Classes Seladas e Abstratas e Interfaces

? EXERCÍCIO 1: CAMINHO DE PASTAS E SUBPASTAS

```
using System;
namespace SAI1
{public class Pasta

 {protected string Desig;
  public Pasta(string D)
   {Desig = D;}}

 public class SubPasta1 : Pasta
  {protected string SubDesig;
   public SubPasta1(string DPasta, string DSubPasta):base(DPasta)
    {this.SubDesig = DSubPasta; }}

 public sealed class SubSubPasta : SubPasta1
  {private string SubSubDesig;
   public SubSubPasta(string DPasta, string DSubPasta, string
   DSubSubPasta): base(DPasta, DSubPasta)
    {SubSubDesig = DSubSubPasta; }
  public string LerDesig()
   {return base.Desig + "\\" + base.SubDesig +
   "\\"+SubSubDesig;}}

 public class ImpressaoDoCaminho
  {static void Main(string[] args)
   {SubSubPasta P = new SubSubPasta("Estatistica", "Aulas",
   "Slides1");
    Console.WriteLine("Caminho desde a raiz: {0}",
    P.LerDesig());}}}
```

? EXERCÍCIO 2: LIMITES PARA ESTRADAS

```
using System;
namespace SAI2

{public abstract class Estradas
 {protected string Desig;
  protected int Minimo;
  protected int Maximo;
  public Estradas(string D, int Mi, int Ma)
```

```
   {this.Desig=D;
    this.Minimo=Mi;
    this.Maximo=Ma; }
  public abstract string LerDesig();
  public abstract int LerVelocidadeMax();
  public abstract int LerVelocidadeMin();}

 public class AutoEstrada : Estradas
  {public AutoEstrada(string D, int Mi, int Ma):base(D, Mi, Ma)
   {}
  public override string LerDesig()
   {return Desig; }
  public override int LerVelocidadeMax()
   {return Maximo ; }
  public override int LerVelocidadeMin()
   {return Minimo;}}

 public class LimitesParaEstradas
  {static void Main(string[] args)
   {AutoEstrada A = new AutoEstrada("AutoEstrada",50,120);
    Console.WriteLine("Limites para {0}", A.LerDesig());
    Console.WriteLine("Max em autoestrada: {0} km/h",
    A.LerVelocidadeMax());
    Console.WriteLine("Min em autoestrada: {0} Km/h",
    A.LerVelocidadeMin());}}}
```

➲ **EXECUÇÃO**

```
Limites para AutoEstrada
Max em autoestrada: 120 km/h
Min em autoestrada: 50 Km/h
```

? EXERCÍCIO 3: AVALIAÇÃO DE EMPRESAS

```
using System;
namespace SAI3
{public abstract class AvaliacaoOrg
 {protected string Desig;
  public AvaliacaoOrg(string D)
   {Desig=D;}
 public abstract double Valor();
 public override string ToString()
  {return "Avaliação de " + Desig+"=";}}

public class ONG : AvaliacaoOrg
 {private double Patro, VPatro, Custos;
```

```
 public ONG(double P, double V, Double C, string D): base (D)
  {Patro = P;
   VPatro = V;
   Custos = C;}
 public override double Valor()
  {return Patro * VPatro - Custos; }}

public class Comercial : AvaliacaoOrg
 {private double Custos, Proveitos;
  public Comercial(double C, double P,string D): base ( D)
  {Custos = C;
   Proveitos = P;}
  public override double Valor()
  {return Proveitos - Custos; }}
 class AvaliacaoConjOrg
  {static void Main(string[] args)
   {AvaliacaoOrg[] Org = new AvaliacaoOrg[2];
    Org[0] = new Comercial(50000, 200000, " E1");
    Org[1] = new ONG(30, 20000, 10000," E2");
    double V,Totv = 0;
    for (int i = 0; i < Org.Length; i++)
     {V = Org[i].Valor();
      Console.WriteLine(Org[i].ToString()+ V +" euros");
      Totv += V;}
    Console.WriteLine("Valor Total das empresas {0, 8:F} euros",
    Totv);}}}
```

? EXERCÍCIO 4: NOME COMPLETO

```
using System;
namespace SAI4
{public interface INome
 {string NomeCompleto(); }
 public class Professor: INome
  {private string Titulo;
   private string Nome;
   private string Apelido;
   public Professor (string T, string N, string A)
    {Titulo=T;
     Nome=N;
     Apelido=A;}
   public string NomeCompleto()
    {return "Prof"+" "+Titulo + " "+ Nome+" "+Apelido; }}

 public class Aluno : INome
  {private string Nome;
```

```
    private string Apelido;
    public Aluno(string N, string A)
     {Nome = N;
      Apelido = A;}
    public string NomeCompleto()
     {return "Aluno:"+ " " +Nome + " " + Apelido;}}

 public class ProfEAlunos
  {static void Main(string[] args)
   {Professor P = new Professor("Dr", "João", "Pinto");
    Aluno A = new Aluno("Pedro", "Paiva");
    Console.WriteLine(P.NomeCompleto());
    Console.WriteLine(A.NomeCompleto());}}}
```

? EXERCÍCIO 5: ÁREAS E PERÍMETROS

```
using System;
namespace SAI5
{public interface IMedidas
   {double Area ();
    double Perimetro();}

 public class Quadrado: IMedidas
 {private double Lado;
  public Quadrado (double L)
   {Lado=L;}
  public double Area()
   {return Lado*Lado; }
  public double Perimetro()
   {return 4*Lado; }}

 public class Triangulo: IMedidas
  {private double Base;
   private double Altura;
   private string Tipo;
   public Triangulo (double B, double A, string T)
    {Base=B;
     Altura=A;
     Tipo=T;}
   public double Area()
    {return Base*Altura/2;}
   public double Perimetro()
    {double P, Hip;
     switch (Tipo)
     {case "Equilatero":
        P=3*Base;
```

```
        break;
      case "Isosceles":
        Hip = Math.Sqrt(Math.Pow(Base/2,2)+Math.Pow(Altura, 2));
        P = Base+2*Hip;
        break;
      default:
        P=Base+(Base+1)+(Base+2);
        break;}
     return P;}}

 public class AreasEPerimetros
  {static void Main(string[] args)
   {Quadrado Q = new Quadrado(2);
    Triangulo T = new Triangulo(2,3,"Isosceles");
    Console.WriteLine("Quadrado: Área={0} Perímetro={1}"
    ,Q.Area(),Q.Perimetro());
    Console.WriteLine("Triângulo: Área={0} Perímetro={1,4:F2}"
    ,T.Area(),T.Perimetro());}}}
```

? EXERCÍCIO 6: INTERFACE PARA LEITURA DE NOME E CÁLCULO DE IDADE

```
public interface IPessoa
 {string NomeCompleto {get;}
  int Idade();}
```

? EXERCÍCIO 7: IMPLEMENTAÇÃO DO INTERFACE PARA PESSOA

```
using System;
namespace SAI7
{public interface IPessoa
 {string NomeCompleto {get;}
  int Idade();}

public class Utente : IPessoa
 {private string NomeProprio;
 private string Apelido;
 private int AnoNasc;
 public Utente(string NP, string A, int Ano)
  {NomeProprio = NP;
   Apelido = A;
   AnoNasc = Ano;}
 public string NomeCompleto
  {get
   {return NomeProprio + " " + Apelido;}}
```

```
  public int Idade()
   {return DateTime.Now.Year - AnoNasc;}}

  public class NomesEIdades
   {static void Main(string[] args)
    {Utente[] U=new Utente[5];
     U[0]= new Utente("Teresa", "Paiva", 1970);
     U[1]= new Utente("Joaquim", "Aires", 1980);
     U[2]= new Utente("Rui", "Taveira", 1995);
     U[3]= new Utente("Rosa", "Batista", 1960);
     U[4]= new Utente("Telmo", "Correia", 1950);
     for (int P=0; P<U.Length; P++)
      Console.WriteLine("{0} tem {1} anos", U[P].NomeCompleto,
      U[P].Idade());}}}
```

? EXERCÍCIO 8: EMPREGADOS E SALÁRIOS

```
using System;
namespace SAI8
{public interface IEmpregado
 {double Salario();
  int Antiguidade(int AnoCorrente);}

public abstract class Pessoa
 {protected string Nome;
  public Pessoa(string N)
   {Nome=N;}
  public abstract string LerNome();}

 public class Empregado : Pessoa, IEmpregado
  {private int AnoAdmissao;
   private double SalHora;
   private int Horas;
   public Empregado(string D, int Ano, double S, int H): base(D)
    {AnoAdmissao = Ano;
     SalHora = S;
     Horas = H;}
   public override string LerNome()
    {return Nome;}
   public double Salario()
    {return SalHora * Horas; }
   public int Antiguidade(int AnoCorrente)
    {return (AnoCorrente - AnoAdmissao);}}

public class EmpregadosESalarios
 {static void Main(string[] args)
```

```
  {Empregado E= new Empregado("J. Costa", 1998, 6.10, 30);
   Console.WriteLine("Nome {0}", E.LerNome());
   Console.WriteLine("Antiguidade: {0} anos",
   E.Antiguidade((System.DateTime.Now.Year)));
   Console.WriteLine("Salário: {0} Euros", E.Salario());}}}
```

? EXERCÍCIO 9: SERVIÇO SOCIAL

```
using System;
namespace SAI9
{public interface IPessoa
 {string NomeCompleto {get;}
  int Idade();}

 public class ServicoSocial
  {protected string Centro;
   public ServicoSocial(string C)
    {Centro = C; }
   public string PCentro
    {get
     {return Centro;}}}

  public class Utente : ServicoSocial, IPessoa
   {private string NomeProprio;
    private string Apelido;
    private int AnoNasc;
    public Utente(String C, string NP, string A, int Ano):base(C)
     {NomeProprio = NP;
      Apelido = A;
      AnoNasc = Ano;}
    public string NomeCompleto
     {get
      {return NomeProprio + " " + Apelido; }}
    public int Idade()
     {return DateTime.Now.Year - AnoNasc; }}

 public class LocalNomesEIdades
  {static void Main(string[] args)
   {Utente[] U=new Utente[5];
    U[0]= new Utente("Porto", "Teresa", "Paiva", 1970);
    U[1]= new Utente("Lisboa", "Joaquim", "Aires", 1980);
    U[2]= new Utente("Braga","Rui", "Taveira", 1995);
    U[3]= new Utente("Porto","Rosa", "Batista", 1960);
    U[4]= new Utente("Porto","Telmo", "Correia", 1950);
    for (int P=0; P<U.Length; P++)
```

```
      Console.WriteLine("{0} - {1} tem {2} anos", U[P].PCentro
      ,U[P].NomeCompleto, U[P].Idade());}}}
```

? EXERCÍCIO 10: PESSOAS E BENS

```
using System;
namespace SAI10
{public interface ISumario
 {string Sumario {get;}}

 public class Pessoa: ISumario
 {private string Titulo;
  private string Apelido;
  public Pessoa(string T, string A)
  {Titulo = T;
   Apelido=A;}
  public string Sumario
  {get
   {return Titulo+" "+ Apelido+"\n";}}}

 public class Carro: ISumario
 {private string Marca;
  private string Modelo;
  public Carro(string M, string Mo)
  {Marca = M;
   Modelo = Mo;}
 public string Sumario
  {get
   {return "Carro: "+Marca+" "+ Modelo +"\n";}}}

 public class Telemovel : ISumario
  {private string Marca;
   private int Numero;
  public Telemovel(string M, int N)
  {Marca = M;
   Numero = N;}
  public string Sumario
   {get
    {return "Telemóvel: "+Marca+" "+ Numero+"\n";}}}

 public class PessoasEBens
  {static void Main(string[] args)
   {Pessoa P=new Pessoa("Sr.", "Lima");
    Carro C=new Carro("Opel", "Corsa");
    Telemovel T=new Telemovel("Nokia", 9123456);
     Console.WriteLine(P.Sumario + C.Sumario+T.Sumario);}}}
```

? EXERCÍCIO 11: DUAS MÉDIAS PARA UMA AMOSTRA

```
using System;
namespace SAI11
{interface IEstatisticasI
  {double Media();}

 interface IEstatisticasII
  {double Media();}

 public class Calculos : IEstatisticasI, IEstatisticasII
  {private int Tamanho;
   private double[] Amostra;
   public Calculos(int N, double[] Notas)
    {Tamanho = N;
     Amostra = new double[N];
     Amostra = Notas;}
   double IEstatisticasI.Media ()
    {double Med = 0;
     for (int I = 0; I < Amostra.Length; I++)
      Med += Amostra[I];
   return Med/Tamanho;}
   double IEstatisticasII.Media()
    {double Med = 0;
    int Peso = 0;
    int TPeso=0;
    for (int I = 0; I < Amostra.Length; I++)
     {if (I % 2 == 0)
      Peso = 1;
      else
       Peso = 2;
     TPeso += Peso;
     Med += Amostra[I] * Peso;}
   return Med / TPeso;}}

 public class DuasMedias
  {static void Main(string[] args)
   {double[] Notas = { 12, 15, 16, 13, 12, 10 };
    Calculos C = new Calculos(Notas.Length, Notas);
    IEstatisticasI IT1 = (IEstatisticasI) C;
    Console.WriteLine("Média aritmética={0}", IT1.Media());
    IEstatisticasII IT2 = (IEstatisticasII) C;
    Console.WriteLine("Média ponderada={0}", IT2.Media ());}}}
```

? EXERCÍCIO 12: HIERARQUIA DE INTERFACES PARA ESTATÍSTICAS

```
using System;
namespace SAI12
{interface Estatisticas
  {double Media();}

 interface MaisEstatisticas: Estatisticas
  {double Variancia(double M);
   double DesvioPadrao(double V); }

 public class AnaliseDados: MaisEstatisticas
  {private int Tamanho;
   private double[] Amostra;
   public AnaliseDados(int N, double[] Notas)
    {Tamanho = N;
     Amostra = new double[N];
     Amostra = Notas;}
   public double Media ()
   {double Med = 0;
    for (int I = 0; I < Amostra.Length; I++)
     Med += Amostra[I];
   return Med/Tamanho;}
   public double Variancia(double Med)
    {double Var=0;
    for (int I = 0; I < Amostra.Length; I++)
     Var+= Math.Pow((Amostra[I]-Med),2);
   return Var / (Tamanho - 1);}
   public double DesvioPadrao(double Var)
    {return Math.Pow(Var, 0.5);}}

   public class HierarquiaDeInterfacesParaEstatistica
    {static void Main(string[] args)
     {double[] Notas = { 12, 15, 16, 13, 12, 10 };
      AnaliseDados C = new AnaliseDados(Notas.Length, Notas);
      double Med=C.Media();
      double Var=C.Variancia(Med);
      double DesvPad=C.DesvioPadrao(Var);
      Console.WriteLine("Média={0}", Med);
      Console.WriteLine("Variância={0} e Desvio padrão ={1} ",
      Var, DesvPad);}}}
```