Muhammad Hamza Khan

Assignment: Class Assignment

EP#1750044

BSSE-1 Section A

Seat # 22

Submitted to: Miss Shaista Raees

Object Oriented Programming

Q1. Briefly define "stack" collection. How it is implemented in C#

STACK:

Stack is the process of putting data on top of other data. This can be visualized very easily by thinking that your data is a stack of plates or book which are situated on top of each other. In this example the top item of the stack can be taken first and the bottom most item can only be accessed last. Just like this example "stack" represents **"First in Last out (FiLo)".**

Three Primary Operators of Stack:

-Push() – Put new Data on the top of the stack.

-Pop() – Remove data from the top of the stack.

-Peek() – Get a copy of the data on the top of the stack.

Example of the stack where it can be used:

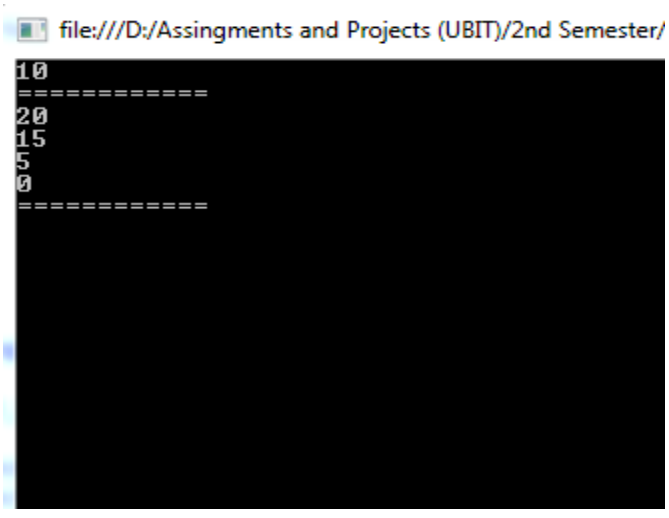
They are very useful in games as in they can be used in "stack of cards or pawns in chess or loot items etc."

IMPLEMENTATION:

Code example of stack is given below:

```
namespace Stack_Example
{
    class Program
    {
        static void Main(string[] args)
        {
            Stack<int> Default_Stack = new Stack<int>();

            Default_Stack.Push(0);
            Default_Stack.Push(5);
            Default_Stack.Push(10);
            int topOfStack = Default_Stack.Pop();
            Console.WriteLine(topOfStack);
            Default_Stack.Push(15);
            Default_Stack.Push(20);
            Console.WriteLine("============");
            foreach (int stackPrint in Default_Stack)
            {
                Console.WriteLine(stackPrint);
            }
            Console.WriteLine("============");
            Console.ReadKey();
        }
    }
}
```



Q2. Describe Operator Overloading in C#.

OPERATOR OVERLOADING:

The process of overloading predefined operators in C# is called operator overloading. It is a type of polymorphism most closely related to static polymorphism. C# allows us to overload most of the predefined operators. Therefore, a computer programmer can use operators with user defined meaning.

Following Operators can be overloaded:

1. Unary Operators:

+,-,!,~,++,--

2. Binary Operators:

+,-,*,/,%

3. Compression Operators:

==, !=,<,>,<=,>=

Following Operators cannot be overloaded:

1. Conditional Logical Operators:

&&, ||

2. Assignment Operators:

+=, -=, *=, /=, %=

3. These Operators:

=, ., ?:, ->, new, is, sizeof, typeof

Code Example:

```
namespace Object_Overloading
{
    class Program
    {
        static void Main(string[] args)
        {
            OverloadingOfObject obj1 = new OverloadingOfObject();
            obj1.Num1 = 50;
            obj1.Str1 = "Hamza";
            OverloadingOfObject obj2 = new OverloadingOfObject();
            obj2.Num1 = 20;
            obj2.Str1 = "Khan";
            OverloadingOfObject obj3 = new OverloadingOfObject();
            obj3 = obj1 + obj2;
            Console.WriteLine(obj3);
            Console.ReadKey();
        }
    }
    class OverloadingOfObject
    {
        public int Num1 = 0;
        public string Str1 = "";
        public static OverloadingOfObject operator +(OverloadingOfObject obj1, OverloadingOfObject
obj2)
        {
            OverloadingOfObject obj3 = new OverloadingOfObject();
            obj3.Num1 = obj1.Num1 + obj2.Num1;
            obj3.Str1 = obj1.Str1 + obj2.Str1;
            return obj3;
        }
    }
}
```

```
70
Hamza Khan
```

Q3. Differentiate between Value and Reference Types.

	VALUE TYPE	REFERENCE TYPE
1	Variable of value types are stored in stacks.	Variables with reference types are stored in heap.
2	Values are stored directly in variable values. No memory address is needed.	Values are stored via memory address.
3	Value types can be created during compile time and then stored in stack memory because of this, Garbage collectors can't collect them.	They are stored in heap because of this they can be marked for garbage collection.
4	Code Example:	Code Example:

	`namespace ValueType`
	`{`
	`  class Program`
	`  {`
	`  static void Main(string[] args)`
	`  {`
	`  int valueType = 30;`
	`  valueType = valueType + DateTime.Today.Day;`
	
	`Console.WriteLine(valueType);`
	`  Console.ReadKey();`
	`  }`
	`  }`
	`}`
		`namespace Reference_Type`
		`{`
		`  class Program`
		`  {`
		`  public static void Operation(ref int Num)`
		`  {`
		`  Num = Num * Num * Num / Num;`
		`  }`
		`  static void Main(string[] args)`
		`  {`
		`  int originalNum = 20;`
		
		`Console.WriteLine("Original Value>{0}", originalNum);`
		`  Program.Operation(ref originalNum);`
		
		`Console.WriteLine("Reference value>{0}", originalNum);`
		`  Console.ReadLine();`
		`  }`
		`  }`
		`}`

Q4. Differentiate between string and System.String.

	System.String	string
1	String is used to access the methods of String Library.	string is a data type which is used for declarations.
2	String methods can only be used if we add the String Library in our program.	No library is needed to declare something with string.
3	String is a class name.	string is a data type meaning it can't be used as a variable.
4	Code Example:	Code Example:

	`namespace System_String`
	`{`
	`  class Program`
	`  {`
	`  static void Main(string[] args)`
	`  {`
	`  string str = "My";`
	`  str += " name";`
	`  str += " is";`
	`  str += " Hamza";`
	
	`Console.WriteLine(str);`
	`  Console.ReadKey();`
	`  }`
	`  }`
	`}`
		`namespace ConsoleApplication4`
		`{`
		`  class Program`
		`  {`
		`  static void Main(string[] args)`
		`  {`
		`  StringBuilder newString = new StringBuilder("");`
		`  newString.Append("My");`
		`  newString.Append(" name");`
		`  newString.Append(" is");`
		`  newString.Append(" Hamza");`
		`  string str = newString.ToString();`
		`  Console.WriteLine(str);`
		`  Console.ReadKey();`
		`  }`
		`  }`
		`}`