Government Polytechnic College, Jodhpur
Department of Computer Science (NBA Accredited)

Programme: Diploma **Class Test**: II **Session**: 2017-18
Course: Dot Net Technology **Year**: IIIrd
Course CODE: CS-305 **Time:** 14:45 to 15:45
Max.Marks : 15 **Date:** 24-01-2018
Instructions to candidates: Attempt Any Three Questions

SI#	Question	Marks	CO MAPPING
1	Explain parameter passing in VB.net.	5	CO3
2	Explain Object Oriented Programming features of vb.net.	5	CO4
3	What are Abstract Base Classes? Explain with example.	5	CO4
4.	Explain array in vb.net with the help of example.	5	CO3

Solution:

Q1. Explain parameter passing in VB.net.

Sol: Code in a procedure needs some information about the state of the program to do its job. These are passed to the procedure as an argument. The parameter are passed by value or by reference to the program.

(i) Pass by value:

This is the default mode for passing arguments in .NET. In this only one copy of the arguments is passed to the procedure. Any changes in the value caused by procedure never affect the original variable, but only affect the copy. We can also write explicitly By Val keyword, otherwise automatically compiler inserts this keyword.

(ii) Pass by reference:

In this, the address of the argument is passed to the procedure and procedure can change its value permanently. Since the copy is not passed, the procedure can have the full access to the arguments. ByRef keyword can be provided before the name of the argument. Example:

```
Sub CostPlusInterest (ByRef Cost As Single, ByRef Total As Single)
Cost = Cost * 1.05
Total = Int(Cost)
End Sub
..
..
Dim Price, TotalPrice As Single
Price = 100
TotalPrice = 0
CostPlusInterest (Price, TotalPrice)
MsgBox (Price & “ at 5% Interest is “ & TotalPrice)
```

Q2. Explain Object Oriented Programming features of vb.net.

Sol: Visual Basic is object-Based, which means it is a Object-Oriented Programming Language. Visual Basic .Net supports all features of OOP like Abstraction, Encapsulation, Polymorphism and Inheritance.

Data Abstraction

Data Abstraction increases the power of programming language by creating user defined data types. Data Abstraction also represents the needed information in the program without presenting the details. Classes use the concept of abstraction and are defined as a list of abstract attributes.

Encapsulation

Data Encapsulation combines data and functions into a single unit called Class. When using Data Encapsulation, data is not accessed directly; it is only accessible through the functions present inside the class. Data Encapsulation enables the important concept of data hiding possible.

Polymorphism

Poly means many and morph means form. Thus, polymorphism refers to being able to use many forms of a type without regard to the details. It is the characteristic of being able to assign a different meaning specifically, to allow an entity such as a variable, a function, or an object to have more than one form. In Polymorphism objects behaves differently depending on their data types.

Inheritance

Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class, The new class that is formed is called derived class. Derived class is also known as a child class or sub class. Inheritance helps in reducing the overall code size of the program, which is an important concept in object-oriented programming.

Q3. What are Abstract Base Classes? Explain with example.

Sol: VB.NET enables us to create an abstract class using MustInherit keyword. Abstract class can contain the signature of the methods that the derived class can implement in its own class.

In this case, we cannot create the objects of those classes. The methods with the keyword mustoverride should not contain any implementation. The class with even one mustoverride method should be declare as mustinherit. To implement these methods, we have to use overrides keyword in the subclass. The subclass is not implementing any one of the abstract methods from the abstract class, then the child class must be declared as abstract one. Abstract class can contain non abstract methods also.

Example:
```
Public MustInherit Class test
  Public MustOverride Sub totalsal (ByVal basic As Integer)
 Public Sub disp()
        Msgbox (“from abstract class”)
 End Sub
End Class
Public Class second
        Inherits test
 Public Overrides Sub totalsal( ByVal basic As Integer)
        Dim total As Integer
        Total = basic + (basic * 0.4)
        MsgBox(“total salary is “ & total)
End Sub
 End Class
```

Accessing the class:

```
 Dim second As New second()
 Second.totalsal (4500)
Second.disp()
```

The third statement calls the disp method from the abstract class.

Q4. Explain array in vb.net with the help of example.

Sol: An array can hold a collection of values of the same type. The variables in an array are called elements. Array has a name and these elements can be accessed by using index or subscript.

An array can have single, double and multi dimentions. Lower bound of the array for all the dimention start from 0. Upper bound can be defined by using the code. Array is a reference type and can be static or dynamic array. The size is of the dynamic array changes during the execution of the code.

Static Arrays:

Arrays can be declared using Dim keyword. You can initialize the values in the array. If array is declared as object type, we can store different types of datatypes also.

```
Dim a(10) as Integer
Dim a(2,3) as String
Dim b() as String = {12.3, 78.9}
Dim (3) as Object
```

Dynamic Array:

If you don’t know the size of the array, this would be very helpful for storing the data. The Redim and Preserve statements are used for this purpose. For Dynamic array we should not provide the size when we declare the array. After the declaration we cannot change the datatype using redim statement. Preserve keyword is used to preserve the data without losing when changing the dimention of the array. Using Redim Preserve Keyword, only last dimention can be resized.

```
Dim a() as String
Redim a(5)
Dim b(,) as Integer
Redim b(3,2)
Redim Preserve b(3,4)
```