```
public class classstr
{
 static integer r;
    public static void add()
    {
        integer x=90; // static variables
        integer y=100;
       // integer r;
        r=x+y; //static variable can be used in static method
        system.debug('show me result '+r);
    }
    public void mul()
    {
        integer x=900;
         r=x+400;  ////static variable can be used in non static method
        system.debug('show me value r'+r);
    }


}
```

…………………………………………………………………………………….

Task1: static variable can be used in Nonstatic method[already done]

Task2:Non static variable is not allowed in static method.

Task3:Static variable can be used in static also and non static method also

Task4:Static methods can be called in non- static method.

……………………………………………………………………………………………………..

```
public virtual class parentclass {
    //how to use inheritance
    //First concept is one parent class and one child class
    //Child class inherits the properties of parent class and have its own Functionality
    public static integer add(integer x,integer y)
    {
        integer result;
        result=x+y;
        system.debug('show me addition '+result);
        return result;


    }
    private void disp()
    {
        system.debug('hello');
    }




}
```

………………………………………………………………………………………………………

```
public class childclassA extends parentclass
{


    public  static void display(integer k,integer z)
    {
        //integer k=90;
        //integer z=100;
        integer result;
        result=add(k,z);
        disp();
    }
}
```

……………………………………………………………………………………………………..

```
public virtual class parentclass {
    //how to use inheritance
    //First concept is one parent class and one child class
    //Child class inherits the properties of parent class and have its own Functionality
    public static integer add(integer x,integer y)
    {
        integer result;
        result=x+y;
        system.debug('show me addition '+result);
         disp();
        return result; //program execution terminates



    }
    private static void disp()
    {
        system.debug('hello');
    }




}
```

…………………………………………………………………………………………………..

```
public class childclassA extends parentclass
{



    public  static void display(integer k,integer z)
    {
        //integer k=90;

        //integer z=100;

        integer result;

        result=add(k,z);



    }

}
```

……………………………………………………………………………………………………..

Example on Protected Method

```
public virtual class parentclass {

    //how to use inheritance

    //First concept is one parent class and one child class

    //Child class inherits the properties of parent class and have its own Functionality

    protected integer add(integer x,integer y)

    {

        //protected method cannot be called outside

        //proteced method cannot use static

        integer result;

        result=x+y;

        system.debug('show me addition '+result);


        return result; //program exution terminates



    }


}

public class childclassA extends parentclass

{
    public void display(integer k,integer z)
    {
        //integer k=90;
        //integer z=100;
        integer result;
        result=add(k,z);



    }
}
```
……………………………………………………………………………………………………

Example on Multilevel Hierarchy

```
public virtual class parentclass {
    //how to use inheritance
    //First concept is one parent class and one child class
    //Child class inherits the properties of parent class and have its own Functionality
    protected integer add(integer x,integer y)
    {
        //protected method cannot be called outside
        //proteced method cannot use static
        integer result;
        result=x+y;
        system.debug('show me addition '+result);

        return result; //program exution terminates



    }
    public void disp()
    {
        system.debug('welcome');
    }

}
```
…………………………………………………………………………………….
```
public virtual class childclassA extends parentclass
{


    public void display(integer k,integer z)
    {
        //integer k=90;
        //integer z=100;
        integer result;
        result=add(k,z);



    }

    public integer sub(integer x,integer y)
    {
        //protected method cannot be called outside
        //proteced method cannot use static
        integer result;
        result=x-y;
        system.debug('show me subtraction '+result);
        return result; //program exution terminates


    }


}
```

……………………………………………………………………………………………..

```
public class childclassofA extends childclassA
{

    public void show(integer m,integer n)
   {
       display(m,n);
       sub(m,n);




    }



}
```

…………………………………………………………………………………………………

Example on Interface

public interface interface1

```
{
    //interface keryword is used for multiple inheritance
    //interface is only used for declaration of Methods not for definition of methods
    //no variable declartion is allowed inside interface.
       void tax_admin(); //no public and no static is allowed
   //integer x; //no variable declaration is allowed in interface


}
public interface interface2
{


    //interface keryword is used for multiple inheritance
    //interface is only used for declaration of Methods not for definition of methods
    //no variable declartion is allowed inside interface.
     void tax_It();
    void div();




}
public class childinterface implements interface1,interface2
{
    public static void tax_admin()
    {
        integer x=5;
        integer y=100;
        integer r;
        r=Y*x/100;
        system.debug('show me r'+r);


    }


public static void tax_It()
    {
        integer x=6;
        integer y=200;
        integer r;
        r=Y*x/100;
        system.debug('show me r'+r);


    }
public static void div()
    {
        integer x=6;
        integer y=200;
        integer r;
        r=y/6;
        system.debug('show me r div value'+r);


    }
}
```

Execution

```
childinterface.tax_admin();
childinterface.tax_It();
childinterface.div();
```

……………………………………………………………………………………………………..

Example of Abstraction

```
public abstract class parentabstract

{



    public abstract void cal_sal(integer x,integer y); //user want to use function in abstract class then user have to define that function with abstract keyword

    //wheneever use use abstarct function function ovveride is must

    public integer tax=5;


}

public class childabstract extends parentabstract

{


    public override void cal_sal(integer x,integer y) //ovveride indicates that user use function from abstract class.

    {

        integer result;

        result=x+y-tax;

        system.debug('Show me result '+result);


    }

}
```

……………………………………………………………………………………………………..

Scenario1: Create abstraction class for discount.If discount is allocated to employee it will be 10% if discount is allocated to Customer it will be 5%.Create two child class using abstract class.

…………………………………………………………………………………………….

Example on Sobject

```
public class sclass
{


   public static void disp()
   {
      account ac=[select name from account limit 1]; //specific
      system.debug(ac.name);

      sobject s=[select firstname,lastname from lead limit 1];//generic
      system.debug(s);
      //system.debug(s.firstname);
      //system.debug(s.lastname);

   }

}
```

…………………………………………………………………………………..

Task 1

Function overloading

```
add()
add(integrer x,integer y)
add(integer x,integer y,integer z);
```

……………………………………………………………………………………………………..

Example on constructor overloading

```
public class cclass
{
   public cclass()
```

```
    {
        system.debug('This is example of constructor');
    }
    public cclass(integer x)
    {
        x=x+200;
        system.debug('show me value of x'+x);
    }


}
```

……………………………………………………………………………………………

```
cclass obj=new cclass(); // using  aconstructor of aclass
cclass obj1=new cclass(500);
```

…………………………………………………………………………………..

Example on Operator Overloading

```
 public static void add(integer x,integer y)
    {
        system.debug('Show me additions of '+(X+Y));//AN EXAMPLE of operator overload
    }
```

………………………………………………………………………………………………

Example on Property class

```
public class pclass {

    public static integer A{get;set;} //get set value
    public static string  b{get;set;}

    public static integer x; //no get set
    public static integer age
    {
        get
        {
            return x;
        }
        set
        {
            x=value;
        }


    }



}
```

Execution

```
//pclass.a=10;
//system.debug(pclass.a);
//pclass.b='Manisha';
//system.debug(pclass.b);
pclass.age=23;
system.debug(pclass.age);
```