new Carro("Hyundai Tucson", 2004, "Vermelho");

        // Testando os métodos Acelerar e Freiar
        Console.WriteLine("Carro 1: " + carro1.Acelerar());
        Console.WriteLine("Carro 2: " + carro2.Freiar());
        // Imprimindo os atributos dos carros

        Console.WriteLine("\nDetalhes do Carro 1:");
        Console.WriteLine($"Modelo: {carro1.Modelo}");
        Console.WriteLine($"Ano: {carro1.Ano}");
        Console.WriteLine($"Cor: {carro1.Cor}");

        Console.WriteLine("Detalhes do Carro 2:");
        Console.WriteLine($"Modelo: {carro2.Modelo}");
        Console.WriteLine($"Ano: {carro2.Ano}");
        Console.WriteLine($"Cor: {carro2.Cor}");
}
}
```

05-Quais são os benefícios de se criar um diagrama de classes?
Alguns dos benefícios de se criar um diagrama de classes são:
• Ilustrar modelos de dados para sistemas de informação, não importa quão simples ou complexo seja.
• Entender melhor a visão geral dos esquemas de uma aplicação.
• Expressar visualmente as necessidades específicas de um sistema e divulgar essas informações por toda a empresa.
• Criar gráficos detalhados que destacam qualquer código específico necessário para ser programado e implementado na estrutura descrita.
• Fornecer uma descrição independente de implementação de tipos utilizados em um sistema e passados posteriormente entre seus componentes.
06- O que é um array e como é a sua implementação?
Um array é um conjunto de elementos de um mesmo tipo de dados onde cada elemento do conjunto é acessado pela posição no array que é dada através de um índice (uma sequência de números inteiros). Um array de uma dimensão é também conhecido como vetor, e, um array de mais de uma dimensão é conhecido como uma matriz.

O uso de arrays é muito importante em qualquer linguagem de programação, porém é preciso ter atenção ao manusear informações presentes em um array. Iremos limitar ao tamanho de 10 elementos. Para isso também iremos acrescentar métodos para manipular os arrays simulando uma tabela presente em um banco de dados.
07 - Faça um programa em C# (com a estrutura do...while) que leia 20 valores inteiros e:
- Encontre e mostre o maior valor;
- Encontre e mostre o menor valor;
- Calcule e mostre a média dos números lidos;

```
class Program
{
        static void Main()
{
          const int totalValores = 20;
          int contador = 0;
          int valor;
          int soma = 0;
          int maior = int.MinValue; // Inicializado com o menor valor
          int menor = int.MaxValue; // Inicializado com o maior valor
          Console.WriteLine("Digite {0} valores inteiros:",
          totalValores);
do
{
          Console.Write("Digite o {0}º valor: ", contador + 1);
          if (int.TryParse(Console.ReadLine(), out valor))
{
          // Atualizar soma
          soma += valor;
          // Atualizar maior valor

          if (valor > maior)
          maior = valor;
          // Atualizar menor valor

          if (valor < menor)
          menor = valor;
          contador++;
}
  else
{
         Console.WriteLine("digite um valor inteiro
válido.");
}
          } while (contador < totalValores);
          // Calcular média
          double media = (double)soma / totalValores;
          // Exibir resultados
          Console.WriteLine("\Resultados:");
```

```
        Console.WriteLine("Maior valor: {0}", maior);
        Console.WriteLine("Menor valor: {0}", menor);
        Console.WriteLine("Média: {0}", media);
        Console.ReadLine(); // Manter a janela aberta até que o usuário
        pressione Enter
}
}
```

08 - Faça o seguinte programa em C#. Uma loja utiliza o código V para compras à vista e o código P para compras a prazo. Faça um algoritmo que recebe ao código (V ou P) e o valor de 15 transações. Calcule e mostre:

- O valor total das compras à vista.
- O valor total das compras a prazo.
- O valor total das compras efetuadas.

```
class Program
{
        static void Main()
{
const int numTransacoes = 15;
        char codigo;
        double valor;
        double totalV = 0;
        double totalP = 0;
        double valorTotal = 0;
        for (int i = 1; i <= numTransacoes; i++)
{
        Console.Write($"Digite o código (V para à vista, P para a
prazo) da transação {i}: ");
codigo = char.ToUpper(Console.ReadKey().KeyChar);
        if (codigo == 'V' || codigo == 'P')
{
        Console.Write("\Mostre o valor da transação: ");
        if (double.TryParse(Console.ReadLine(), out valor))
{
        // Atualizar valores totais com base no código
        if (codigo == 'V')
        totalV += valor;
        else if (codigo == 'P')
        totalP += valor;
        // Atualizar o valor total das compras efetuadas
        valorTotal += valor;
}
else
{
        Console.WriteLine("Valor inválido. Por favor,
        digite um valor numérico.");
        i--; // Reduz o contador para repetir a transação
        inválida
}
```

```
}
        else
{
        Console.WriteLine("Código inválido. Digite V
        para à vista ou P para a prazo.");
        i--; // Reduz o contador para repetir a transação com
        código inválido
}
}
        // Exibir resultados
        Console.WriteLine("\nResultados:");
        Console.WriteLine("Valor total das compras à vista: {0:C2}",
        totalVista);

        Console.WriteLine("Valor total das compras a prazo: {0:C2}",
        totalPrazo);

        Console.WriteLine("Valor total das compras efetuadas: {0:C2}",
        valorTotal);

        Console.ReadLine(); // Manter a janela aberta até que o usuário
        pressione Enter
        }
        }
```

09 -Faça o seguinte programa em C#. A prefeitura de Luziânia fez uma pesquisa com 200 pessoas, coletando dados sobre o salário e o número de filhos. A prefeitura deseja saber:

- A média do salário dessas pessoas.
- A média do número de filhos.
- O maior salário.
- O menor salário.
- A porcentagem de pessoas com salários até R$1500,00

```
class Program
{
        static void Main()
{
        const int numPessoas = 200;
        double salario;
        int numFilhos;
        double somaSalarios = 0;
        double maiorSalario = double.MinValue;
        double menorSalario = double.MaxValue;
        int somaFilhos = 0;
        int countSalarioAte1500 = 0;
        for (int i = 1; i <= numPessoas; i++)
{
        Console.WriteLine($"Digite os dados da pessoa {i}:");
        // Entrada do salário
```

```
        Console.Write("Salário: R$");
        if (double.TryParse(Console.ReadLine(), out salario))
{
        // Atualizar soma dos salários
        somaSalarios += salario;
        // Atualizar maior e menor salário
        if (salario > maiorSalario)
        maiorSalario = salario;
        if (salario < menorSalario)
        menorSalario = salario;
        // Verificar se o salário está até R$1500,00
        if (salario <= 1500)
        countSalarioAte1500++;
}
else
{
        Console.WriteLine("Salário inválido. Digite
        um valor numérico.");
        i--; // Reduz o contador para repetir a entrada
        inválida
        continue; // Pula para a próxima iteração
}
        // Entrada do número de filhos
        Console.Write("Número de filhos: ");
        if (int.TryParse(Console.ReadLine(), out numeroFilhos))
        {
        // Atualizar soma dos filhos
        somaFilhos += numFilhos;
}
else
{
        Console.WriteLine("Número de filhos inválido. Digite um valor inteiro.");
        i--; // Reduz o contador para repetir a entrada
        inválida
}
}
        // Calcular médias
        double mediaSalario = somaSalarios / numPessoas;
        double mediaFilhos = (double)somaFilhos / numPessoas;
        // Calcular porcentagem de pessoas com salários até R$1500,00
        double percentualSalarioAte1500 = (double)countSalarioAte1500 /
        numPessoas * 100;
        // Exibir resultados
        Console.WriteLine("\nResultados da pesquisa:");
        Console.WriteLine($"Média do salário: R${mediaSalario:F2}");
        Console.WriteLine($"Média do número de filhos:
        {mediaFilhos:F2}");
        Console.WriteLine($"Maior salário: R${maiorSalario:F2}");
```

```
        Console.WriteLine($"Menor salário: R${menorSalario:F2}");
        Console.WriteLine($"Porcentagem de pessoas com salários até
        R$1500,00: {percentualSalarioAte1500:F2}%");
        Console.ReadLine(); // Manter a janela aberta até que o usuário
        pressione Enter
}
}
```

10- Faça um programa em C# que leia uma quantidade indefinida de objetos Carro,
composto pelos atributos, marca, valor, cor, modelo e ano, e:
- Ordene os carros pelo de maior valor;
- Imprima na tela todos os carros ordenados do maior valor para o de menor valor;

```
class Program
{
static void Main()
{
        // Lista para armazenar os carros
        List<Carro> listaCarros = new List<Carro>();
        // Loop para a leitura dos carros
        while (true)
{
        Carro nCarro = LerCarroDoUsuario();
        // Verifica se o usuário deseja encerrar a entrada de
        carros
        if (nCarro == null)
        break;
        listaCarros.Add(nCarro);
}
        // Ordena os carros pelo valor em ordem decrescente
        listaCarros = listaCarros.OrderByDescending(carro =>
        carro.Valor).ToList();
        // Imprime os carros ordenados
        Console.WriteLine("\nCarros Ordenados por Valor (do maior para
        o menor):");
        foreach (var carro in listaCarros)
{
        Console.WriteLine($"Marca: {carro.Marca}, Modelo:
        {carro.Modelo}, Ano: {carro.Ano}, Cor: {carro.Cor}, Valor:
        R${carro.Valor:F2}");
}
}
        static Carro LerCarroDoUsuario()
{
        Console.WriteLine("\nInforme os dados do carro (ou digite
        'sair' para encerrar):");
        Console.Write("Marca: ");
        string marca = Console.ReadLine();
```

```
        if (marca.ToLower() == "sair")
        return null;
        Console.Write("Modelo: ");
        string modelo = Console.ReadLine();
        Console.Write("Ano: ");
        int ano = int.Parse(Console.ReadLine());
        Console.Write("Cor: ");
        string cor = Console.ReadLine();
        Console.Write("Valor: R$");
        decimal valor = decimal.Parse(Console.ReadLine());
        return new Carro
{
        Marca = marca,
        Modelo = modelo,
        Ano = ano,
        Cor = cor,
        Valor = valor
};
}
}
```

11- Baseado no programa anterior (Questão 1) Crie uma interface para Cadastrar,
Excluir e Listar os carros.
- Cadastre um carro.
- Exclua um carro.
- Liste todos os carros do menor valor para o maior valor.

```
interface IOperacoesCarro
{
        void CadastrarCarro();
        void ExcluirCarro();
        void ICarros();
}
        class Carro : IComparable<Carro>
{
        public string Marca { get; set; }
        public decimal Valor { get; set; }
        public string Cor { get; set; }
        public string Modelo { get; set; }
        public int Ano { get; set; }
        // Implementação do método CompareTo para ordenação
        public int CompareTo(Carro outroCarro)
{
        // Ordena por valor em ordem crescente
        return this.Valor.CompareTo(outroCarro.Valor);
}
}
        class Program : IOperacoesCarro
{
```

```
        // Lista para armazenar os carros
        private List<Carro> lCarros = new List<Carro>();
        static void Main()
{
        Program programa = new Program();
        // Loop do programa principal
        while (true)
{
        Console.WriteLine("\nEscolha uma opção:");
        Console.WriteLine("1 - Cadastrar Carro");
        Console.WriteLine("2 - Excluir Carro");
        Console.WriteLine("3 - Listar Carros (do menor valor para o
        maior valor)");
        Console.WriteLine("4 - Sair");
        string escolha = Console.ReadLine();
        switch (escolha)
{
        c "1":
        programa.CadastrarCarro();
        break;
        c "2":
        programa.ExcluirCarro();
        break;
        c "3":
        programa.ListarCarros();
        break;
        c "4":
        // Encerra o programa
        Environment.Exit(0);
        break;
        default:
        Console.WriteLine("Opção inválida. Tente
        novamente.");
        break;
}
}
}
        // Implementação da interface para cadastrar carro
        public void CadastrarCarro()
{
        Console.WriteLine("\nInforme os dados do carro:");
        Console.Write("Marca: ");
        string marca = Console.ReadLine();
        Console.Write("Modelo: ");
        string modelo = Console.ReadLine();
        Console.Write("Ano: ");
        int ano = int.Parse(Console.ReadLine());
        Console.Write("Cor: ");
```

```
        string cor = Console.ReadLine();
        Console.Write("Valor: R$");
        decimal valor = decimal.Parse(Console.ReadLine());
        // Adiciona o novo carro à lista
        lCarros.Add(new Carro
{
        Marca = marca,
        Modelo = modelo,
        Ano = ano,
        Cor = cor,
        Valor = valor
});
        Console.WriteLine("Carro cadastrado com sucesso!");
}
        // Implementação da interface para excluir carro
        public void ExcluirCarro()
{
        if (lCarros.Count == 0)
{
        Console.WriteLine("Nenhum carro cadastrado para excluir.");
        return;
}
        Console.WriteLine("\nEscolha o carro a ser excluído:");
        lCarros();
        int indice;
        do
{
        Console.Write("Digite o número do carro a ser excluído: ");
} while (!int.TryParse(Console.ReadLine(), out indice) ||
        indice < 1 || indice > listaCarros.Count);
        // Remove o carro da lista
        lCarros.RemoveAt(indice - 1);
        Console.WriteLine("Carro excluído com sucesso!");
}
        // Implementação da interface para listar carros
        public void ListarCarros()
{
        if (listaCarros.Count == 0)
{
        Console.WriteLine("Nenhum carro cadastrado.");
        return;
}
        // Ordena os carros pelo valor em ordem crescente
        listaCarros.Sort();
        Console.WriteLine("\nCarros (do menor valor para o maior
        valor):");
        for (int i = 0; i < lCarros.Count; i++)
{
```

```
        Console.WriteLine($"{i + 1} - {lCarros[i].Marca}
{lCarros[i].Modelo} ({lCarros[i].Ano}), Cor:
{lCarros[i].Cor}, Valor: R${lCarros[i].Valor:F2}");
}
}
}
```

12- Escreva um programa em C# que deverá ter as seguintes opções:
- Carregar Vetor.
- Listar Vetor.
- Exibir apenas os números pares do vetor.
- Exibir apenas os números ímpares do vetor.
- Exibir a quantidade de números pares existentes nas posições ímpares do vetor.
- Exibir a quantidade de números ímpares existentes nas posições pares do vetor.
- Sair

```
class Program
{
        static int[] vetor;
        static void Main()
{
        while (true)
{
        Console.WriteLine("\nEscolha uma opção:");
        Console.WriteLine("1 - Carregar Vetor");
        Console.WriteLine("2 - Listar Vetor");
        Console.WriteLine("3 - Exibir somente os números pares do
        vetor");
        Console.WriteLine("4 - Exibir somente os números ímpares do
        vetor");
        Console.WriteLine("5 - Exibir a quantidade de números pares
        nas posições ímpares do vetor");
        Console.WriteLine("6 - Exibir a quantidade de números
        ímpares nas posições pares do vetor");
        Console.WriteLine("7 - Sair");
        string escolha = Console.ReadLine();
        switch (escolha)
{
        c "1":
        CarregarVetor();
        break;
        c "2":
        ListarVetor();
        break;
        c "3":
        ExibirPares();
        break;
        c "4":
```

```
        ExibirImpares();
        break;
        c "5":
        ContarParesNasPosicoesImpares();
        break;
        c "6":
        ContarImparesNasPosicoesPares();
        break;
        c "7":
        Environment.Exit(0);
        break;
        default:
        Console.WriteLine("Opção inválida. Tente
        novamente.");
        break;
}
}
}
        static void CarregarVetor()
{
        Console.Write("\nInforme o tamanho do vetor: ");
        int tamanho = int.Parse(Console.ReadLine());
        vetor = new int[tamanho];
        for (int i = 0; i < tamanho; i++)
{
        Console.Write($"Informe o valor para a posição {i + 1}: ");
        vetor[i] = int.Parse(Console.ReadLine());
}
        Console.WriteLine("Vetor carregado com sucesso!");
}
        static void ListarVetor()
{
        if (vetor == null)
{
        Console.WriteLine("Vetor não carregado. Utilize a opção 1
        para carregar o vetor.");
        return;
}
        Console.WriteLine("\nVetor:");
        for (int i = 0; i < vetor.Length; i++)
{
        Console.WriteLine($"Posição {i + 1}: {vetor[i]}");
}
}
        static void ExibirPares()
{
        if (vetor == null)
{
```

```
        Console.WriteLine("Vetor não carregado. Utilize a opção 1
        para carregar o vetor.");
        return;
}
        Console.WriteLine("\nNúmeros Pares do Vetor:");
        for (int i = 0; i < vetor.Length; i++)
{
        if (vetor[i] % 2 == 0)
{
        Console.WriteLine($"Posição {i + 1}: {vetor[i]}");
}
}
}
        static void ExibirImpares()
{
        if (vetor == null)
{
        Console.WriteLine("Vetor não carregado. Utilize a opção 1
        para carregar o vetor.");
        return;
}
        Console.WriteLine("\nNúmeros Ímpares do Vetor:");
        for (int i = 0; i < vetor.Length; i++)
{
        if (vetor[i] % 2 != 0)
{
        Console.WriteLine($"Posição {i + 1}: {vetor[i]}");
}
}
}
        static void ContarParesNasPosicoesImpares()
{
        if (vetor == null)
{
        Console.WriteLine("Vetor não carregado. Utilize a opção 1
        para carregar o vetor.");
        return;
}
        int contador = 0;
        for (int i = 1; i < vetor.Length; i += 2)
{
        if (vetor[i] % 2 == 0)
{
        contador++;
}
}
                Console.WriteLine($"\nQuantidade de Números Pares nas Posições
        Ímpares: {contador}");
```

```
}
        static void ContarImparesNasPosicoesPares()
{
        if (vetor == null)
{
        Console.WriteLine("Vetor não carregado. Utilize a opção 1
        para carregar o vetor.");
        return;
}
        int contador = 0;
        for (int i = 0; i < vetor.Length; i += 2)
{
        if (vetor[i] % 2 != 0)
{
        contador++;
}
}
        Console.WriteLine($"\nQuantidade de Números Ímpares nas
        Posições Pares: {contador}");
}
}
```