S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

OOPS

By. Kannababu

OOPS: - Object Oriented Programming System

OOPS is a concept which is used to write programs by using classes

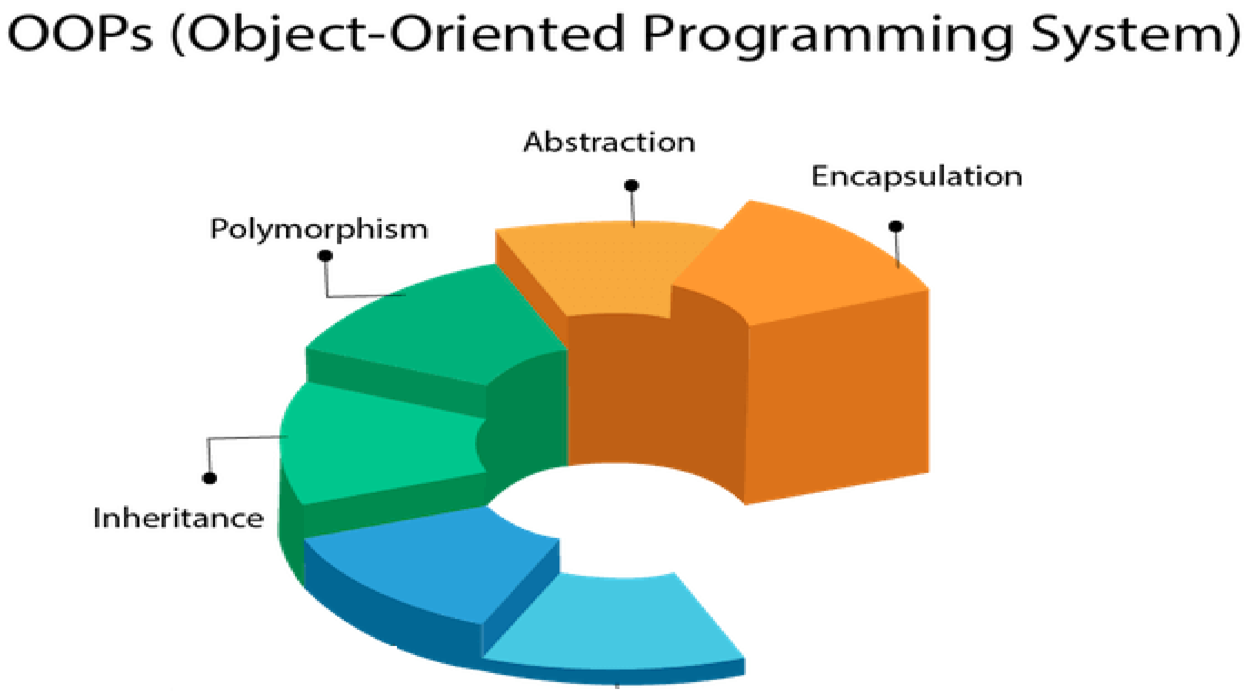
Principles of OOPS:-

1. Abstraction

2. Encapsulation

3. Inheritance

4. Polymorphism



Abstraction:- Data **abstraction** is the process of hiding certain details and showing only essential information

Encapsulation:-it is a process of Binding variables and Methods in a single container

In Object Oriented Programming Languages we can achieve Encapsulation by using class

Inheritance:- it is a mechanism of establishing the relationship between classes

In C#.net we can achive inheritance by using :keyword

Polymorphism:- Polymorphism means anything that exist in many forms

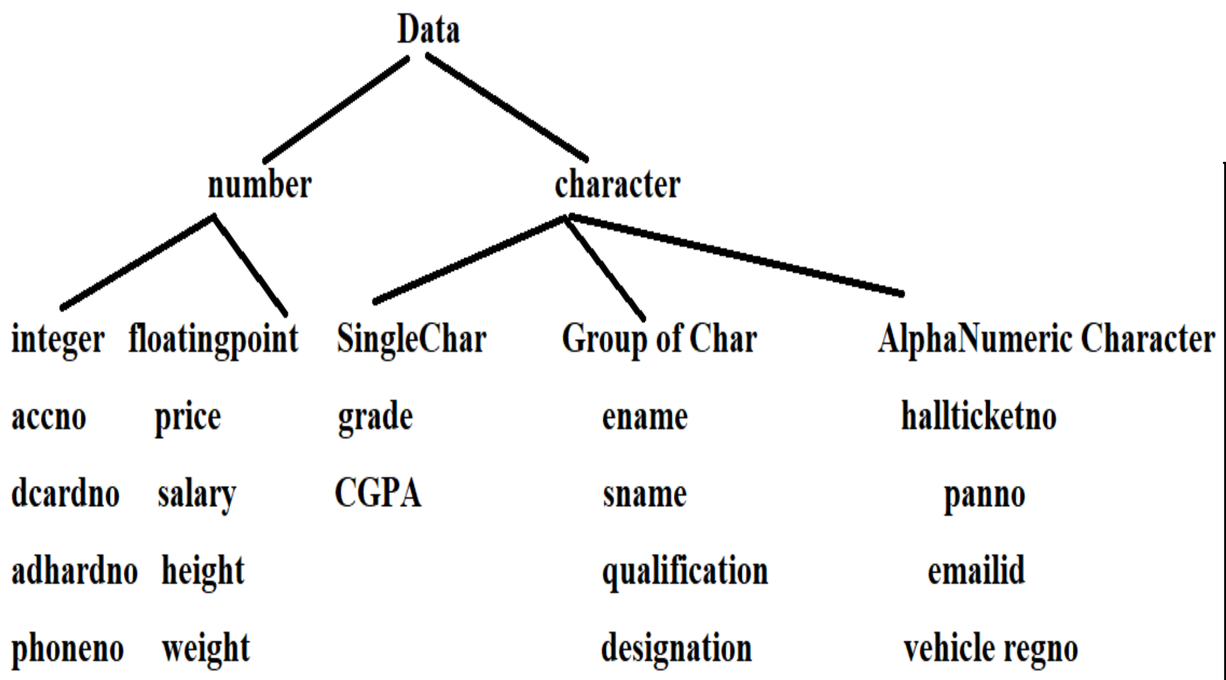
In C#.net we can achive **Polymorphism** by using OverLoading and OverRiding

Advantages of OOPS:-

1. DataOrganization
2. DataSecurity
3. Reusability
4. Extensability
5. ReImplementation
6. Code Maintenance
7. Design Benefits

Q) What is Data?

Data is collection of rawfacts



Q)what is integer?

a number without decimal point is called as integer

in C#.net we can represent integer values by using

byte,short,int,long

Q)what is floating point?

a number with decimal point is called as floatingpoint

in C#.net we can represent floatingpoint values by using

float,double

Q)what is character ?

character is an alphabet or symbol or digit that was

declared within single quotes

char grade='A';

char x='@';

Q)what is string ?

string is group of characters

string value can be declared within " "

```
string   ename="kannababu";
string   designation="softwaredeveloper";
string   emailid="kannabanna1022@gmail.com";
string   companyname="sathyatechnologies";
string   panno="BAG9267R";
```

Q)what is Datatype?

Datatype specifies the type of data that we store in memory

i.e how much memory was required was decided by datatype

Q)what is variable?

variable is the name given for a particular memory location

Q)what is the pupose of variable?

the pupose of variable is to store the data

Q)what is Method?

Method is a subprogram which is used to perform a specific operation

i.e a specific task

identify the variables and methods

Requirement:- Developing an appn for an engineering college to Maintain student,employee,course,dept details?

Sno	hra	percentage	SetCourse
sname	tsal	grade	SetCredits
m1	experience	bloodgroup	SetCourseDuration
m2	dateofjoin	courseduration	DisplayCourse
m3	offerdate	SetStudent	SetDept
phoneno	pfno	GetStudent	DisplayDept
emailid	deptno	SetEmpData	DisplayCredits
address	dname	CalculateTotal	GetHodinfo
gender	hodname	GetEmpData	UpdateDept
qualification	deptphno	CalculatePercentage	DeleteEmp
ename	coursed	CalGrade	AddStudent
eno	coursename	DisplayTotalMarks	DeleteCourse
designation	credits	CalculateDa	UpdateCourse
basicsal	total	CalculateHra	CreateEmployee
da		CalculateTsal	SearchEmployee

Q)what is class?

class is userdefined reference type datatype which consists of variables and methods

in object orineted programming Languages like C#.net,C#.net,python,Typescript we have to declare variables and methods inside the class only

```
class Student
{
 int rollno;
 String sname;
 int age;
 long phoneno;
 String emailid;
 char gender;
 int total;
 double per;
 void SetStudent()
 { }
 void calTotal()
 { }
 void calPercentage()
 { }
}
```

```
class Employee
{
  int eno;
  String ename;
  Strig designation;
  double bsal;
  double da;
  double hra;
  double tsal;
  void calDa(){ }
  void calHra{ }
  void calTsal(){ }
}
```

```
class Course
{
    int cid;
    String cname;
    int credits;
    void setCourse()
    { }
}
```

Q) Consider that we are developing an application for a Online FoodOrder Management?

Apply Abstraction, Encapsulation?

Identify the states, behaviour belongs to particular classes?

Stated	eno	SetLocation	GetGstAmt
statename	ename	GetLocation	SetDeliveryBoy
cityid	doj	AddLocation	GetDeliveryBoy
cityname	salary	UpdateEmployee	Update DeliveryBoy
locationid	da	ViewEmployee	SetRestaurantType
locationname	hra	SetItems	GetRestaurantType
streetid	tsal	UpdateStock	CheckDeliveryStatus
streetname	deliveryboyid	GetPrice	custid
ino	deliveryboyname	CheckStock	custname
iname	address	SetCusine	OrderFood
price	resttypeid	SetRestaurant	CancelFood
qty	resttypename	GetRestaurant	ChageDeliverAddress
cusineid	SetState	GetCusine	ViewSales
cusinename	GetState	UpdateCusine	AddStreet
restaurantid	UpdateState	ViewCusine	DeleteStreet
restaurantname	SetCity	CreateEmployee	UpdateStreet
restaurantaddress	GetCity	DeleteEmployee	ViewStreet
phno	CreateState	CalDa	
emailid	CreateCity	CalHra	
noofoutlets	UpdateLocation	CalTsal	

Variable:-variable is the name given for a particular Memory Location

Q)what is the purpose of variable?

The purpose of variable is to store the value

Different Types of variables are:-

1. static variable
2. instance variable
3. MethodParameters
4. Local variables

Static variable:- static variable must declare with static keyword

static variable is the static member of a class

static variable must declare inside the class and outside the method with static keyword

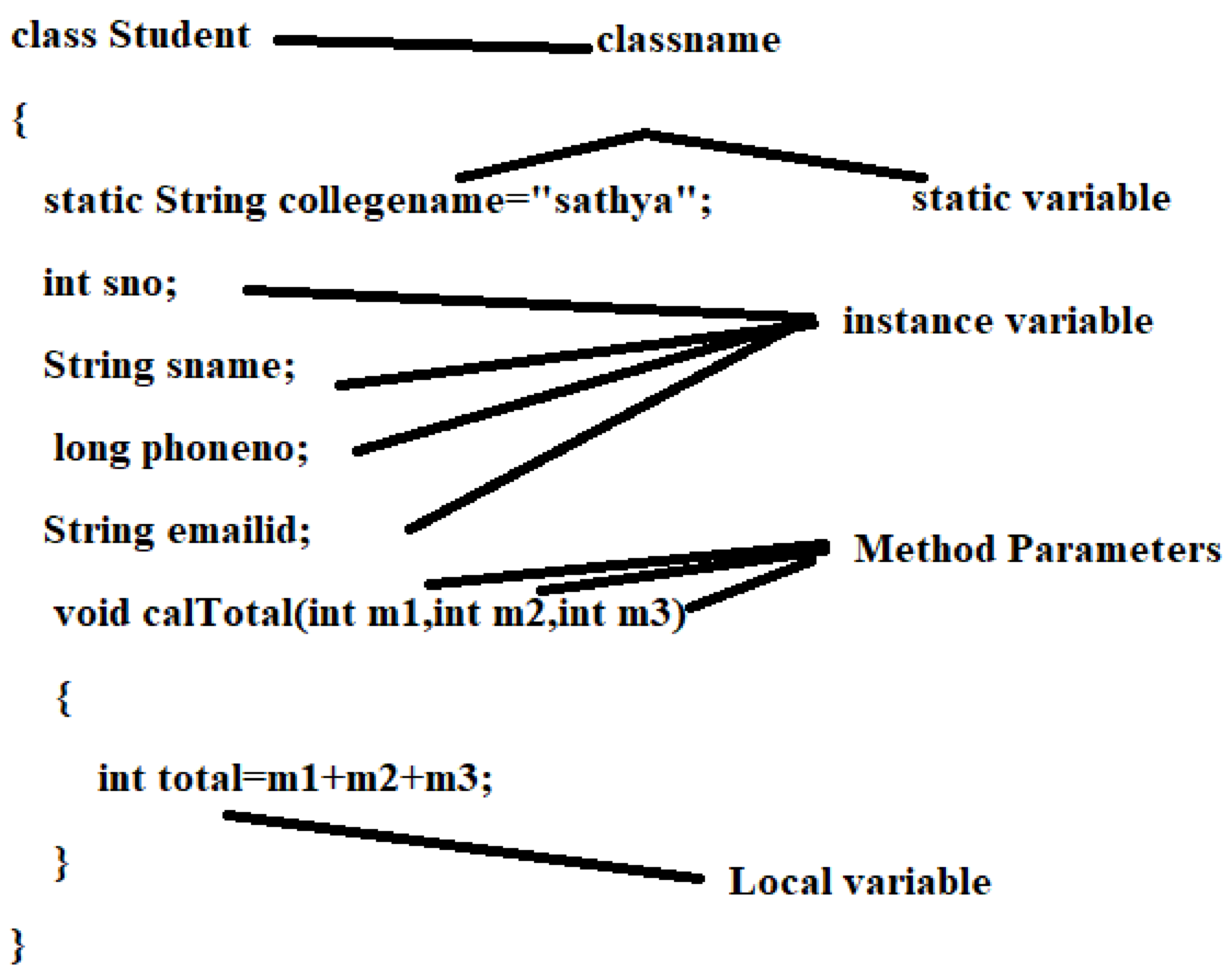
Instance variable:- instance variable must declare inside the class and

outside the method without any keyword

MethodParameters :- The variables that are declared within the method paranthesis are called as MethodParameters

Local variables:- The variables that are declared inside the method are called as Local variables

```
class Employee
{
 static String   collegename="sathya";
  int eno;
  String   ename;
  void CalDa(double bsal)
  {
   double da=0.2*bsal;
   }
}
```



```
class Student
{
  static String collegename="sathya";
  int sno;
  String sname;
   long phoneno;
  String emailid;
   void calTotal(int m1,int m2,int m3)
   {
     int total=m1+m2+m3;
   }
}
```

Q)what is variable declaration?

Declaraing the variable without assigning the value is called as variable declaration

int x;

it is always recomended to declare instance variable

Q)what is variable initialization?

Assigning the value to the variable at the time of declaration

it is always recomended to initialize local variable and static variable

Q)what is variable assignment?

assigning the value to the variable after declaration is called as variable assignment

it is always recomended to declare instance variable

Q)what is object?

object is instance of class

instance means allocating sufficient memory space for instance variables

Q)what will happen when object was created?

whenever we create object for a class then memory will allocate for instance variables

Q)what is Reference vartiable?

Reference vartiable is the name given for the object

Reference vartiable is also called as objectname

Evey object must be identified with some reference

Q)what is the purpose of Reference vartiable?

The purpose of Reference vartiable is to access instance variables and

instance methods

syn to create object:-

```
classname objectname=new classname();

Ex:-  A  a1  =     new A();
      Reference object is created
      is created
```

Every object will have 2 References

1. Default Reference variable (this)
2. User Defined Reference variable

Q)what is this?

this is the default reference variable given for the object

the purpose of this is used to access instance variables

Q)what is the purpose of User Defined Reference variable?

The purpose of User Defined Reference variable is to access instance methods

```
class Student
{
   int sno;
   string   sname;
   int total;
   static void Main(String  [] args)
   {
     Console.WriteLine("Welcome");
   }
}
```

O/p for the above program?

Welcome

Q) Does memory was allocated for instance variables?

No Memory was not allocated for instance variables because object was not created

```
class Student
{
   int sno;
   String   sname;
   int total;
      void SetStudent()
    {
       this.sno=101;
       this.sname="anil";
    }
   static void Main(string  [] args)
   {
     new Student();
    //  object was created and Reference was not created
           Memory was allocated for instance variables
  Student s1=new Student();
  // object and reference both are  created

  }
 }
```

static variable	**instance variable**
static variable must declare with static keyword	no special keyword is required to declare instance variable
static variable must declare inside the class and outside the method with static keyword	instance variable must declare inside the class and outside the method without any keyword
The memory for static variable will allocate on stack	The memory for instance variable will allocate on heap
The memory for static variablewill allocate by CLRwhenever the class was loaded in Ram	The memory for instance variable will allocate whenever a new object was created for the class
one time memory allocation and time memory destruction	memory will allocate everytime whenever a new object was created

The lifespan of static variable is until the program was live	The lifespan of instance variable is until the object was live
	A a1=new A();
	a1=null; //object was destroyed

if the value is common for all the users then declare the variable as static	if the value will change from user to user then decare the variable as instance
we can directly access static variables in static method	we can directly access instance variables in instance method

class A
{
static int x; //var declaration
  static int y;
public static void Main()
{
x=10; //assign the value to
y=20;   variable
Console.WriteLine(x);
        // printing variable value
}
	class A
	{
	int x; //var declaration
	  int y;
	public void show()
	{
	x=10; //assign the value to
	y=20; variable
	Console.WriteLine(x);
	        // printing variable value
	}

we can directly access static variable
in instance method
class A
{
static int x; //var declaration
  static int y;
public void show()
{
x=10; //assign the value to
y=20;   variable
Console.WriteLine(x);
        // printing variable value
}
	if we want to access instance variable in static method then object creation is manadatory
	class A
	{
	 int x; //var declaration
	 int y;
	public static void Main()
	{
	A a1=new A();
	A1.x=10; //assign the value to
	A1.y=20; variable
	Console.WriteLine(a1.x);
	        // printing variable value
	}

Interview Questions:-

1. What is a variable?

   variable is the name given for a particular memory location

   The purpose of variable is to store the value

2. When should we create a variable?

   whenever we want to store any value then we have to create variable

3. **How can we create a variable?**

   using datatype

   syn :- datatype varname;

4. **How to access instance variable?**

   by using this keyword

5. **How many types of variables are there in C#.net?**

   - Static variable
   - Instance variable
   - Local variable
   - Method parameters

6. **What are different places we can create a variable in C#.net?**

   inside the class and outside the method with static keyword

   inside the class and outside the method without static keyword

   inside the method or inside the block

   within the method paranthesis

7. **What is the difference between variable declaration, initialization and assignment?**

variable declaration:- declaring the variable without assigning the value int eno;

it is always recomended to declare instance variables

variable initialization:- assigning the value to the variable at the time of declaration

static String   Collegename="sathyatech";

it is always recomeneded to initialize the values for static variables and Local variables

variable Assignment:- assiging the value to the variable after declaration

modifying the variable data

int cbal; declaration

cbal=5000; assignment

cbal=cbal+2000; assignment

8. What are instance variables?

instance variables are instance members of the class

a sepeate memory was allocated for instance variables when a new object was created

9. What are static variables?

static variables are the static members of a class

10. **What are local variables?**

The variables that are declared inside the method or block are called as local variables

11. What is the difference between method parameters and local variables?

Local variables:-The variables that are declared inside the method or block are called as local variables

MethodParameters:-These are the variables that are declared within the method Paranthesis are called as **MethodParameters** The scope of Method Parameters is more compare with Local variables

```
class A
{
   public void Add(int x,int y)
  {
     if(x>y)
    {
        int z=x+y;
     }
     Console.WriteLine(x+y);
     Console.WriteLine(z);
     }
      public static void Main(String  [] args)
     {
        A a1=new A();
          a1.Add(6,3);
       }
     }
```

12. Where the memory is allocated for instance variables?
Heap

13. Where the memory is allocated for static variables?
stack

14. Where the memory is allocated for local variables?
stack

15. **When the memory is allocated for instance variable?**
    when object was craeted
16. **When the memory is allocated for local variable?**
    when we invoke the method
17. **When the memory is destroyed for local variable?**
    once method execution was completed
18. **What is the scope of local variable?**
    within the method or within the block
19. **What is the scope of instance variable?**
    within the class
20. **What is the scope of static variable?**
    within the class
21. **When to declare the variable as static?**
    if the value is common for all the objects then declare the variable as static
22. **When to declare the variable as instance?**
    if the value is unique for the object then declare the variable as instance
23. **When to declare the variable as local?**
    if we want to declare the variable inside the method
    LV is onetime usage
24. **What is the life span of static variable?**
    until the program is live
25. **What is the lifespan of instance variable?**
    until the object is live
    A a1=new A(); object created and assigned to reference variable

26. What variables will have default value?

instance variables

27. What are the rules on local variables?

LV must declare inside the method or block

LV must initialize with som value before use

LV must not declare as static

```
class A
{
public static void Main(String  [] args)
{
        int x;
    Console.WriteLine(x);
}
}
```

Error:- because value was not assigned

28. If we declare a local variable without assigning a value, what is the value stored?

Error

29. Can we access local variable from other methods?

no

30. Can we access local variable before its declaration statement?

no

31. How can we access static variables?

by using classname

32. How can we access instance variables?

this

33. Can we apply static keyword to a local variable/parameter?

no

34. How many times memory was allocated for static variable?

only once

35. How many times memory was allocated for instance variable?

Every time when a new object was created

36. Can we access Local variable outside the method?

no

37. Can we access Local variable without initialization?

no

38. Can we declare local variable as static?

no

39. The variables that were created inside static method are static?

no local

40. The variables that were created inside instance method are instance?

no local

Developing the First Program:-

1. goto D Drive and create a folder with name Demo

2. open notepad and write the program

```
class Student
{
static string   CollegeName="sathyatech";
int sno=101;
string   sname="anil";
int age=20;
static void Main(String  [] args)
{
Console.WriteLine(CollegeName);
Student s1=new Student();
Console.WriteLine(s1.sno);
Console.WriteLine(s1.sname);
Console.WriteLine(s1.age);
}
}
```

3. save the program in D:/demo with name Student.cs

4. open command prompt compile and execute the program

5. change the Drive

```
D:
```

6. chang the directory

```
cd Demo
```

7. compile the program

```
csc student.cs
```

whenever we compile the program

. compiler will check for synatx errors

. if there are no syntax errors then compiler will generate .exe file

8. execute the program

```
Student.exe
```

Sathya Technologies

S A T H Y A T E C H N O L O G I E S

K A N N A B A B U

Chapter-2

Methods

By. Kannababu

Method: -Method is a subprogram which is used to perform specific operation

steps to work with methods:-

1. Example for static method

```
class A
{
  static int x;
  static int y;
  static int z;
  public static void main(String[] args)
  {
     z=x+y;
    System.out.println("sum is "+z);
  }
}
```

static method

B

Example for instance method:-

```
class A
{  int x;   int y;   int z;
   public void setValues()
   {
        x=5;
        y=3;
   }
   public void getSum()
   {    z=x+y;
      System.out.println(z);
   }
   public static void main(String[] args)
   {
       A a1=new A();
        a1.setValues();
        a1.getSum();
   }
}
============================
```

instance method

Methods are of 2 types

1. Static method

2. Instance method

Static method: -static method is used to perform operations on static variables

static method can be invoked by using classname

Instance method: - instance method is used to perform operations on instance variables

Instance method can be invoked by using object name

Q) What are Method parameters?

The variables that are declared within method parenthesis are called as Method parameters

Q) What is the purpose of Method parameters?

The purpose of Method parameters is to pass the data to instance variables at runtime

Q)consider we are developing an appn for a college to display student totalmaks,percentage?

```
class Student
{
  static string   cname;
   static string   caddress;
   int sno; string   sname;
  static void setCollegeData()
  {
     cname="sathyatech";
     caddress="Hyderabad";
  }
```

```
void getStudent()
 {
    Console.WriteLine(sno+sname);
  }
public static void Main(String  []
args)
 {
   Student.setCollegeData();
   Student.getCollegeData();
  Student s1=new Student();
  s1.setStudent(101,"anil");
```

```
static void getCollegeData()
 {
Console.WriteLine(cname+caddress);
  }
 void setStudent(int no,String   name)
 {
   sno=no;
   sname=name;
 }
```
	```
	   s1.getStudent();
	   Student s2=new Student();
	   s2.setStudent(102,"sunil");
	   s2.getStudent();
	 }
	}
	```

observation:-

1. write the program

2. save the program in D:/ Demo with name Student.cs

3. compile the program

    csc Student.cs

   whenever we compile the program then csc compiler will check for syntax errors

   if there are no syntax errors then the compiler will generate .exe file

4. execute the program

   Student.exe

   Whenever we execute the program

   a. class will loaded in CLR

   b. CLRwill allocate memory on Ram

   c. CLR will allocate memory for static variables on stack

   d. All the methods will load in methodarea

   e. CLR wil search for static void Main() and it will load Main() in execution engine and the code that was written inside Main() will gets executed

f. Student.setCollegeData();

invoking setCollegeData(); method

setCollegeData(); will be loaded in executionengine

g. Student.getCollegeData();

getCollegeData() will invoke and print cname and caddress in console window

h.

Student s1=new Student();

s1.setStudent(101,"anil");

s1.getStudent();

new is a dynamic memory allocation operator which is used to create object

Whenever object was created then memory was allocated for instance variables

s1 is reference variable

```
class A
{
   void show()          ✓
   {
   }
  void display()
  {
    return 10;          ✗
  }
  int getData()
  {
    return 10;          ✓
  }
  String printData()
  {
    return "sathya";
  }                     ✓

  float getResult()
  {
    return 2.3f;        ✓
  }
  String getString(int x)
  {
    return x+"abc";     ✓
  }
  double getDouble(int x)
  {
    return 10+x;        ✗
  }
  bool getR(int x,int y)
  {
    return x>y;         ✓
  }
}
```

1. What is a method, What is the use of method?
2. What is the syntax for creating a method?
3. What is the difference between method prototype and signature?
4. What is the return type of a method that does not returns any value?
5. Which method can be defined only once in a program?
6. Can we create a method inside another method?
7. What are the different types of methods do we have in C#.net?
8. How to declare static method?
9. How to declare instance method?
10. What is a void and non-void method?
11. What is a parameterized and non parameterized method?
12. How to access static method?
13. How to access static method?
14. How to access instance method?
15. Can we access static method by using objectname?
16. Can we access instance method by using classname?

S A T H Y A T E C H N O L O G I E S

K A N N A B A B U

17. What is the difference between calling a method and executing a method?

18. Can we call a method inside another method?

19. Can we call a static method in another static method?

20. Can we call a static method in instance method?

21. Can we call instance method in a static method?

22. Can we call method inside a constructor?

23. Can we call method without reference variable?

24. When Main() is called?

25. Why Main() is static?

26.

```
class A
{
   public static void Add(int x, int y)
   {
     Console.WriteLine(x+y);
   }
}
```

Method Type	Keyword	Return type	Method name	i/p parameters

```
    Invoke Method
    public  static void Main(String  [] args)
   {



   }
```

Expected O/P:-

27.
```
class A
{
    public  void Sub(int x, int y)
    {
        Console.WriteLine ("Diff is"+(x+y));
    }
}
```

Method Type	Return type	Method name	i/p parameters

Invoke Method
```
public  static void Main(String  [] args)
{



}
```

Expected O/P:-

28.
```
class A
{
    public  int Add(int x, int y)
    {
        return x + y;
    }
}
```

Method Type	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p
```
public  static void Main(String  [] args)
{



}
```

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

29.class A
```
{
public static  String   Add(int x, int y)
{
  int z = x + y;
  return "sum is"+z;
}
```

Method Type	Keyword	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p
```
public  static void Main(String  [] args)
{



}
```

K
A
N
N
A
B
A
B
U

30.
```
class A
{
    public  static int CalTotalMarks(int m1, int m2,int m3)
    {
        int total = m1 + m2 + m3;
        return total;
    }
}
```

Method Type	Keyword	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p
```
public  static void Main(String  [] args)
{


}
```

31.
```
class A
      {
        int cbal = 5000;
       public  double Deposit(double amt)
       {
           cbal = cbal + amt;
            return cbal;
         }
      }
```

Method Type	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p

```
public  static void Main(String  [] args)
{



 }
```

32.
```
class A
 {
 public static int GetString  (char x,String   y)
 {
    int z = x + y;
    return z;
 }
 }
```

Method Type	Keyword	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p

```
public  static void Main(String  [] args)
{

}
```

33.
```
class A
          {
            public bool GetValue(int x,int y)
           {
                return x > y;
           }
          }
```

Method Type	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p

```
public  static void Main(String  [] args)
{


}
```

34.
```
class A
{
   int x=5;
   int y=4;
   public int Add(A a1)
   {
       int z = a1.x + a1.y;
       return z;
   }
}
```

Method Type	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p

```
public  static void Main(String  [] args)
{

 }
```

35.

```
class A
{
 int x=5;    int y=4;
    public A Add()
    {
        return new A();
    }
}
```

Method Type	Return type	Method name	i/p parameters	returnvalue

Invoke Method and print the o/p

```
public  static void Main(String  [] args)
{




}
```

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

Chapter-3

Constructors

By. Kannababu

Constructors:-

Constructor is a default method which is used to initialize the values for instance variables at the time of creating object

Different types of constructors:-

1. Default costructor
2. parameterized constructor
3. copy constructor

Rules to be followed whil creating constructor:-

1. constructor name and class name both must be same
2. constructor does not have returntype
3. constructor will gets invoked whenever object was created for a class

Default constructor:- it is used to initialize default values for instance variables

```
  0--->int
  0.0 --> float,double
   null--->string
```

syn to create default constructor:-

```
  class A
  {
       public A()
       {


        }
  }
```

Q)what is object?

object is instance of a class

instance means allocationg sufficient memory space for instance variables

syn to create object:-
classname objectname=new constructorname();

1. object is created
new is a keyword which is used to create object i.e. memory was allocated for instance variables
2. Constructor will gets invoked and values are initialized for instance variables
3. Reference variable is created
4. address of the object was assigned to reference

note:-
Consider class with name A

1. new A();
    a. object created
    b. reference created
    c. object and reference created
    d. object created and constructor invoked
2. A a1;
    a. object created
    b. reference created
    c. object and reference created
    d. none
3. A a1=new A();
    a. object created
    b. reference created
    c. object and reference created
    d. none

Default constructor:-
1. System defined Default constructor
2. Userdefined Default constructor

System defined Default constructor:- If the developer is not creating any constructor within the class then compiler will create a default constructor and initialize default values for instance variables

User defined Default constructor:- it is used to initialize userdefined values instead of default values

parameterised constructor:-it is used to pass values at runtime time at the time of creating object

->At the time of creating the constructor we have to declare parameters and the time of creating object we have to pass values

-> The no of values,order of values,type of values that we pass must match with no of,order of,type of parameters

syn:-

```
class classname
{
  classname(parameters)
  {
  }
}
```

Requirement:-Develop an application that Maintain 3 student details

identify the no of classes that are required and no of objects that are required

```
class Student
{
  int sno;    String   sname;     int age;
  Student(int no,String   name,int a)
  {
   sno=no;
   sname=name;
   age=a;
  }     void GetStudent()
  {
     Console.WriteLine(sno);
     Console.WriteLine(sname);
     Console.WriteLine(age);
  }
  static void Main(String  [] args)
  {
     Student s1=new Student(101,"anil",22);
      s1.GetStudent();
    Student s2=new Student(102,"sumitbajaj",24);
     s2.GetStudent();
  }
}
```

SATHYATECHNOLOGIES

KANNABABU

Constructors

1. What is a constructor?
2. What is the use of constructor?
3. What are the rules in creating a constructor?
4. If we place return type in constructor declaration, is it leads to
   Compile time error?
5. Can we define a method with class name?
6. How many types of constructors will C#.net supports?
7. What is a copy constructor will C#.net supports its creation?
8. What is a static constructor?
9. When static constructor is invoked?
10. If static constructor and static void Main() are declared in same class which one will first execute?
11. What is default constructor?
12. What are the differences between default and parameterized constructor?
13. When will compiler provide constructor in a class?
14. Can we have both default & parameterized constructor in class?
15. What is the constructor overloading?
16. If we invoke one constructor, will all other constructors are executed?
17. What are the differences between methods and constructors?
18. Can we create object without constructor?
19. Can we call a method in constructor?
20. Can we declare constructor in another constructor?

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

Chapter-4

Inheritance

By. Kannababu

inheritance:-inheritance is a mechanism of Establishing the relationship between the classes

it is a mechanism of obtaining the variables and methods from one class to another class

The class which is giving variables and methods is called as base class or baseclass or parent class and the class which is taking variables and methods is called as subclass or child class or derived class

Rules to be followed while applying inheritance:-

1. Minimum 2 classes must exist

2. Relationship must exist between the classes

Advantages of inheritance:-

1. Reusability

2. Extensability

3. Reimplementation

Q) what is Reusability?

Elimination of repetition of code is called as Reusability

class Student	class Employee

```
class Student
     {
int no;
String   name;
int age;
String   gender;
long phoneno;
String   emailid;
     int total;
     int percentage;
char grade;
  }
```
	```
	class Employee
	 {
	int no;
	String   name;
	int age;
	String   gender;
	long phoneno;
	String   emailid;
	    double bsal;
	   double da;
	   double hra;
	   double tsal;
	 }
	```

```
class Student
     {
int no;
String   name;
int age;
String   gender;
long phoneno;
String   emailid;
     int total;
     int percentage;
char grade;
  }
```

```
class Employee
 {
int no;
String   name;
int age;
String   gender;
long phoneno;
String   emailid;
    double bsal;
   double da;
   double hra;
   double tsal;
 }
```

Repeatedly writing the same variables and methods in multiple classes will increase the no of lines of code

Solution:- Code Reusability

```
      class Person
     {
int no;
String   name;
int age;
String   gender;
long phoneno;
String   emailid;
  }
```

class Student :Person
{
int total;
int percentage;
char grade;
}
	class Employee :Person
	{
	double bsal;
	double da;
	double hra;
	double tsal;
	}

Only variables and methods will participate in inheritance

constructors will never participate in inheritance

```
class A
{
  2 variables     no of variables in A 2
  3 methods       no of methods in A 3
  2 constructor   no of constructors in A 2
}
class B :A
{
  3 variables     no of variables in B 5
  2 methods       no of methods in B 5
  1 constructor   no of constructors in B 1
}
```

Points to remember:-

1. Minimum 2 classes must exist to apply inheritance

2. Relationship must exist between the classes

3. we can establish the relation between the classes by using :keyword

4. only variables and methods will particpate in inheritance

5. constructors will never particpate in inheritance

6. in inheritance always create object for Derived class

7. whenever we create object for derived class then object was not created for base class

8. whenever we create object for derived class then memory will allocate for base class

and derived instance variables

class A
{
int x;
int y;
}
	class B :A
	{
	int z;
	}

new B();

9. we can refer subclass object in 2 ways

a. Base classname

b. Derived classname

Q)what is Upcasting?

sub class object assigned to baseclass Reference

```
A a1=new B();
```

we can refer derived class object with derivedclassname

```
B b1=new B();
```

Q)in what scenario we have to create base class Reference?

in the scenario where single parent and multiple child exist then we have to

create base class Reference

```
        class A
        {
        }
class B :A class C :A class D :A
{              {              {
}              }              }
        A a1=new -----------();
```

in these scenarios server,Framework,Container,CLRwill create object

Q)in what scenario we have to create sub class Reference?

in the scenario where single parent and single child exist then we have to

create sub class Reference

10. Accessability

what methods we can access with base class Reference ?

with base class reference we can access only base class methods

```
class A
{
  public void Show()
  {
    Console.WriteLine("i am show");
  }                                    A a1=new B();
}                                      a1.Show();   valid
class B :A                             a1.Display(); Error
{
  public void Display()
  {
    Console.WriteLine("i am Display");
  }   }
```

Q) what methods we can access with sub class Reference ?

with subclass Reference we can access both baseclass methods and subclass methods

```
B b1=new B();

b1.Show();          valid

b1.Display();       valid
```

Different Types of inheritance:-

1. Single Level Inheritance

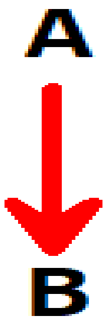
2. MultiLevel Inheritance

3. Heiracheal Inheritance

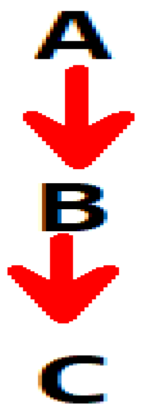
4. Multiple Inheritance

5. Hybrid Inheritance

Single Level Inheritance:- createing a Derived class by using Single Base class

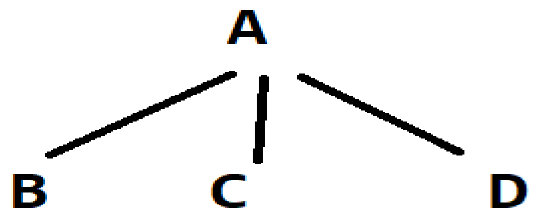
class A
{
}
class B :A
{
}
	

MultiLevel Inheritance:- creating a Derived class by using another Derived class

class A
{ }
class B :A
{ }
class C :B
{ }
	

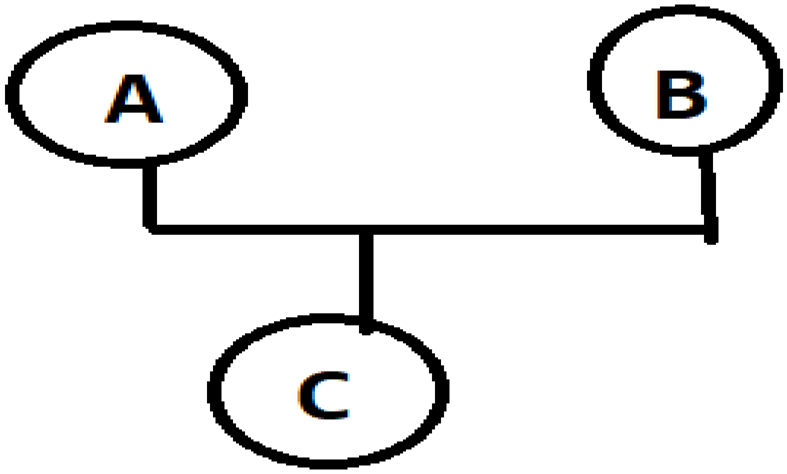
Hierarchal Inheritance:- creating multiple derived class by using single base class

```
          class A
          {


          }
class B :A class C :A class D :A
{            {            {


}            }            }
```



Multiple Inheritance:- creating a derived class by using multiple base classes

C#.net doesnot support multiple inheritance by using classes



Q)why C#.net doesnot support multiple inheritance?

if same method exists in both the classes

because of inheritance these 2 methods are available in Derived class

with derived class reference if we access the method then an ambiguity problem will occur to avoid the avoid teh above ambiguity C#.net doesnot support multiple inheritance by using classes

in C#.net ,C#.net we can achive multiple inheritance by using interfaces
Hybrid inheritance:- it is the combination of any 2 inheritance except multiple inheritance

===

Inheritance

1. What is inheritance?
2. What are the advantages of inheritance?
3. Define Reusability?
4. What are the types of inheritance?
5. How can we implement inheritance in C#.net?
6. Does a variable will inherit?
7. Does a static variable will participate in inheritance?
8. Can we inherit methods?
9. Can we inherit static methods?
10. Whenever we create object for derived class for how many classes object is created?
11. Can we refer derived class object by using baseclass reference?
12. If the derived class object is refer with base classname which class variables and methods we can access?

13. If the derived class object is refer with Derived classname which class variables and methods we can access?
14. Does constructors will participate in inheritance?
15. In how many ways derived class object can refer?
16. which members are not inherited from base class?
17. Which operator is used for inheritance?
18. How many subclasses can create from a base class?
19. If a variable value or method logic is changed in base class will it affected to all its sub classes?
20. How many types of inheritance supported by C#.net?
21. Why C#.net does not support multiple inheritance with classes?
22. How do you implement multiple inheritance in C#.net?
23. Can a class extend itself?
24. How do you restrict a member of a class from inheriting to its sub class?
25. How to stop inheritance?
26. What is single level inheritance?
27. Can we derive a class from another class?
28. Is it possible to develop a class without inheritance?
29. If we develop single class, is it really single class?
30. What is multi level inheritance?

31. If we create single level inheritance, is it really single level inheritance?
32. What is Hierarchical inheritance?
33. How many subclasses can create from a single class?
34. As a subclass developer can we stop multilevel inheritance?
35. Assume you have developed 3 classes A, B, C as normal classes,
36. is there any inheritance
37. What is Multiple inheritance?
38. What is the order of compiling, loading and instantiating classes in inheritance?
39. When we load sub class will all its base classes also loaded to JVM?
40. When we load base class will all its sub classes also loaded into JVM?
41. Can we call base class variable and methods in sub class directly by their name?
42. Can we call sub class variable and methods in base class directly by their name?
43. Will Main method executed from both base class and sub class?
44. Will constructor executed from both base class and sub class?

45. Will base class object also created, when sub class object is created?
46. Prove that base class object is not created when sub class object is created?
47. When base class object is not created, why constructor is executed from base class?
48. What a sub class object contains?
49. Prove that base class object is not created when sub class object is created?
50. Which class is the base class for all classes in C#.net?
51. Does interface will participate in inheritance?

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

Chapter-5

Polymorphism

By. Kannababu

The word polymorphism means having multiple forms. The term Polymorphism gets derived from the Greek word where poly + morphos where poly means many and morphos means forms.

In Object Oriented programming Languages like C#.net,C#.net we can achieve Polymorphism in 2 ways:-

1. OverLoading
2. OverRiding

OverLoading or Method OverLoading: - it is a process of defyning multiple methods with same method name and with different method parameters

```
class Mltply {
 void mul(int a, int b) {
  Console.WriteLine("Sum of two=" + (a * b));
 }

 void mul(int a, int b, int c) {
  Console.WriteLine("Sum of three=" + (a * b * c));
 }
}
class Polymorphism {
 static void Main(String   args[]) {
  Mltply m = new Mltply();
  m.mul(6, 10);
  m.mul(10, 6, 5);
 }
}
```

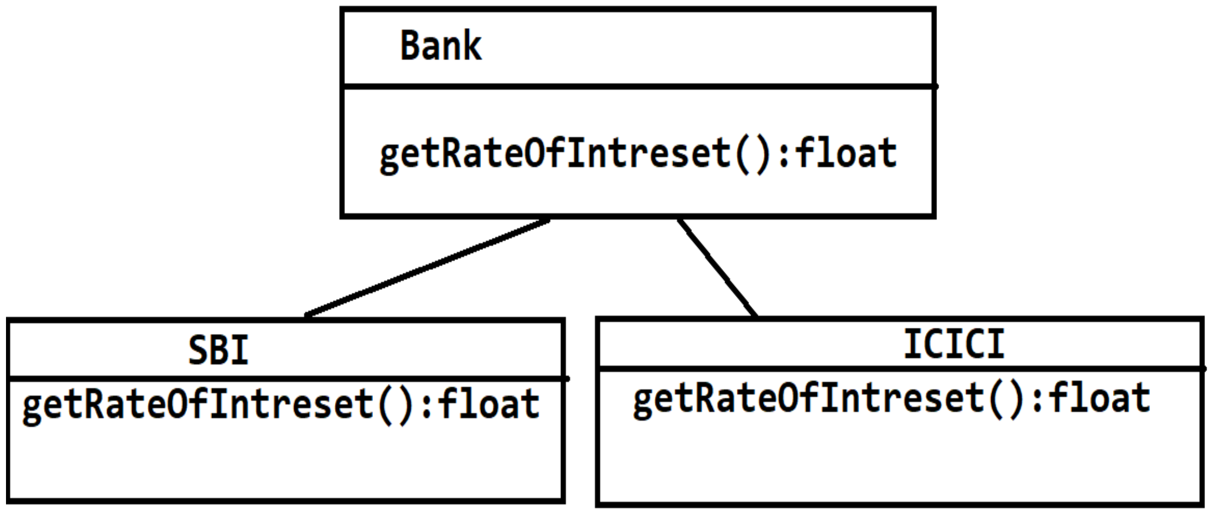
Method Overriding: - it is a process of Reimplementing the base class method in Derived class

Rules of Method Overriding in C#.net:-

- **Argument list:** The argument list at the time of overriding method need to be same as that of the method of the parent class. The data types of the arguments along with their sequence must have to be preserved as it is in the overriding method.
- **Access Modifier:** The Access Modifier present in the overriding method (method of subclass) cannot be more restrictive than that of an overridden method of the parent class.
- The private, static and final methods can't be overridden as they are local to the class.
- Any method which is overriding can throw any unchecked exceptions, in spite of whether the overridden method usually method of parent class might throw an exception or not.

```
class parent {
 public void work() {
  Console.WriteLine("Parent is under retirement from
work.");
 }
}
class child :parent {
 public void work() {
  Console.WriteLine("Child has a job");
  Console.WriteLine(" He is doing it well");
 }
 public static void Main(String   argu[]) {
  child c1 = new child();
  c1.work();
 }
```

Consider a scenario where Bank is a class that provides a method to get the rate of interest. However, the rate of interest may differ according to banks. For example, SBI, ICICI, and AXIS banks are providing 8.4%, 7.3%, and 9.7% rate of interest.


Bank
getRateOfIntreset():float
SBI
getRateOfIntreset():float
ICICI
getRateOfIntreset():float

```
class Bank
{
float getRateOfInterest(){return 0;}
}
class SBI :Bank
{
float getRateOfInterest(){return 8.4f;}
}
class ICICI :Bank
{
float getRateOfInterest(){return 7.3f;}
}
class AXIS :Bank{
float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism
{
static void Main(String   args[]){
Bank b;
b=new SBI();
Console.WriteLine("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
Console.WriteLine("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
Console.WriteLine("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Output:

```
SBI Rate of Interest: 8.4

ICICI Rate of Interest: 7.3

AXIS Rate of Interest: 9.7
```

Sathya Technologies

S A T H Y A T E C H N O L O G I E S

K A N N A B A B U

Chapter-6

this, base,this(),base()

By. Kannababu

this:- this is the default reference variable given for the object

Base Keyword in C#.net

The base keyword in C#.net is a reference variable which is used to refer immediate parent class object.

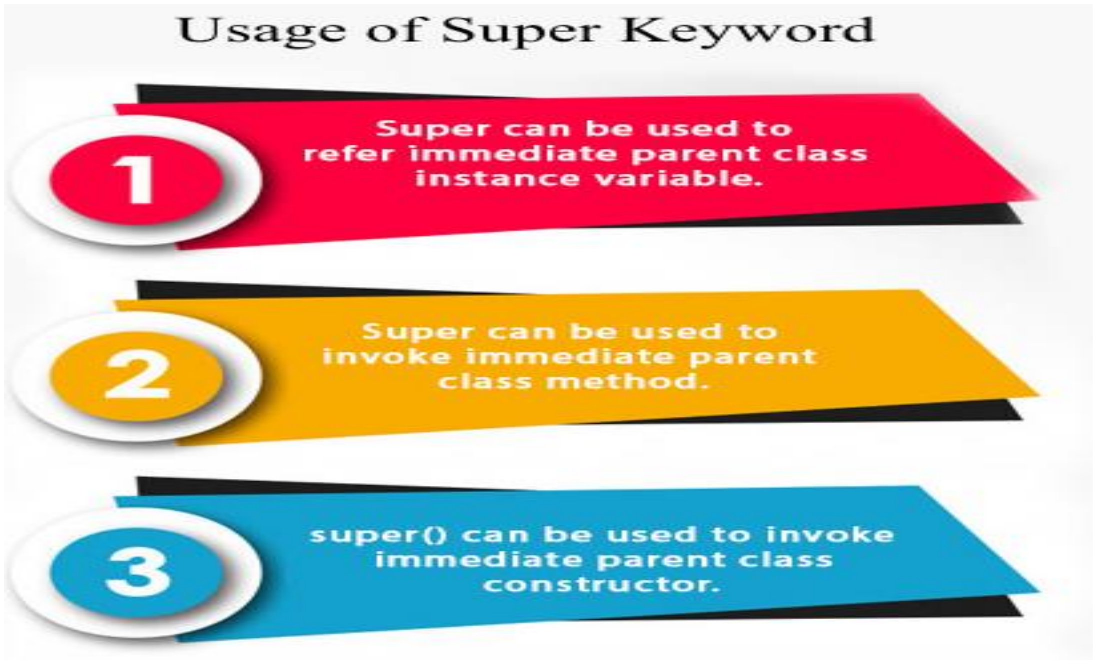
Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by base reference variable.

Usage of C#.net base Keyword

base can be used to refer immediate parent class instance variable.

base can be used to invoke immediate parent class method.

base() can be used to invoke immediate parent class constructor.


Usage of Super Keyword
1
Super can be used to refer immediate parent class instance variable.
2
Super can be used to invoke immediate parent class method.
3
super() can be used to invoke immediate parent class constructor.

1) base is used to refer immediate parent class instance variable.

```
class A{
String   color="white";
}
class B :A{
String   color="black";
void printColor(){
Console.WriteLine(color);
Console.WriteLine(base.color);
}
}
class TestBase1{
public static void Main(String   args[]){
B b1=new B();
b1.printColor();
}}
```

In the above example, A and B both are classes have a common property color. If we print color property, it will print the color of current class by default. To access the parent property, we need to use base keyword.

base can be used to invoke parent class method

The base keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.

```
class A{
void show(){Console.WriteLine("A show..");}
}
class B :A{
void show(){Console.WriteLine("B show...");}
void print(){Console.WriteLine("print...");}
void display()
{
        base.show();
        print();
}
}
        }
        class TestBase2{
        public static void Main(String   args[]){
        B b1=new B();
        B1.display();
        }}
```

```
o/p:- I am A show
      print...
```

base is used to invoke parent class constructor.

The base keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
class A{
A(){Console.WriteLine("A constructor  is invoked");}
}
class B :Al{
A(){
base();
Console.WriteLine("B constructor is invoked");
}
}
        class TestBase3{
        public static void Main(String   args[]){
        B b1=new B();
        }}
```

Output:-
```
A constructor  is invoked

B constructor  is invoked
```

base example: real use

Let's see the real use of base keyword. Here, Emp class inherits Person class so all the properties of Person will be inherited to Emp by default. To initialize all the property, we are using parent class constructor from child class. In such way, we are reusing the parent class constructor.

```
class Person{
int id;
String   name;
Person(int id,String   name){
this.id=id;
this.name=name;
}
}
class Emp :Person{
      float salary;
      Emp(int id,String   name,float salary){
      base(id,name);//reusing parent constructor
      this.salary=salary;
      }
      void display(){Console.WriteLine(id+" "+name+" "+salary);}
      }
      class TestBase5{
      public static void Main(String  [] args){
      Emp e1=new Emp(1,"ankit",45000f);
      e1.display();
      }}
```

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

Chapter-7

Abstract class

By. Kannababu

Concrete method:-

A method is said to be concrete, if it contains both declaration and definition.

Concrete class: A class is said to be concrete, if all the methods of that class are concrete.

A concrete class can be instantiated i.e. we can create an object of concrete class and using that object we can access that members of that class.

Abstrarct Method:

If a method contains only declaration without any defination then, it is said to be an abstract method.

Syntax of abstract method:

```
abstract returntype methodName(list of parameters);
```

Abstract methods must end with semi colon(;) and they must be declared with abstract keyword.

Abstract class: if a class contains some abstract methods then, the class should be called as abstract class.

Syntax of abstract class:

```
abstract class ClassName{

}
```

An abstract class can be combination of abstract methods and non-abstract (concrete) methods.

An abstract class can contain zero or more abstract methods.

If a class does not contain any one abstract methods then, declareing the class as abstract is optional.

If a class contains at least one abstract method then, declaring the class as abstract is mandatory.

An abstract class cannot be instantiated i.e. we cannot create object of the abstract class and therefore we cannot access the members of that class. In order to access the members of the abstract class, we need to inherit the abstract class into another class and override all tha abstract methods available in the abstract class.

```
Syntax: abstract class A{

 }

 class B :A{

}
```

If the subclass does not override at least one abstract method then, declare that subclass also as abstract.

```
abstract class Operation{
void msg() {
Console.WriteLine("welcome friends");
}
abstract void twice(int a);
class Program1 :Operation{
void twice(int x){
Console.WriteLine(" result1:"+(x+x));
}
class program2 :Operartion{
Void twice=(int y)  {
Console.WriteLine("reult2:"+(y*2));
}
}
class program3 :Operartion{
Void twice=(int z)  {
Console.WriteLine("reult3:"+(z<<1));
}
}
class AbstractDemo{
public static void Main(String  []args) {
Program1 p1=new Program();
p1.mgs();
p1.twice(5);
Program1 p2=new Program();
p2.mgs();
p2.twice(6);
Program1 p3=new Program();
p3.mgs();
p3.twice(8);
}}
```

Q) why we are allowed to instantiate an abstract class?

A) Assume we are allowed to create an object of abstract class, uisng that object if we invoke a concrete method then, it will be executed because it contains defination, but using that object if we invokw an abstract method then, it will lead to an unsafe operations because it does not conatin any defination. To avoid the unsafe operations, we are not allowed to instantiate an abstract class.

Q)Why should we declared a method as abstract?

A) A method should be declared as abstract, when we want a method to be implemented by differnt programmers with different logics.

Q)Why should we declare a class as abstract even though it doesn't contain any abstract methods?

A) We declare the class as abstract even though it does not contain any abstract methods, when we don't want an object our class to be created. By the class as abstract we are restricting HAS-A relationship and forcing to use IS-A relationship.

An abstract class can contain Main method and it can be executed.

```
abstract classs Sample{

public static void Main(String  []args){

Console.WriteLine("abstract class Main method");

}}
```

Every class in C#.net, either predefined or user defined ,either abstract or concrete will be sub class of Object class either abstract directly or indirectly

Every class in C#.net, either abstract class or concrete class will contain a constructor whether we specify or not.

The constructor of the abstract class cannot be executed directly, it can be executed indirectly by creating an object of its child class.

Abstract class cannot be instantiated, but we can declare a reference of the abstract class.The reference of the abstract can used to refer to an object of any of its child class which is concrete.

```
abstrac class Shape{

int dim1;dim2;

Shape(int dim1,int dim2){

this.dim1=dim1;

this.dim2.dim2;

}

abstract double area();

}

class Rectangle :Shape{
```

```
Rectangle(int length,int breadth) {

base(length, breadth);

}

double area() {

return dim1*dim2;}}

class Traingle :Shape {

Traingle (int base, int height) {

base(base,height);

}

double area() {

return 0.5*dim1*dim2;

}

}

class Calculation{

public static void Main(String  []args)

{

Shape s;

s=new Rectangle(3,4);
double res=s.area();
Console.WriteLine("Rectangle area;"+res);
s.new Trainglel(5,6);
res=s.area();
Console.WriteLine("Traingle area:"+res);
}}
```

Modifiers:-

Modifiers can be applied to variable, methods and inner classes.

The final modifier can be applied to variables, methods and classes.

The abstract modifier can be applieed to methods and classes.

Illegal combination of modifiers for a method:

A method can't be decalred as abstract and final because the abstract keyword says we must override the method where as the final keyword says we must not override the method.

A method can't be declared as abstract and static because if a static method is invoked directly by using class name then, it will lead to unsafe operation because it doesn’t have definition.

Illegal combination of modifiers for a class:

A class can't be declared as abstract and final because the abstract keyword says we must inherit the class whereas the final keyword says we must not inherit the class.

We can restrict HAS-A relationship by declaring the class as abstract and we can restrict IS-A relationship by declaring the class as final but we can't restrict both relationships at the same time.

An abstract class can have final method but a final class cannot have abstract method.

S
A
T
H
Y
A
T
E
C
H
N
O
L
O
G
I
E
S

K
A
N
N
A
B
A
B
U

Chapter-8

Interfaces

By. Kannababu

Interface

Interface is a type which consists of public abstract methods and public static final variables

An interface is used as a contract or an agreement between itself and its implemented class

We can achieve multiple inheritance by using interfaces

Syntax:

```
  interface interfaceName
{
 variables;
 methods;
 }
```

Ex:-

```
interface A
{
 Void show();
}
class B : A
{
 Public void show()
{
  Console.WriteLine("I am Show");
}
}
```

```
Class Test
{
  Public static void Main(String  [] args)
{
   B b1=new B();
 b1.show();
}
}
```

Example:

```
   interface InterfaceName {

    }

class SubClass : InterfaceName {

}
```

→If a class is implemented an interface then that class must provide implementation to all the members available in that interface.

→If the subclass does not provide implementation to atleast one abstract method then,declare the subclass as abstract.

→an interface can have any number of implementation classes.

Ex:-

Syntax:-

```
class ClassName : interface1,interface2,.... {

 }
```

Note:- we cannot create an object for interface and abstract class?

Rules for inheriting a class and interface

1) A class can implements almost one class.

2) A class can extends any number of interfaces.

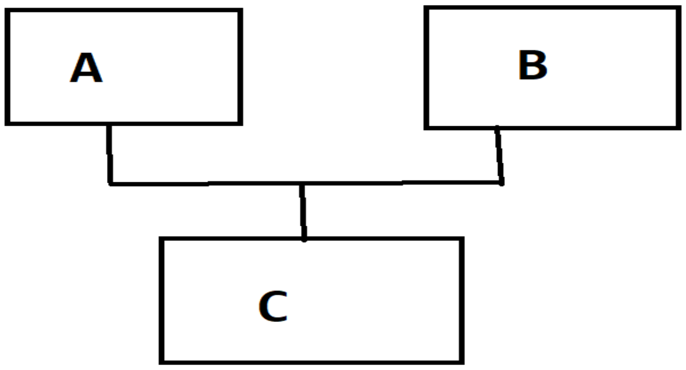
3) An interface can extends any number of interface.

4) A class can extend another class and implement any number of interfaces together at the same time

Q)what is multiple Inheritance?

Creating a Derived class from multiple base classes is called as multiple inheritance

We can achieve multiple inheritance by using interfaces



```
interface A      interface B
{                {
}                }

  class C  implements A,B
  {

  }
```

Differences between abstract class and interfaces:-

Abstract class	interface
1) An abstract class is a combination of abstract methods and concrete	1) An interface is a collection of only abstract methods
2) An abstract class can contains abstract methods, concrete methods,	2) An interface will contain only abstract methods which are instance methods.
3) Declaring an abstract method with abstract keyword in an abstract method with abstract keyword in an interface is optional	3) Declaring an abstract method with abstract keyword in an interface is optional.
4.The members of an abstract class can be either public or non public	4) The members of an interface will be only public.
5) The value of the variable in as abstract class can be modified or fixed 6)An abstraction class can contain both instance variables and statice variables 7).An abstraction class will contain a constructor where we specified or not 8)An abstract class can be inherited into a class by using :key	5) The value of the variables in an an interface cannot be modified because they are by default fixed. 6)An interface only can contain only static variable 7.)An interface will not contain constructor 8)An interface can be inherited into a class can be by using implement key word

word	
9)An abstract class can be :at most one class	9) An interface can :any number of interfaces
10).Using abstract class we can not achive multiple inheritance	10)Using interface we can achive multiple inheritance
11).An abstract class can can implement any number of interface	11)An interface cannot implement any interface
12)An abstract class can inherite from both class and interface	12)An interface can inherit from only interface
13)Object is the base most class of all the methods	13)There is no base interface in C#.net
	14)An interface cannot have final methods
15)An abstract class will contain abstract methods so that different programmers provide different implemention	15)An interface will contain abstract methods given by the client so that the programmer provides implementation class

.