OOPS

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
    pen p1= new pen();
    p1.name="cello";
    p1.tip=5;
    System.out.println(p1.name);
    System.out.println(p1.tip);}
    class pen {// class created
    String name;
    int tip;}



Output: cello
5
```

Setters and getters

```
import java.io.ObjectInputStream.GetField;

public class prac1oops {
    // getters and setters
    public static void main(String args[]){
    pen p1= new pen();
    p1.setname("cello");
    p1.settip(5);
    System.out.println(p1.gettip());
    System.out.println(p1.getname());
    }
}

class pen {// class created
    String name;
    int tip;

    void setname(String newname){
        this.name=newname;
    }

    void settip(int newtip){
        this.tip=newtip;
    }
    String getname(){
         return this.name;
    }
    int gettip(){
         return this.tip;
    }


}
Output:5
cello
```

constructer

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
    pen p1= new pen();

class pen {// class created
    String name;
    int tip;
    pen(){// default constructer
        System.out.println("default pen constructer is called");
    }
Output:
Default constructer is called
```

Parameterized constructer

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
    pen p1= new pen("cello");
    System.out.println(p1.name);

    }
```

```
}

class pen {// class created
    String name;
    int tip;
    pen(String name){// default constructer
        this.name= name;
        System.out.println("parameterized pen constructer is called");
    }
Output:
parameterized pen constructer is called
cello
```

copy constructer:

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
   student s1= new student();
   s1.marks[0]=0;
   s1.marks[1]=1;
   s1.marks[2]=2;
   student s2=new student(s1);
   for(int i=0;i<3;i++){
    System.out.println(s2.marks[i]);
   }

    }
}

class student{
    String name;
    int marks[];
    student(){
        marks= new int[3];
    }
    student(student s1){
        marks=new int [3];
        this.marks=s1.marks;

    }
}
Output:012
```

Inheritance

Single level

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
   fish f1=new fish();
   f1.eats();
   f1.breath();
   f1.swims();
    }
}

class animal{
    void eats(){
        System.out.println("animal eats");
    }
    void breath(){
        System.out.println("animal breath");
    }
}
class fish extends animal{
    void swims(){
        System.out.println("fish swims");
    }
}
Output:
animal eats
animal breath
fish swims
```

multi-level

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
   dog f1=new dog();
   f1.eats();
   f1.emotions();
   f1.barks();
    }
}

class animal{
    void eats(){
        System.out.println("animal eats");
    }

}
class mammal extends animal{
    void emotions(){
        System.out.println("man contains emotions");
    }
}
class dog extends mammal{
    void barks(){
        System.out.println("dog barks");
    }

}
Output:
animal eats
man contains emotions
dog barks
```

Heirarchial

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
   bird f1=new bird();
   f1.eats();
   f1.fly();
   mammal m1=new mammal();
   m1.eats();
   m1.emotions();

    }
}

class animal{// parent class
    void eats(){
        System.out.println("animal eats");
    }
}
class mammal extends animal{//derrived class
    void emotions(){
        System.out.println("man contains emotions");
    }
}
class bird extends animal{//derrived class
    void fly(){
        System.out.println("bird fly");
    }

}
Output:
animal eats
bird fly
animal eats
man contains emotions
```

hybrid inheritance

```
public class prac1oops {
    // simple class and object creation
    public static void main(String args[]){
  peackok f1=new peackok();
   f1.eats();
   f1.fly();
   f1.feathers();
   fish m1=new fish();
   m1.eats();
   m1.emotions();
   m1.gills();

    }
}

class animal{// parent class
    void eats(){
        System.out.println("animal eats");
    }
}
class mammal extends animal{//derrived class
    void emotions(){
        System.out.println("man contains emotions");
    }
}
class bird extends animal{//derrived class
    void fly(){
        System.out.println("bird fly");
    }

}
class fish extends mammal{
    void gills(){
        System.out.println("fish contains gills");
    }

}
class peackok extends bird{
     void feathers(){
        System.out.println("peackock has feathers");
     }
}
Output:
animal eats
bird fly
peackock has feathers
animal eats
man contains emotions
fish contains gills
```

multiple inheritance : is not possible by using classes but possible by using interfaces

polymorphism:

→ method overloading

→method overriding

Method overloading:

```
public class prac1oops {
    public static void main(String args[]){
        Student s1=new Student();
       int f1= s1.sum(1,2);
       System.out.println(f1);
```

```
     System.out.println(s1.sum(1.3,1.5));

    }
}
class Student{
    String name;
    int sum;
    int sum( int a,int b){
        return a+b;
    }
    double sum(double a,double b){
        return a+b;

    }
    float sum(int a, int b,int c){
        return a+b+c;
    }
}
Output:
3
2.8
```

Method overriding:

```
public class prac1oops {
    public static void main(String args[]){
        dog d1= new dog();
        d1.print();

    }
}
class animal{
    String name;
    void print(){
        System.out.println("this is animal class");
    }
}
class dog extends animal{
    void print(){
        System.out.println("this is dog class");
    }
}
Output:
this is dog class
```

abstract class

```
public class prac1oops {
    public static void main(String args[]){
        dog d1= new dog();
        d1.eat();
        d1.walks();
        cat c1= new cat();
        c1.eat();
        c1.walks();



    }
}
 abstract class animal{

    void eat(){
        System.out.println(" animal eats");
    }
    abstract void walks();
}
class dog extends animal{
    void walks(){
        System.out.println("dog walks");
    }
}
class cat extends animal{
    void walks(){
        System.out.println("cat walks");
    }
}
Output:
animal eats
dog walks
animal eats
cat walks
```

interface

```
public class prac1oops {
    public static void main(String args[]){
        queen q1=new queen();
        q1.moves();
    }
}
interface chessplayer{
    void moves();
}
class queen implements chessplayer{
    public void moves(){
        System.out.println("queen moves");
    }
}
class king implements chessplayer{
    public void moves(){
        System.out.println("king moves");
    }
}
Output:
queen moves
```

Multiple -inheritance using interface

```
public class prac1oops {
    public static void main(String args[]){
       bear b1= new bear();
       b1.eatflesh();
       b1.eatplant();
 }
}
interface herbivore{
    void eatplant();
}
interface carnivore{
    void eatflesh();
}
class bear implements herbivore,carnivore{
    public void eatflesh(){
        System.out.println("beer eat flesh");
    }
    public void eatplant(){
        System.out.println("beer eat plant");
    }
}
beer eat flesh
beer eat plant
```

static keyword:

```
public class prac1oops {
    public static void main(String args[]){
      student s1= new student();
      s1.schoolname="sai";
      student s2= new student();
      System.out.println(s2.schoolname);

    }
}
class student{
    String name;
    int roll;
    static String schoolname;

void setname(String name){
    this.name=name;
}
String getname(){
    return this.name;
}}
Output: sai
```

Super keyword:

```
public class prac1oops {
    public static void main(String args[]){
      student s1= new sai();
      System.out.println(s1.name);

    }
}
class student{
    String name;

    student(){
        System.out.println("student constructer is called");
    }
}
class sai extends student{

    sai(){
        super.name="hello";
        System.out.println(" sai constructer is called");
    }
    }

output:

student constructer is called

sai constructer is called

hello
```

QUESTION:
Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by the user

```
public class prac1oops {
    public static void main(String args[]){
        complex c= new complex(3,5);// first input
        complex d= new complex(2, 6);//second input
        complex e=complex.add(c,d);
        complex f= complex.diff(c, d);
        complex g= complex.product(c, d);

        e.print();
        f.print();
        g.print();
    }
}
class complex{
    int real;
    int imag;
    complex(int r,int i ){
        real=r;
        imag=i;
    }

    public static complex add(complex a, complex b){
        return new complex((a.real+b.real),(a.imag+b.imag));
    }
    public static complex diff(complex a, complex b){
        return new complex((a.real-b.real),(a.imag-b.imag));
    }

    public static complex product(complex a, complex b){
        return new complex(((a.real*b.real)-
(a.imag*b.imag)),((a.real*b.imag)+(a.imag+b.real)));
    }

    public void print(){
        if(real==0  &&  imag!=0){
            System.out.println(imag+"i");
        }
        else if(imag==0 && real!=0){
            System.out.println(real);

        }
        else{
            System.out.println(real+"+"+imag+"i");
        }
    }

}
Output:
5+11i
1+-1i
-24+25i
```