**Advantage** :

* All the elements share common name
* We can access elements thru their position(index)
* The elements are stored in a contiguous location
* No need to declare different elements
* Similar data type, arrays are type-safe

**Disadvantages of Arrays**

* Size is fixed , memory is static

Int[] num = new int[20];

* Insertion and deletion of elements is time consuming because it requires lots of reshuffling of elements

Collection

Advantages

* Size is dynamic , memory is dynamic which means memory is not wasted
* Insertion and deletion of elements is easy. We can call inbuilt methods to do these operations

ArrayList list = new ArrayList();

List.Add(10);

Disadvantages of Collections

1. Elements are not of same type , Collections are not type-safe

// int[] num = new int[10];

ArrayList list = new ArrayList();

list.Add(1); // Boxing is done

list.Add(2);

list.Add("Deepak");

list.Add(100);

list.Add(1.9);

Console.WriteLine("Elements are ");

foreach(var temp in list) // Unboxing is done

{

Console.WriteLine(temp);

}

Int x;

String name;

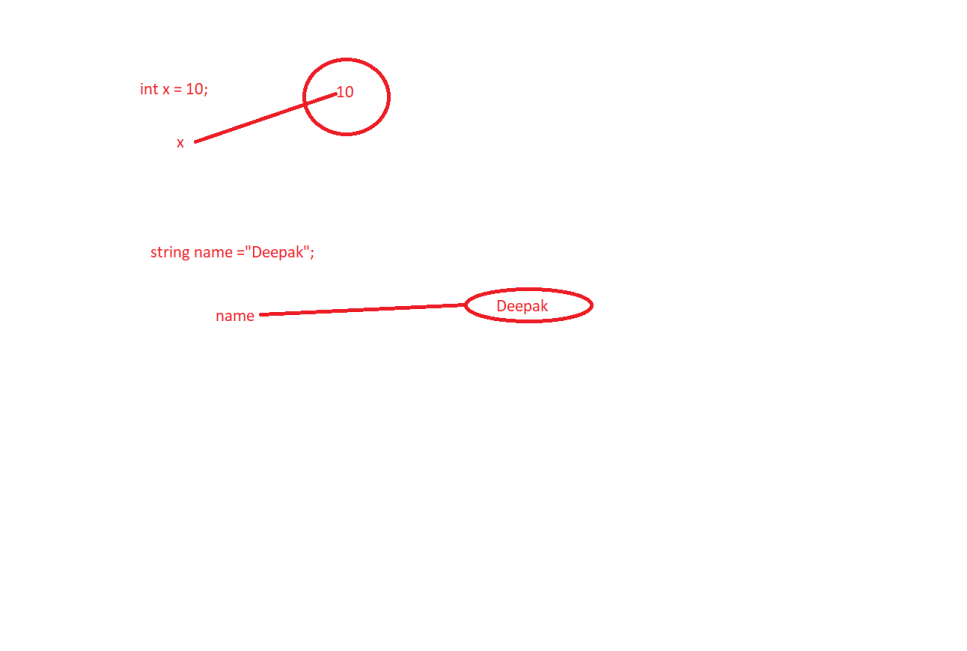
In RAM, we have 2 structures > Stack Heap

Variables of how many types

* Value Type : which stores the value . int , float , Boolean, char, struct, enum
* Reference Type : which stores the reference of the variable , string , class

When we declare varibale , they get memory and are stored somewhere in memory

Value type varibales are stored in stack

Reference type variables are stored in heap

Value type variables are stored in stack Reference type variables are stored in heap

Int x= 40

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| + |
| 40 |
| 10  Deepak |
| 90 |

String name = “Deepak”

Garbage Collection : Removing unused memory blocks

**Boxing > Converting value type variable to reference type**

**Unboxing > Converting reference type variable to value type**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class BoxingUnboxingDemo

{

static void Main()

{

int x = 20;

// Boxing

Object obj = x;

// Unboxing

int y = (int)obj;

}

}

}

**for vs foreach**

for loop

int[] num = new int[10];

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

{

Console.Write(num[i]);

}

// foreach

int[] num = new int[10];

foreach(int temp in num)

{

Console.Write(temp);

}

Out of these 2 loops, for loop is faster

* Because in foreach , it may require boxing unboxing sometimes
* Purpose of foreach loop is to only get information , it does not allow to change something in the loop

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class BoxingUnboxingDemo

{

static void Main()

{

int x = 20;

// Boxing

Object obj = x;

// Unboxing

int y = (int)obj;

int[] num = new int[] { 1, 2, 3, 4 };

Console.WriteLine("Using For loop");

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

{

num[i]++;

Console.WriteLine(num[i]);

}

Console.WriteLine("Using Foreach loop");

foreach (int temp in num)

{

// temp++; // It will give compile time error

Console.WriteLine(temp);

}

}

}

}

using System;

using System.Collections;

class Program

{

static void Main()

{

ArrayList list = new ArrayList();

list.Add(1); // Boxing happens here

list.Add(12);

list.Add(2);

list.Add(3);

foreach(int x in list)

Console.WriteLine(x);

list.Insert(1, 200);

list.RemoveAt(1);

list.Remove(100);

list.Add("Ajay");

list.Add(12.8f);

foreach (var x in list)

Console.WriteLine(x);

// LIFO

Stack stack = new Stack();

stack.Push(1);

stack.Push(2);

stack.Push(3);

stack.Push("Ajay");

foreach (var x in stack)

Console.WriteLine(x);

stack.Pop();

// FIFO

Queue queue = new Queue();

queue.Enqueue(1);

queue.Enqueue(2);

queue.Enqueue(3);

queue.Enqueue("Ajay");

queue.Dequeue();

foreach (var x in queue)

Console.WriteLine(x);

Hashtable hs = new Hashtable();

hs[1] = 20;

hs[2] = 19;

hs[3] = 21;

foreach(int x in hs.Keys)

Console.WriteLine(hs[x]);

hs["Deepak"] = "Delhi";

}

}

**We take advantages of Arrays & Collections**

**advantages of Arrays** Arrays are type-safe, elements are of same type

**advantages of Collections** Collections are dynamic , insertion/ deletion is easier

When we combine both of their advantages , **We get GENERIC COLLECTIONS**

Generic collections are collection only but they are type-safe , which means they can store elements of same type

**Classes for generic collections are present in System.Collections.Generic;**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class GenericCollectionsDemo

{

static void Main()

{

// ArrayList list = new ArrayList();

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

list.Add(1);

list.Add(2);

list.Add(100);

foreach(int temp in list)

Console.WriteLine(temp);

Queue<string> queue = new Queue<string>();

queue.Enqueue("Ajay");

queue.Enqueue("Deepak");

foreach (string temp in queue)

Console.WriteLine(temp);

Stack<float> stack = new Stack<float>();

stack.Push(10.9f);

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

marks["Ajay"] = 90;

marks["Deeapk"] = 89;

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

class Program

{

static void Main()

{

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

list.Add(1);

//list.Add("aa");

foreach(int x in list)

Console.WriteLine(x);

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

stack.Push("Ajay");

foreach (string x in list)

Console.WriteLine(x);

Queue<float> queue = new Queue<float>();

queue.Enqueue(198.5f);

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

records[1] = 10;

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

record[1] = "Ajay";

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

void DisplayDetails()

{

Console.WriteLine("RollNo is {0}" , rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class ClassDemo

{

static void Main()

{

// Value type variables take default values when we declare them

int x;

bool b;

// Reference type variables get by defualt null

Student student;

// Right now, this variable/object

// has not been assigned any memory from heap

// It gets allocated memory when we use

// new keyword

student = new Student();

//Student student;

//student = new Student();

Student student1 = new Student();

}

}

}

By default, all class members are private, which means they can not be accessed outside the class. If you want to access them outside the class , we have to use public access specifier

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is {0}" , rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class ClassDemo

{

static void Main()

{

// Value type variables take default values when we declare them

int x;

bool b;

// Reference type variables get by defualt null

Student student;

// Right now, this variable/object

// has not been assigned any memory from heap

// It gets allocated memory when we use

// new keyword

student = new Student();

student.GetDetails();

student.DisplayDetails();

//Student student;

//student = new Student();

Student student1 = new Student();

}

}

}

Store & Display records for 2 objects

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is {0}" , rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class ClassDemo

{

static void Main()

{

Student student1 = new Student();

student1.GetDetails();

student1.DisplayDetails();

Student student2 = new Student();

student2.GetDetails();

student2.DisplayDetails();

}

}

}

Array of Objects

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is {0}", rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class StudentClassDemo

{

static void Main()

{

// Array of Objects

Student[] students = new Student[10];

for (int i = 0; i < 2; i++)

{

Console.WriteLine("Enter Details for Student No {0} ", i + 1);

students[i] = new Student();

students[i].GetDetails();

}

for (int i = 0; i < 2; i++)

{

Console.WriteLine("Details for Student No {0} ", i + 1);

students[i].DisplayDetails();

}

}

}

}

Collection of Students

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ThirdProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is {0}", rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class StudentCollectionClassDemo

{

static void Main()

{

// Collection Objects

ArrayList students = new ArrayList();

for (int i = 0; i < 2; i++)

{

Student student = new Student();

Console.WriteLine("Enter Details for Student");

student.GetDetails();

students.Add(student);

}

foreach(Student student in students)

{

student.DisplayDetails();

}

}

}

}

Generic Collection

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ThirdProject

{

class Student

{

int rn;

string name;

string batch;

int marks;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Batch");

batch = Console.ReadLine();

Console.WriteLine("Enter Marks");

marks = Byte.Parse(Console.ReadLine());

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is {0}", rn);

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

Console.WriteLine("Batch is " + batch);

Console.WriteLine("Marks are " + marks);

}

}

class StudentCollectionClassDemo

{

static void Main()

{

// Collection Objects

List<Student> students = new List<Student>();

for (int i = 0; i < 2; i++)

{

Student student = new Student();

Console.WriteLine("Enter Details for Student");

student.GetDetails();

students.Add(student);

}

**// students.Add(1);**

foreach(Student student in students)

{

student.DisplayDetails();

}

}

}

}

**Class access specifier is internal, which means it is accessible in the entire project**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class StudentCollectionClassDemo

{

static void Main()

{

// Collection Objects

List<Student> students = new List<Student>();

for (int i = 0; i < 2; i++)

{

Student student = new Student();

Console.WriteLine("Enter Details for Student");

student.GetDetails();

students.Add(student);

}

// students.Add(1);

foreach(Student student in students)

{

student.DisplayDetails();

}

}

}

}