Object Oriented programming in C#:

Object Oriented programming is a programming style which is associated with the concepts like class, object, Inheritance, Encapsulation, Abstraction, Polymorphism.

Class: A class is a collection of method and variables.

We can define a class using the class keyword and the class body enclosed by a pair of curly braces, as shown in the following example:

```
public class A
{

}
```

Inheritance: Inheritance feature allows code reusability when a class includes property of another class. In C# ":" is used to represent inheritance, as shown in the following example:

```
public class Papa {

}

class Child : Papa {

}
```

Objects: Object is a component of a program that knows how to perform certain actions and how to interact with other elements of the program, as shown in the following example:

```
Rectangle Rect = new Rectangle();
```

Here, Rectangle is a class, Rect is an object of class Rectangle, new is a keyword, Rectangle() is a constructor of class Rectangle.

Abstraction: Abstraction can be achieved using abstract classes in C#. Abstract classes contain abstract methods, which are implemented by the derived class.

Encapsulation: Encapsulation means bundling of data and methods within one unit, e.g., a class in C#. Encapsulation enables a programmer to implement the desired level of abstraction.

Polymorphism: Polymorphism means overloading and overriding, as shown in the following example:

Polymorphism overloading example:

```
using System;
namespace PolyOverloadingExample
{
 public class OverLoading
 {
```

```
  public void sum(int a, int b)
  {
    Console.WriteLine(a + b);
  }
  public void sum(int a, int b, int c)
  {
    Console.WriteLine(a + b + c );
  }
 }
}
```

Polymorphism overriding example:

```
using System;
namespace PolyOverridingExample
{
 public class Parent
 {
  public virtual void Show()
  {
    Console.WriteLine("Welcome");
  }
 }

 public class Child:Parent
 {
  public override void Show()
  {
    Console.WriteLine("Swagatam");
  }
 }
}
```

Viva Vice on OOP in C#

Q1. The capability of an object in Csharp to take number of different forms and hence display

behaviour as according is known as:-

Ans. Polymorphism.

Q2. Which of the following keyword is used to change data and behavior of a base class by replacing a member of the base class with a new derived member?

Ans. New.

Q3. What is the correct way to overload +operator?

Ans. public sample operator + (sample a, sample b), public abstract operator + (sample a,sample b).

Q4. Selecting appropriate method out of number of overloaded methods by matching arguments in terms of number,type and order and binding that selected method to object at compile time is called?

Ans. Compile time polymorphism.

Q5. Correct syntax for while statement is:-

Ans.

```
while(condition)
{
}
```

References:

1. Sebesta,"Concept of programming Language", Pearson Edu
2. Louden, "Programming Languages: Principles & Practices" , Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms ", Tata McGraw –Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in Java | PPL
18. PPL: Abstraction & Encapsulation
19. PPL: Semaphores
20. PPL: Introduction to 4GL
21. PPL: Variable Initialization
22. PPL: Conditional Statements
23. PPL: Array
24. PPL: Strong Typing
25. PPL: Coroutines

26. PPL: Exception Handler in C++
27. PPL: OOP in PHP
28. PPL: Character Data Type
29. PPL: Exceptions
30. PPL: Heap based storage management
31. PPL: Primitive Data Type
32. PPL: Data types
33. Programming Environments | PPL
34. Virtual Machine | PPL
35. PPL: Local referencing environments
36. Generic Subprograms
37. Local referencing environments | PPL | Prof. Jayesh Umre
38. Generic Subprograms | PPL | Prof. Jayesh Umre
39. PPL: Java Threads
40. PPL: Loops
41. PPL: Exception Handling
42. PPL: C# Threads
43. Pointer & Reference Type | PPL
44. Scope and lifetime of variable
45. Design issues for functions
46. Parameter passing methods
47. Fundamentals of sub-programs
48. Subprograms
49. Design issues of subprogram
50. Garbage Collection
51. Issues in Language Translation
52. PPL Previous years solved papers

53. Type Checking | PPL | Prof. Jayesh Umre
54. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
55. PPL Viva Voce
56. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
57. Concurrency
58. Basic elements of Prolog
59. Introduction and overview of Logic programming
60. Application of Logic programming
61. PPL: Influences on Language Design
62. Language Evaluation Criteria PPL
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++