1.Factorial (oops).

public class factorial {  
  
 void fact(int n){  
 int fact=1;  
 for(int i=1;i<=n;i++){  
 fact=fact\*i;  
 }  
 System.*out*.println(fact);  
 }  
  
 public static void main(String[] args) {  
 factorial f=new factorial();  
 f.fact(5);  
  
 }  
  
}

Output.



2.Area of circle (oops).

a)with passing parameters to constructor.

public class circle {  
  
 int r;  
 circle(int a){  
  
 r=a;  
 }  
 void cir(){  
 double area=3.14\*r\*r;  
 System.*out*.println("area of circle = "+area);  
 }  
  
 public static void main(String[] args) {  
 circle c=new circle(3);  
 c.cir();  
 }  
}

output.



b)without passing parameters in constructor.

public class over {  
  
 int r;  
 over(){  
  
  
 }  
 void cir(int r){  
 double area=3.14\*r\*r;  
 System.*out*.println("area of circle = "+area);  
 }  
  
 public static void main(String[] args) {  
 over c=new over();  
 c.cir(3);  
 }  
}

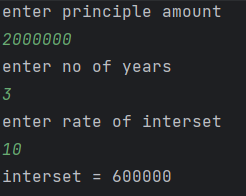
output.



3.Simple interest (oops).

import java.util.\*;  
public class roi {  
 int p,y,r;  
 roi(int a,int b,int c){  
 p=a;  
 y=b;  
 r=c;  
 }  
 void interset(){  
 int roi=p\*y\*r/100;  
 System.*out*.println("interest = "+roi);  
 }  
 public static void main(String[] args) {  
 int p,y,r;  
 Scanner s=new Scanner(System.*in*);  
 System.*out*.println("enter principle amount ");  
 p=s.nextInt();  
 System.*out*.println("enter no of years ");  
 y=s.nextInt();  
 System.*out*.println("enter rate of interest ");  
 r=s.nextInt();  
 roi r1=new roi(p,y,r);  
 r1.interset();  
  
 }  
}

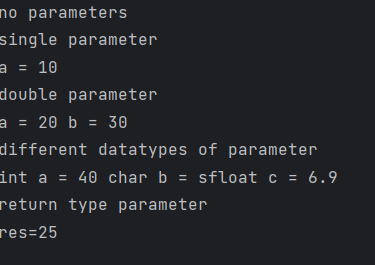
output.



4.Polymorphism.

public class polymorphism {  
 void test(){  
 System.*out*.println("no parameters");  
 }  
 void tes(int a){  
 System.*out*.println("single parameter");  
 System.*out*.println("a = "+a);  
 }  
 void test(int a,int b){  
 System.*out*.println("double parameter");  
 System.*out*.println("a = "+a+" b = "+b);  
 }  
 void test(int a,char b,float c){  
 System.*out*.println("different datatypes of parameter");  
 System.*out*.println("int a = "+a+" char b = "+b+"float c = "+c);  
 }  
 int test(int a){  
 return a\*a;  
 }  
  
 public static void main(String[] args) {  
 polymorphism p=new polymorphism();  
 p.test();  
 p.tes(10);  
 p.test(20,30);  
 p.test(40,'s',6.9f);  
 int res=p.test(5);  
 System.*out*.println("return type parameter ");  
 System.*out*.println("res="+res);  
 }  
}

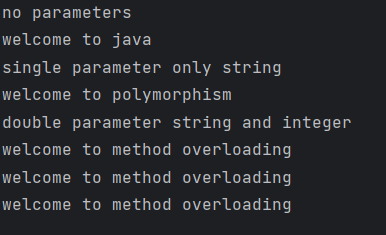
output.



5.Overloading.

public class polyover {  
 void test(){  
 System.*out*.println("no parameters");  
 System.*out*.println("welcome to java");  
 }  
 void tes(String a){  
 System.*out*.println("single parameter only string");  
 System.*out*.println(a);  
 }  
 void test(String a,int b){  
 System.*out*.println("double parameter string and integer");  
 for(int i=0;i<b;i++) {  
 System.*out*.println(a);  
 }  
 }  
  
 public static void main(String[] args) {  
 polyover p=new polyover();  
 p.test();  
 p.tes("welcome to polymorphism");  
 p.test("welcome to method overloading",3);  
 }  
}

output.



6.Inheritance.

public class inheritance {  
 protected double area;  
  
 void circle(int r){  
 area=3.14\*r\*r;  
 }  
  
}  
class cylinder extends inheritance{  
 private double vol;  
  
 void cal1(int h){  
 vol=area\*h;  
 }  
 void diplay(){  
 System.*out*.println("area of circle = "+area);  
 System.*out*.println("volume of cylinder = "+vol);  
 }  
 public static void main(String[] args) {  
 cylinder c=new cylinder();  
 c.circle(5);  
 c.cal1(10);  
 c.diplay();  
 }  
 }

output.

