

Programação Visual

Coleções Resumo

1

C# – Coleções

Classes de Colecção

System.Collections.Generic

Listas

List < T >

Conjuntos

HashSet < T >

Tabelas

Dictionary < TKey, TValue >

são os MAP do java

System.Collections.Classes

Não veremos em PV

System.Collections.Concurrent

Não veremos em PV

2

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

2

C# – Coleções

Usamos coleções para armazenar grupos de elementos

Os objetos do tipo coleção devem ter como identificador um nome plural

Uma coleção é uma classe pelo que temos de declarar e criar um instância dessa classe ...

antes de lhe adicionar elementos

Interessam-nos 3 tipos de coleções:

Listas

coleções ordenadas pela ordem de inserção e com possível repetição de elementos. (List<T> , Stack<T>, SortedList<T> ou ArrayList)

Conjuntos

coleções não ordenadas e sem elementos repetidos. (HashSet<T>, SortedSet<T>)

Tabelas

coleções de pares (chave, valor) em que não existem chaves repetidas (Dictionary<TKey, TValue>, SortedDictionary<TKey,TValue>)

Alunos / Turma

List<Aluno> alunos = new List<Aluno>();

alunos.Add(new Aluno () { Nome = "Jose Antunes", Numero = 1234};

HashSet<int> numerosDeAluno = new HashSet<int>();

var alunosDaEST = new Dictionary <int , Aluno>();

3

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

3

C# – Coleções – List<T>

public class Alunos : List<Aluno>

{

override

public String ToString()

{

String str = "";

foreach (Aluno a in this)

{

str += "\n" + a;

}

return str;

}

}

Alunos é uma (herda de) Lista de Aluno.

E assim sendo já tem todos os métodos que tem uma List

4

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

4

C# – Coleções – HashSet<T>

public class NumerosDeAlunos : HashSet<String>

{

public NumerosDeAlunos (Unidade unidade, int ano, int qtdAlunos)

{

for (int i = 0; i < qtdAlunos; i++)

this.Add(ano.ToString().Substring(2) + "\0" + (int)unidade + i.ToString("0000"));

}

override public String ToString()

{

String str = "";

foreach (String s in this)

{

str += s + "\n";

}

return str;

}

NumerosDeAlunos é um (herda de) HashSet de Strings

O construtor com 3 parâmetros adiciona qtdAlunos strings no formato AA0UUU#### A si próprio (this)

O ToString retorna uma String com todos os elementos do conjunto (this)

5

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

5

C# –Iterar um HashSet<T>

static void Main(string[] args)

{

NumerosDeAlunos numeros = new NumerosDeAlunos (Unidade. EST_SETUBAL, 2019, 3);

Enumerator é um iterador para uma coleção

GetEnumerator criar e retorna um Enumerator sobre a coleção numeros

NumerosDeAlunos.Enumerator it = numeros.GetEnumerator();

it.Current retorna o valor atualmente apontado pelo Enumerator

it.MoveNext tenta mover para o elemento seguinte. Se existir um seguinte move e retorna true caso contrário retorna false

string val = "";

do {

val = it.Current;

}

while (!val.Equals(190210001) && it.MoveNext());

Console.WriteLine(val);

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.hashset-1?view=netframework-4.8>

}

6

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

6

CTESP TPSI 2017/18
José Braz (EST Setúbal/DSI)

C# – Coleções – SetAlunos é um HashSet<Aluno>

```
public class SetAlunos : HashSet<Aluno> {
    // Como SetAlunos é um HashSet<Aluno> tem já todos os métodos de HashSet<T>
    // 1. Mas herda os Equals e GetHashCode de Object para os quais dois
    // objetos do tipo Aluno são iguais se referenciarem o mesmo endereço
    // 2. Por isso teremos de redefinir os Equals e o GetHashCode
    // da classe Aluno por forma a que dois alunos sejam iguais
    // de acordo com os nossos critérios: dois alunos serão
    // iguais se tiverem o mesmo número de aluno (ver classe Aluno)
    // restante código omitido }

    public class Aluno {
        public String Numero {get; set;}
        public String Nome {get; set;}

        // restante código omitidos

        public override int GetHashCode()
        {
            return Numero.GetHashCode();
        }

        public override bool Equals(object obj) {
            // Is null
            if (object.ReferenceEquals(null, obj)){
                return false;
            } // Is the same object
            if (object.ReferenceEquals(this, obj)){
                return true;
            } // Is not the same type
            if (obj.GetType() != this.GetType()){
                return false;
            }
            Aluno a = obj as Aluno;
            return String.Equals(Numero, a.Numero);
        }
    }
}
```

7

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

7

C# – Coleções – Dictionary<TKey, TValue>

```
namespace TP04_Collections
{
    enum Lugares { FAL1, FAL2, FAL3, FAL4,
        FBL1, FBL2, FBL3, FBL4, FBL5, FBL6 }

    class Sala : Dictionary<Lugares, Aluno>
    {
        public override string ToString()
        {
            String str = "";
            foreach (KeyValuePair<Lugares, Aluno> a in this)
                str += a.Key.ToString() + "- " +
                    a.Value.ToString() + "\n";
            return str;
        }
    }
}
```

Sala é um (herda de) Dictionary em que as chaves são Lugares e os valores Alunos

Leia-se: para cada par (chave, valor) a

8

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

8

C# – Coleções – Dictionary<TKey, TValue>

```
namespace TP04_Collections
{
    enum Lugares { F1LA, F1LB, F1LC, F1LD,
        F2LA, F2LB, F2LC, F2LD, F2LE, F2LF }

    class Sala : Dictionary<Lugares, Aluno>
    {
        public override string ToString()
        {
            String str = "";
            foreach (var a in this)
                str += a.Key.ToString() + "- " +
                    a.Value.ToString() + "\n";
            return str;
        }
    }
}
```

Podemos simplificar e usar o "Tipo Implícito" var

9

Programação Visual

TeSP TPSI

José Braz (ESTSetúbal / DSI)

out-25

9