using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace ConsoleApp7

{

class Class1

{

static void Main()

{

int[] num = new int[10];

num[0] = 100;

num[2] = 200;

for(int i=0;i<num.Length;i++)

{

num[i] = 100;

Console.WriteLine(num[i]);

}

foreach(int temp in num)

{

// temp = 100;

Console.WriteLine(temp);

}

Console.WriteLine("ArrayList");

ArrayList list = new ArrayList();

list.Add(10);

list.Add(20);

list.Add(30);

foreach(int temp in list)

Console.WriteLine(temp);

list.Insert(1, 1000);

list.RemoveAt(0);

Console.WriteLine("Stack");

Stack stack = new Stack();

stack.Push(1);

stack.Push(2);

stack.Push(3);

stack.Pop();

foreach (int temp in stack)

Console.WriteLine(temp);

Console.WriteLine("Queue");

Queue queue = new Queue();

queue.Enqueue(1);

queue.Enqueue(2);

queue.Dequeue();

foreach (int temp in queue)

Console.WriteLine(temp);

}

}

}

**Limitations of Arrays>**

1. Static Memory Allocation , Memory requirement shud be known in beginning

2. Memory can be wasted

Int[] nu = new int[100];

3. Insertion / Deletion are time consuming because it requires lots of reshuffling of elements

**Advantages of Arrays:** Store multiple elements in a single structure

Elements are of same type

**Limitations of Collection>**

**Elements are not of same type**

**Advantage of collection > Dyamic . Easy to insert , delete elements**

**Generic Collections**

**Typed Colletcions**

Elements are of same type

**Dyamic . Easy to insert , delete elements**

Memory is not wasted

ArrayList List<type>

Stack Stack<type>

Queue Queue<type>

HashSet Dictionary<type of key, type of value>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace ConsoleApp7

{

class Class1

{

static void Main()

{

int[] num = new int[10];

num[0] = 100;

num[2] = 200;

for(int i=0;i<num.Length;i++)

{

num[i] = 100;

Console.WriteLine(num[i]);

}

foreach(int temp in num)

{

// temp = 100;

Console.WriteLine(temp);

}

Console.WriteLine("ArrayList");

ArrayList list = new ArrayList();

list.Add(10);

list.Add(20);

list.Add(30);

list.Add("Ajay");

list.Add(12.4f);

list.Add(11); // Boxing means converting value to reference type

foreach(var temp in list) // Unboxing means converting ref to value type

Console.WriteLine(temp);

list.Insert(1, 1000);

list.RemoveAt(0);

Console.WriteLine("Stack");

Stack stack = new Stack();

stack.Push(1);

stack.Push(2);

stack.Push(3);

stack.Pop();

foreach (int temp in stack)

Console.WriteLine(temp);

Console.WriteLine("Queue");

Queue queue = new Queue();

queue.Enqueue(1);

queue.Enqueue(2);

queue.Dequeue();

foreach (int temp in queue)

Console.WriteLine(temp);

List<int> list1 = new List<int>();

list1.Add(1);

list1.Add(2);

List<string> list2 = new List<string>();

list2.Add("aaa");

foreach(int temp in list1)

{

Console.WriteLine(temp);

}

foreach (string temp in list2)

{

Console.WriteLine(temp);

}

Stack<int> stack1 = new Stack<int>();

Console.WriteLine("Enter Elemment to search");

int x = int.Parse(Console.ReadLine());

foreach(var temp in list)

{

if (x == (int)temp) { }

}

// Serching in ArrayList, List, Stack is sequential

// Searching in Hashtable is random

Hashtable hs = new Hashtable();

hs[1] = 19;

hs[2] = 90;

Console.WriteLine(hs[2]);

hs["ajay"] = 90;

Dictionary<int, int> marks = new Dictionary<int, int>();

marks[1] = 90;

Dictionary<string, int> markList = new Dictionary<string, int>();

markList["ajay"] = 90;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

class Student

{

int rn;

String name;

static string batch;

// Def const (Cud be only 1)

public Student() { }

// Para

// Para cons cud be more than 1

public Student(int rn, string name)

{

this.rn = rn;

this.name = name;

}

public Student(int rn )

{

this.rn = rn;

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

// static con (Can be only one)

static Student()

{

batch ="B001";

}

public void get() {

Console.WriteLine("Enter RollNo");

rn = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void show() {

Console.WriteLine("RollNo is " + rn);

Console.WriteLine("Name is " + name);

}

}

class Progarm2

{

static void Main()

{

Student student1 = new Student();

student1.get();

student1.show();

Student student2 = new Student(2);

Student student3 = new Student(3,"Deepak");

student2.show();

student3.show();

}

}

}

Exception Handling

Error : Something goes wrong

1. Syntax Error / detected at compile time CAN BE CORRECTED
2. Logical Error / In this case , we always get response, but that is not correct

It is detected at run time/ Difficult to debug CAN BE CORRECTED

1. Run time Error > These are the exceptions which come sometimes

It depends upon what user enters at that time CAN NOT BE CORRECTED

We cannot correct them, but we can handle them

Enter Name :

Try catch finally block

Try > will contain statements which can cause exceptions

Catch > which will handle the exception

Finally > is always executed

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

class Class3

{static void Main()

{

int x, y, res = 0;

try

{

Console.WriteLine("Enter x");

x = int.Parse(Console.ReadLine());

Console.WriteLine("Enter y");

y = int.Parse(Console.ReadLine());

res = x / y;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Console.WriteLine("Res is " + res);

}

}

}

}

Exception Handling

Abstract Class

Interface

Method Overriding