# LAST ASSIGNMENT

Abdul Wahid Alias Kashif Ali

BSCS-II(H)

Lecture 19

```
import java.sql.*;

public class jdbc1{
    public static void main(String args[]){
            String url=("jdbc:mysql://localhost:3306/students");
            String user="root";
            String password="2006";
    try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con=DriverManager.getConnection(url,user,password);
            Statement stmt=con.createStatement();
            ResultSet result=stmt.executeQuery("select * from student_info;");
            while(result.next()){
                        System.out.println(result.getInt("id"));
                        System.out.println(result.getString("name"));
                        System.out.println(result.getInt("age"));
                        System.out.println(result.getString("department"));

            }
            stmt.close();
            con.close();
            result.close();
                }
            catch(Exception e){
                e.printStackTrace();

                                }



                }

    }
```

This code is about connecting jdbc through the sql bridge . furthermore , this code gets connection through url, user and password. And executes the query of mysql.
The output is shown below:

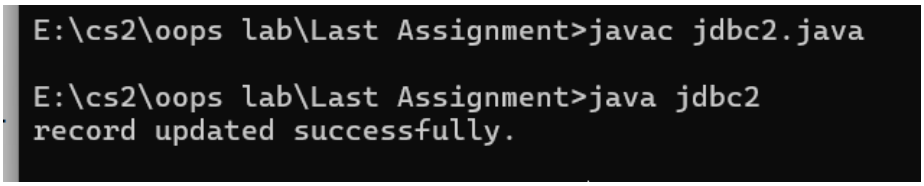
```
E:\cs2\oops lab\Last Assignment>java jdbc1
1
kashif
20
Computer Science
```

```
import java.sql.*;
```

```
public class jdbc2{
                    public static void main(String args[]){
                        String url=("jdbc:mysql://localhost:3306/students");
                        String user="root";
                        String password="2006";
                    try{
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        Connection con=DriverManager.getConnection(url,user,password);
                        Statement stmt=con.createStatement();
                        int a=stmt.executeUpdate("update student_info set name='Abdul
Wahid' where id=1");
                        if( a>=1)
                                System.out.println("record updated successfully.");
                        else
                                System.out.println("record not found");
                        stmt.close();
                        con.close();

                                }
                        catch(Exception e){
                                e.printStackTrace();

                                        }
```

```
        }

    }
```

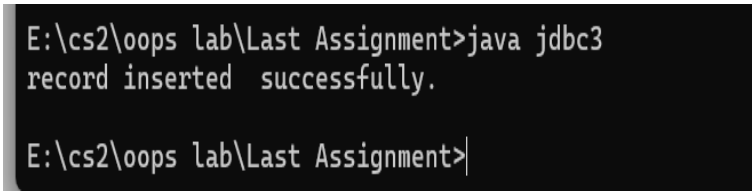
This is same code as above but here we are executing update which returns integer that's why we are storing in integer and other components are same as above first code.the output is provided below.




```
import java.sql.*;


public class jdbc3{
    public static void main(String args[]){
        String url=("jdbc:mysql://localhost:3306/students");
        String user="root";
        String password="2006";
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con=DriverManager.getConnection(url,user,password);
        Statement stmt=con.createStatement();
        int a=stmt.executeUpdate("insert into
student_info(id,name,age,department) values(2,'kashif Ali',19,'camputer sens');");
        if( a>=1)
            System.out.println("record inserted  successfully.");
        else
            System.out.println("record not found");
```

```
                stmt.close();
                con.close();

                    }
                catch(Exception e){
                        e.printStackTrace();

                                    }




                        }

            }
```

This code is same as second one but here we are inserting the record which works same as the second one . the out put is given below:




```
                        import java.sql.*;


public class jdbc4{
                        public static void main(String args[]){
                        String url=("jdbc:mysql://localhost:3306/students");
                        String user="root";
                        String password="2006";
```

```
try{
Class.forName("com.mysql.cj.jdbc.Driver");
Connection
con=DriverManager.getConnection(url,user,password);
Statement stmt=con.createStatement();
int a=stmt.executeUpdate("delete from student_info where
id=2");
if( a>=1)
        System.out.println("record deleted  successfully.");
else
        System.out.println("record not found");
stmt.close();
con.close();

    }
catch(Exception e){
        e.printStackTrace();

                }




        }

}
```

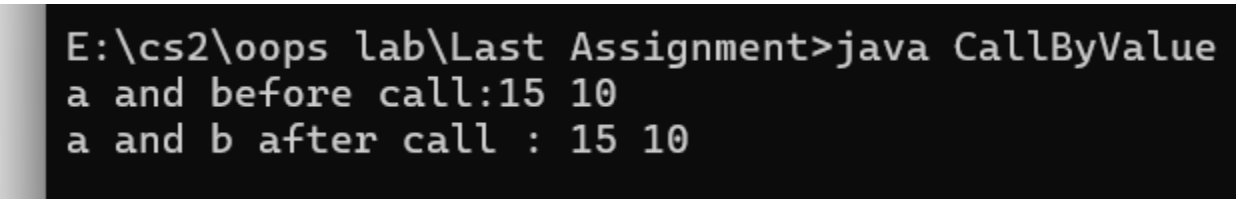
This just deletes the record by executing the update in same way as above programs .the output is given below:

```
E:\cs2\oops lab\Last Assignment>javac jdbc4.java

E:\cs2\oops lab\Last Assignment>java jdbc4
record deleted  successfully.
```

Lecture 20

```
class Test{
void meth(int i , int j ){
i*=2;
j+=2;
}
}

class CallByValue{
public static void main(String args[]){
Test ob= new Test();
int a=15,b=10;
System.out.println("a and before call:" +a +" "+ b);
ob.meth(a,b);
System.out.println("a and b after call : " + a + " "+ b);
}
}
```

This code is contain an example of call by value , when we are putting two integers into the meth so it is working with the values of the integer we have provided and it is not affection the reference of the integers.




```
class Test{
int a,b;
Test(int i , int j ){
 a=i;
 b=j;
 }
 void meth(Test o){
 o.a*=2;
 o.b*=2;
}
 }


class CallByRef{
 public static void main(String args[]){
```

```
        Test ob= new Test(12,12);
        int a=15,b=10;
        System.out.println("ob.a and ob.b before
call:" +ob.a +" "+ ob.b);
        ob.meth(ob);
        System.out.println("ob.a and ob.b after
call : " + ob.a + " "+ ob.b);
    }
    }
```

This is an example of call by reference in which the operations on the variables changes the value.Here first two number were sent to the constructer of Test and object was created and later using the same object the variables were accessed with changed value.

Given output proves that:

```
E:\cs2\oops lab\Last Assignment>javac CallByRef.java

E:\cs2\oops lab\Last Assignment>java CallByRef
ob.a and ob.b before call:12 12
ob.a and ob.b after call : 24 24
```

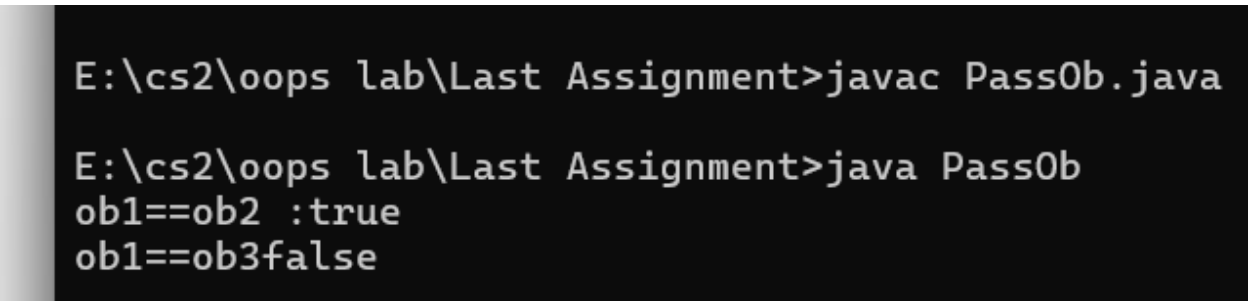
```
class Test{
int a,b;
Test(int i , int j){
    a=i;
```

```
        b=j; class Test{
    int a,b;
    Test(int i , int j){
        a=i;
        b=j;
        }
    Boolean equals(Test o){
            if(o.a==a&&o.b==b)
                return true;
            else
                return false;
            }
        }
class PassOb{
            public static void main(String args[]){
                Test ob1= new Test(100,22);
                Test ob2= new Test(100,22);
                Test ob3= new Test(-1,-1);

                System.out.println("ob1==ob2
:"+ ob1.equals(ob2));
```

```
                                        System.out.println("ob1==ob3"
+ ob1.equals(ob3));
                                        }
                                }
                                }
                        Boolean equals(Test o){
                                if(o.a==a&&o.b==b)
                                        return true;
                                else
                                        return false;
                                }
                        }
class PassOb{
                        public static void main(String args[]){
                                Test ob1= new Test(100,22);
                                Test ob2= new Test(100,22);
                                Test ob3= new Test(-1,-1);

                                System.out.println("ob1==ob2
:"+ ob1.equals(ob2));
                                System.out.println("ob1==ob3"
+ ob1.equals(ob3));
```

```
    }
}
```

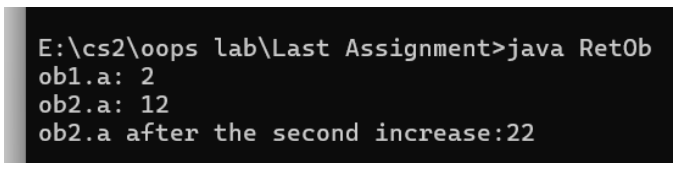
Here 3 different objects of test class are being made each's variables storing values passed through parameterized constructer and then in last there values are being compared using equals method.following is the output:




```
class Test {
    int a;
    Test (int i){
        a=i;
    }
    Test incrByTen(){
        Test temp= new Test(a+10);
            return temp;

    }
}
class RetOb{
```

```
    public static void main (String args[]){
        Test ob1= new Test(2);
        Test ob2;

        ob2= ob1.incrByTen();
        System.out.println("ob1.a: "+ ob1.a);
        System.out.println("ob2.a: "+ ob2.a);

        ob2=ob2.incrByTen();
        System.out.println("ob2.a after the
second increase:" + ob2.a);
    }
}
```

ob1 is the object of Test created in RetOb class to a parameterized constructer of integer where the instance the variable of Test Class is being initialized and then 2nd object is being made of Test class which holds the value returned from incrByten function of Test class reached through first object which returns the object of same Test class where the value is being increased by ten .so here the object is being returned.This should have following output:



```
class Overload {
	public static void main(String args[]){
		OverloadDemo ob= new OverloadDemo();
		double result;

		ob.test();
		ob.test(10);
		ob.test(10,20);
		result= ob.test(123.25);
		System.out.println("result of ob.test(123.24) :"+ result);
	}
}
class OverloadDemo {
	void test(){
		System.out.println("NO parameters:");
		}
	void test (int a){
		System.out.println("a:"+ a) ;}
	void test(int a, int b){
		System.out.println("a  and b: " + a+ " " +b);
		}
	double test(double a){
		System.out.println("double a : " + a);
		return a*a;
			}
	}
```

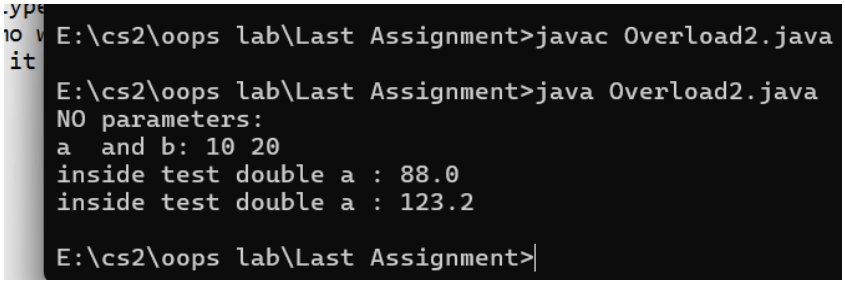
In this code function overloading is being carried out by varying parameters . The Overload class calls methods from Overload demo through various parameters. Following is the output of the program:

```
E:\cs2\oops lab\Last Assignment>java Overload
NO parameters:
a:10
a  and b: 10 20
double a : 123.25
result of ob.test(123.24) :15190.5625
```

```
class Overload2 {
                    public static void main(String args[]){
                            OverloadDemo ob= new OverloadDemo();
                            int i=88;

                            ob.test();

                            ob.test(10,20);
                            ob.test(i);
                            ob.test(123.2);
                        }
                    }
class OverloadDemo {
                    void test(){
                        System.out.println("NO parameters:");
                        }

                    void test(int a, int b){
                        System.out.println("a  and b: " + a+ " " +b);
                        }
                    double test(double a){
                        System.out.println("inside test double a : " + a);
                        return a*a;
                            }
                    }
```

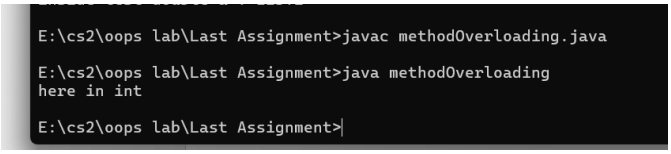
the code is same as previous code but here is slight changing such as automatic type promotion , it happened because while calling funtions in OverloadDemo we entered one

with taking integer but there wasn't any methods defined So it was automatically promoted to the function of double .




```
public class methodOverloading {
        public static void main(String args[]){
                methodOverload obj=new methodOverload();
                obj.call(1);
                }

        }
class methodOverload {
        public void call (int i){
                System.out.println("here in int");
                        }
        public void call(byte i){
                System.out.println("here in byte");
                }
        }
```

Here 1 is passed into methods and it is regarded as integer . The output will be as following:



```
public class methodOverloading1 {
                                                    public static void
main(String args[]){
                                                methodOverload obj=new
methodOverload();
                                                        byte a=1;
                                                        obj.call(a);
                                                        }

                                                    }
class methodOverload {
                                                    public void call (int
i){
                                                System.out.println("here
in int");
                                                            }
                                                    public void call(byte
i){
                                                System.out.println("here
in byte");
                                                        }
                                                    }
```

Here we are passing the byte value to call the method  of byte . so It should take the method of byte show following output.

```
PS E:\cs2\oops lab\Last Assignment> javac methodOverloading1.java
PS E:\cs2\oops lab\Last Assignment> java methodOverloading1
here in byte
```

```
class Box{
        double width;
        double  height;
        double depth;

        Box(double w , double h , double d){
                width=w;
                height=h;
                depth=d;
        }
        Box(){
                width=-1;
                height=-1;
                depth=-1;
        }
        Box(double len){
                width=height=depth=len;
        }
double volume (){
        return width*height*depth;
        }
}
class OverloadsCons{
        public static void main(String args[]){
                Box mybox1= new Box(10,20,15);
                Box mybox2= new Box();
                Box mycube= new Box(7);
                double vol;
```

```
                vol=mybox1.volume();
                System.out.println("volume of Mybox1 is : "+ vol);
                vol=mybox2.volume();
                System.out.println("volume of Mybox2 is : "+ vol);
                vol=mycube.volume();
                System.out.println("volume of Mybox2 is : "+ vol);
        }
    }
```

This code is about overloading constructer where three type of constructers are being used each one of these are called according to need like for my cube we are using constructer with one parameter . Hence the constructer overloading occurs here. The following is the output observed:

```
PS E:\cs2\oops lab\Last Assignment> javac OverloadsCons.java
PS E:\cs2\oops lab\Last Assignment> java OverloadsCons
volume of Mybox1 is : 3000.0
volume of Mybox2 is : -1.0
volume of Mybox2 is : 343.0
```

```
public class constructer {
    public static void main(String args[]){
        Box obj= new Box();
    }
}
class Box {
    Box(){
        this("A");
        }
    Box(String message){
        this(message,"B");
            }
    Box(String message1, String message2){
            System.out.println(message1 +"--"+ message2);
        }
```

```
}
```

Here through the main class the parameterless constructer is being called and in the that constructer other constructer with one parameter is being called and this goes further . in brief, this keyword helps us to call the constructer with in class . the following output was observed:

```
PS E:\cs2\oops lab\Last Assignment> javac constructer.java
PS E:\cs2\oops lab\Last Assignment> java constructer
A--B
```

```
public class example{
        public static void printNumbers(int value1, int value2, int value3,int...numbers){
            System.out.println("numbers received :");
            System.out.println("value 1: "+ value1);
            System.out.println("value 2: "+ value2);
            System.out.println("value 3: "+ value3);
            for(int number:numbers)
                System.out.println(number);


        }
    public static void main(String []args){
            int [] array={10,20,30};
            printNumbers(1,2,3,array);

            }
        }
```

In this code when we passing variable arguments it is working like an array .the following is the output:

```
PS E:\cs2\oops lab\Last Assignment> javac example.java
PS E:\cs2\oops lab\Last Assignment> java example
numbers received :
value 1: 1
value 2: 2
value 3: 3
10
20
30
```

```
public class example{
        public static void printNumbers(String studentName,int...numbers){
            System.out.println("Numbers of "+studentName +"  are:");

            for(int number:numbers)
                System.out.println(number);



            }
    public static void main(String []args){
            int [] array={10,20,30};
            printNumbers("john",array);

            }
            }
```

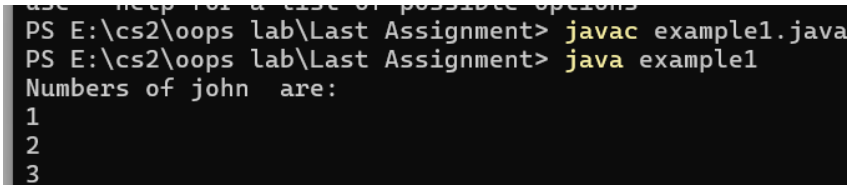
This is same code as prious one but here instead of numbers we are passing string and array through variable arguments.

```
PS E:\cs2\oops lab\Last Assignment> javac example.java
PS E:\cs2\oops lab\Last Assignment> java example
Numbers of john  are:
10
20
30
```

```
  public class example1{
        public static void printNumbers(String studentName,int...numbers){
            System.out.println("Numbers of "+studentName +"  are:");

            for(int number:numbers)
                System.out.println(number);
```

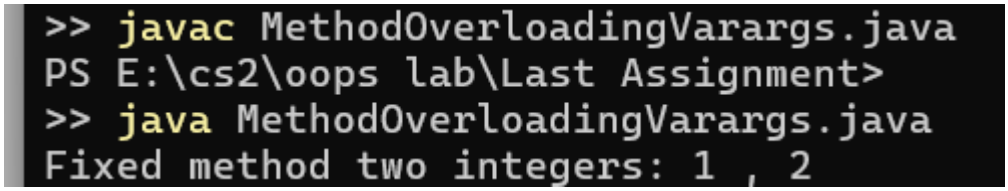
```
    }
public static void main(String []args){
        int [] array={10,20,30};
        printNumbers("john",1,2,3);

    }
}
```

Here instead of passing array we are passing separate number and varargs also accepts this and sets this like an araay. Following is the output:




```
public class MethodOverloadingVarargs{

    public static void printNumbers(int a, int b){
     System.out.println("Fixed method two integers: "+a+" , "+b);
        }
  public static void printNumbers(int... numbers){
      System.out.println("varible arguments with numbers; ");
          for(int num:numbers)
           System.out.print(num + "");
        }

  public static void main(String [] args){
          printNumbers(1,2);
        }
 }
```

In this code we are overloading the method using two arguments method and one varrargs and when we pass two integers into it . It goes for the exact match with is method with two integer . following output further solidifies this:




```
public class example2{
    public static void printNumbers(int... numbers){
        for(int number: numbers)
            System.out.println(number);

    }
    public static void printNumbers(int [] numbers){
        System.out.println("i am the method of array");
    }
    public static void main(String args[]){
        int []numbers={1,2,3};
        printNumbers(numbers);
    }
}
```

Here the compiler is not allowing us to declara the array method and varargs method in a class because varargs works same as integer so it given the following error:

```
PS E:\cs2\oops lab\Last Assignment> javac example2.java
example2.java:7: error: cannot declare both printNumbers(int[]) and printNumbers(int...) in example2
    public static void printNumbers(int [] numbers){
                       ^
1 error
```

# Lecture 22

```
class A{
    int i,j;
    void showij(){
      System.out.println("i and j:"+ i+"  "+ j);
      }
```

```
                                        }
class B extends A{
                                          int k;
                                          void showk(){
                                                  System.out.println("k: "+k );
                                                          }
                                          void sum(){
                                                  System.out.println("i+j+k :"+(i+j+k));
                                        }
}
class SimpleInheritance{
                                          public static void main(String ars[]){
                                                  B subOb=new B();
                                                  subOb.i=7;
                                                  subOb.j=8;
                                                  subOb.k=9;
                                                  System.out.println("contensts of subob: ");
                                                  subOb.showij();
                                                  subOb.showk();
                                                  System.out.println();
                                                  System.out.println("sum of i,j,k in subOb: ");
                                                  subOb.sum();
                                                  }
                                          }
```

Here through the main class we make object of B class which is child of A and through this object we can inherit the method of A. The following is the output of this:

```
PS E:\cs2\oops lab\Last Assignment> javac SimpleInheritance.java
PS E:\cs2\oops lab\Last Assignment> java SimpelInheritance
Error: Could not find or load main class SimpelInheritance
Caused by: java.lang.ClassNotFoundException: SimpelInheritance
PS E:\cs2\oops lab\Last Assignment> java SimpleInheritance
contensts of subob:
i and j:7  8
k: 9

sum of i,j,k in subOb:
i+j+k :24
```

```
class A{
```

```
                    A () {
                      System.out.println("inside A consctructer  ");
                      }
                    }
class B extends A{
                    B(){
                      System.out.println("inside B consctructer  ");
                      }
                    }
class C extends B{
                    C(){
                      System.out.println("inside c consctructer  ");
                      }
                    }
class CallingCons{
                      public static void main(String args[]){
                                                C c= new C();
                            }
                      }
```

In this when we called the parameterless constructer of the child class it invokes the paramterless constructer of parent class .

The output is shown below:

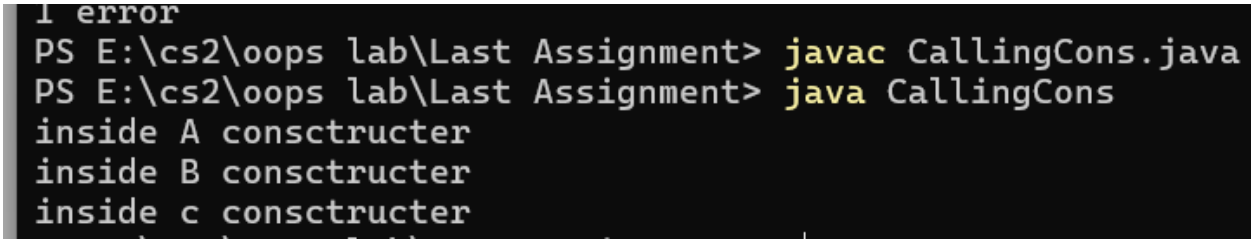
```
PS E:\cs2\oops lab\Last Assignment> javac CallingCons.java
PS E:\cs2\oops lab\Last Assignment> java CallingCons
inside A consctructer
inside B consctructer
inside c consctructer
```

# Lecture 23

```
class A{
                    A () {
                      super();
```

```
    System.out.println("inside A consctructer  ");
    }
  }
class B extends A{

  B(){
    super();
    System.out.println("inside B consctructer  ");
    }
  }
class C extends B{

  C(){
    super();
    System.out.println("inside c consctructer  ");
    }
  }
class CallingCons{
    public static void main(String args[]){
                C c= new C();
        }
    }
```

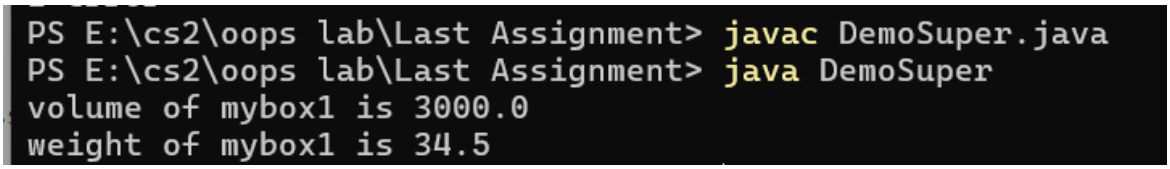
When we are using super keyword it works same as default :




```
class Box{
private double width;
private double height;
private depth;
Box(double w, double h, double d){
```

```
                    width=w;
                    height=h;
                    depth=d;
                    }
        Box(){
            width=-1;
            height=-1;
            depth=-1;
            }
        double volume(){
                    return width*height*depth;
            }
        }
class Boxweight extends Box{
            double weight;
            BoxWeight(double w , double h, double d, double m){
                        super(w,h,d);
                        weight=m;
                        }
            BoxWeight(){
                    super();
                        weight=-1;
                    }
        }
class DemoSuper {
            public static void main (String args[]){
                        BoxWeight mybox1= new BoxWeight(10,20,15,34.5);
                        double vol;
                        vol=mubox1.volume();
                        System.out.println("volume of mybox1 is "+ vol);
                        System.out.println(weight of mybox1 is " + mybox1.weight);
                    }
            }
```

Here we are making an object of the child class where we pass four values and from these four the child use 3 to invoke the super class constructer through super keywrods: following is the output:




```
class Box{
protected double width;
protected double height;
protected double depth;

double volume(){
        return width*height*depth;
    }
}
class BoxWeight extends Box{
    double weight;
    BoxWeight(double w , double h, double d, double m){
        super.width=w;
        super.height=h;
        super.depth=d;

        weight=m;
        System.out.println(super.volume());
        }

}
public class superUse{
    public static void main(String args[]){
        BoxWeight ob= new BoxWeight(10,20,15,34.3);
    }
    }
```

Here first we are making the object of child and passing four values and in the child class we are initializing the value to parent class and finally printin the method of parent class into child class: following is the output:

class Box{

```
        private double width;
        private double height;
        private double depth;

        Box(double w, double h, double d){
                width=w;
                height=h;
                depth=d;
                }
        double volume(){
                return width*height*depth;
        }
}
class BoxWeight extends Box{
        double weight;
        BoxWeight(double w , double h, double d, double m){
                super(w,h,d);

                weight=m;
                }

}
class Shipment extends BoxWeight{
                double cost;
                Shipment(double w, double h, double d, double m,double c){
                        super(w,h,d,m);
                        cost=c;
                                }
        }
public class multiLevel{
        public static void main(String args[]){
                Shipment shipment1= new Shipment(10,20,30,40,50);
```

```
                System.out.println(shipment1.volume());
                System.out.println("weight of the shipment1 is :"+shipment1.weight);
                System.out.println("shipping cost: $" +shipment1.cost);
        }
        }
```

This code is about how through the object of child class we can access the parent class using super keywords which envokes parents constructers.Following is the output of the program:

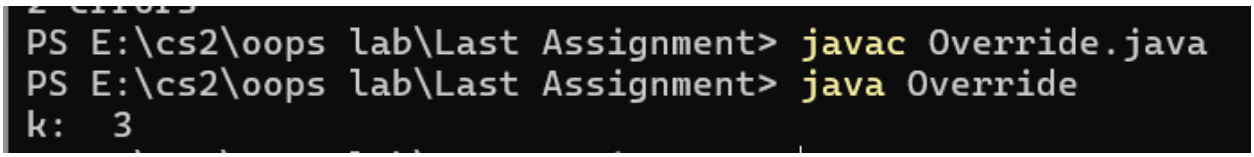
```
PS E:\cs2\oops lab\Last Assignment> javac multiLevel.java
PS E:\cs2\oops lab\Last Assignment> java multiLevel
6000.0
weight of the shipment1 is :40.0
shipping cost: $50.0
```

# LECTURE 24

```
class A {
        int i,j;
        A(int a,int b){
                i=a;
                j=b;
                        }
        void show(){
                System.out.println("i and j: "+ i+ " "+j);
                }
        }
class B extends A {
                int k;
                B(int a , int b, int c){
                        super(a,b);
                        k=c;
                }
                void show(){
                        System.out.println("k:  "+ k);
                        }
                }
```

```
class Override {
                                        public static void main(String args[]){
                                                B subOb= new B(1,2,3);
                                                subOb.show();
                                                }
                                        }
```

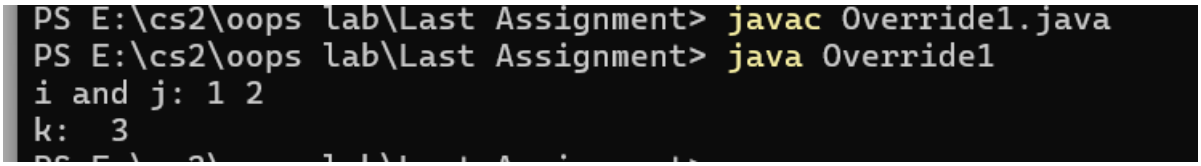
Here we have overrided the method and envoked the method the in child class . following is the output :




```
                                        class A {
                                        int i,j;
                                        A(int a,int b){
                                                i=a;
                                                j=b;
                                                        }
                                        void show(){
                                                System.out.println("i and j: "+ i+ " "+j);
                                                }
                                        }
class B extends A {
                                                int k;
                                                B(int a , int b, int c){
                                                        super(a,b);
                                                        k=c;
                                                }
                                                void show(){
                                                        super.show();
                                                        System.out.println("k:  "+ k);
                                                        }
                                                }
class Override1 {
                                                public static void main(String args[]){
                                                        B subOb= new B(1,2,3);
```

```
        subOb.show();
    }
}
```

This is same as previous code but we have added new thing which is we called the method of parent class in the method of child class . following is the output:

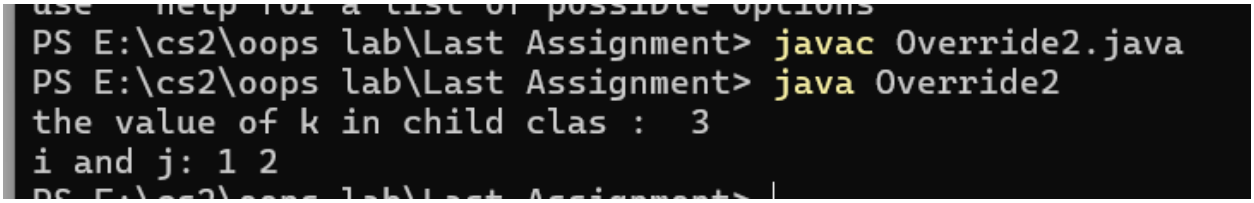



```
class A {
    int i,j;
    A(int a,int b){
        i=a;
        j=b;
    }
    void show(){
        System.out.println("i and j: "+ i+ " "+j);
    }
}
class B extends A {
    int k;
    B(int a , int b, int c){
        super(a,b);
        k=c;
    }
    void show(String message){

        System.out.println(message +k);
    }
}
class Override2 {
    public static void main(String args[]){
        B subOb= new B(1,2,3);
        subOb.show("the value of k in child clas :  ");
        subOb.show();
```

```
    }
    }
```

Here we have just altered the parameters of the constructer that's why we are observing function overloading instead of overriding

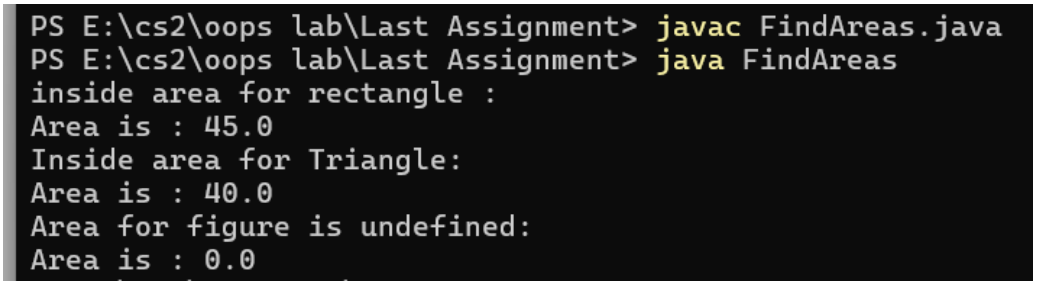
Following is the output of the program:




```
class Figure {
double dim1;
double dim2;
Figure(double a , double b){
    dim1=a ;
    dim2=b;
        }
double area(){
    System.out.println("Area for figure is undefined:");
    return 0;
        }
}
class Triangle extends Figure {
    Triangle (double a, double b){
        super(a,b);
        }
    double area (){
        System.out.println("Inside area for Triangle: ");
            return dim1*dim2/2;
            }
        }
class Rectangle extends Figure {
    Rectangle (double a , double b){
```

```
                    super (a,b);
                    }
    double area (){
        System.out.println("inside area for rectangle : ");
        return dim1*dim2;
                    }
        }
class FindAreas{
    public static void main(String args[]){
        Figure f= new Figure (10,10);
        Rectangle r= new Rectangle (9,5);
        Triangle t= new Triangle (10,8);
        Figure figref;
        figref=r;
        System.out.println("Area is : "+ figref.area());
        figref=t;
        System.out.println("Area is : "+ figref.area());
        figref=f;
        System.out.println("Area is : "+ figref.area());
        }
    }
```

Here we are using dynamic method dispatch , in this we call the method at the run time . Here we have created objects and the run time the parent class run the method of child class which is overridden . Here is the output of the program:




# LECTURE 25

```
class Test{
        int a;
        public int b;
```

```
    private int c;


    void setc(int i){
     c=i;
                    }
    int getc(){
     return c;
            }
    }

class AccessTest{
     public static void main(String args[]){
            Test ob= new Test();
            ob.a=10;
            ob.b=20;
            ob.setc(100);
            System.out.println("a ,b and c :"+ ob.a+" "+ob.b+" "+ob.getc());
            }
    }
```

Here we are getting a and b in the main class as they are accessible according to their modifiers.But , the c is private hence we can not access it directly . therefore we are using getter and setter method to set and get the value of private c.following is the output of the program:

```
PS E:\cs2\oops lab\Last Assignment> javac AccessTest.java
PS E:\cs2\oops lab\Last Assignment> java AccessTest
a ,b and c :10 20 100
```

```
class User {
    private int password;

    public void setPassword(int password){
            this.password=encryptPassword(password);
        }
    public int getPassword(){
        return decryptPassword(password);
            }
```

```
                private int encryptPassword(int password){
                        return password+2;
                                }
                private int decryptPassword(int password){
                        return password-2;
                                }
                }
public class LoginTest{
                public static void main(String args[]){
                        User obj= new User();
                        obj.setPassword(2);
                        }
                }
```

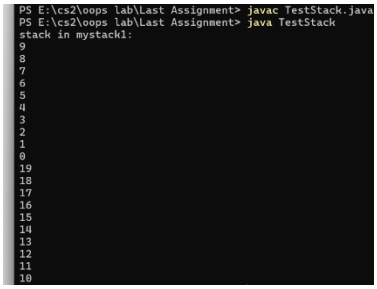
This code is real life example of getting and setting the values to private variables and classes . this denotes how the password encryption works and accessing the private methods and variables.

```
PS E:\cs2\oops lab\Last Assignment> javac LoginTest.java
PS E:\cs2\oops lab\Last Assignment> java LoginTest
```

```
class Stack{
private int stck[]=new int[10];
private int tos;

Stack(){
   tos=-1;
                }
void push(int item){
   if(tos==9)
                System.out.println("stack is full");
   else
                stck[++tos]= item;
                        }
int pop(){
   if(tos<0){
                System.out.println("Stack underflow:");
```

```
                return 0;
                }

        else
                return stck[tos--];
                }
        }
class TestStack{
        public static void main(String args[]){
                Stack mystack1= new Stack();
                Stack mystack2= new Stack();

                for(int i=0;i<10;i++)
                        mystack1.push(i);
                for(int i=10;i<20;i++)
                        mystack2.push(i);

                System.out.println("stack in mystack1: ");
                for(int i=0;i<10;i++)
                        System.out.println(mystack1.pop());
                for(int i=0;i<10;i++)
                        System.out.println(mystack2.pop());
        }
    }
```

Here we are accessing the private elements through push and pop methods .




```
package mypack;
```

```
class Book{
                    String bookname;
                    String auther;
                    Book(String b,String c)
                    {
                       this.bookname=b;
                       this.auther=c;
                       }
                    {
                       public void show()
                    {
                       System.out.println(bookname+ "  "+ auther);
                       }
                    }
```

We are storing this to my package . As we don't have any main method for this so we can't not run it now . but below I am showing how to compile and execute this class.

```
PS E:\cs2\oops lab\Last Assignment> javac mypack/Book.java
PS E:\cs2\oops lab\Last Assignment> java mypack/Book
```

# Lecture 26

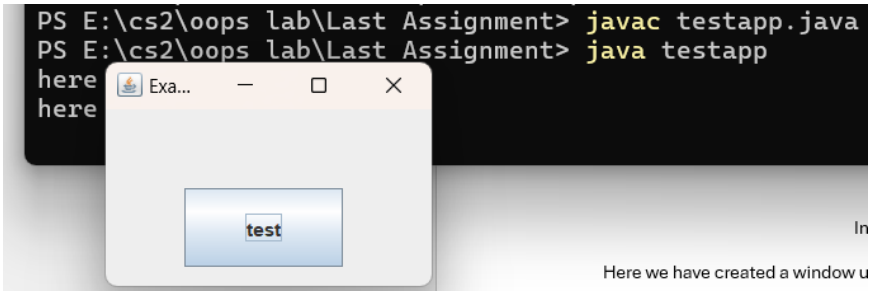
```
import java.awt.event.*;
import javax.swing.*;


public class testapp{
                    public static void main(String args[]){
                              JFrame jfrm= new JFrame ("Example");


                              jfrm.setLayout(null);
                              jfrm.setSize(220,200);
                              jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


                              JButton jbtn= new JButton("test");
                              jbtn.setBounds(50,50,100,50);
                              jbtn.addActionListener(new eventHandler());
```

```
                jfrm.add(jbtn);
                jfrm.setVisible(true);
                }
        }
class eventHandler implements ActionListener{
                public void actionPerformed(ActionEvent ae){
                        System.out.println("here");
                        }
                }
```

In this code we are importing Jframe and jbutton and then using them in our main function.

Here we have created a window using jframe and created a test button using jbutton . we have also added eventlistener to the jbutton

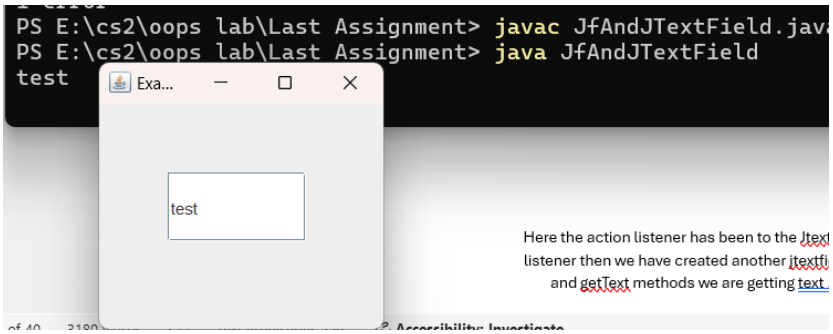
Which works on click or enter press and prints here . this is diagramitically below:




```
import java.awt.event.*;
import javax.swing.*;

public class JfAndJTextField{
        public static void main(String args[]){
                JFrame jfrm= new JFrame ("Example");

                jfrm.setLayout(null);
                jfrm.setSize(220,200);
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                JTextField jtxt= new JTextField("test");
                jtxt.setBounds(50,50,100,50);
        jtxt.addActionListener(new eventHandler());
```

```
        jfrm.add(jtxt);
        jfrm.setVisible(true);
    }
}
class eventHandler implements ActionListener{
    public void actionPerformed(ActionEvent ae){
        JTextField jtxt=(JTextField)ae.getSource();
        System.out.println(jtxt.getText());
    }
}
```

Here the action listener has been to the Jtextfield , when we press enter the event goes to event handler which is implementing action listener then we have created another jtextfield through this we are getting the action listener object casted and then using getSource and getText methods we are getting text . rest of the components are being used to design the frame . below is shown visually:

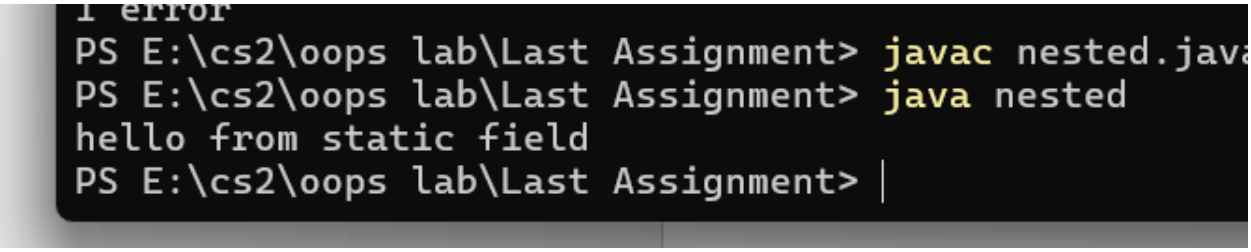



# Lecture 27

```
class OuterClass{
    private static String staticOuterField= "hello from static field";

    static class StaticInnerClass{
        public void displayMessage(){
            System.out.println(staticOuterField);
        }
    }
}
```

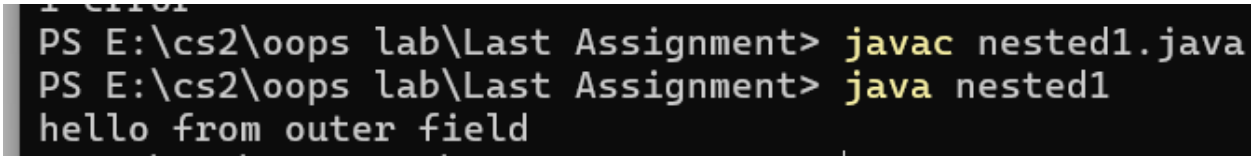
```
public class nested {
                public static void main (String [] args) throws Exception{
                        OuterClass.StaticInnerClass inner= new OuterClass.StaticInnerClass();
                        inner.displayMessage();
                        }
                }
```

Here we have nested inner class which is static so we are making the object of inner class without object of enclosing class.the following is the output of the code




```
class OuterClass{
private  String message= "hello from outer field";

 class Inner{
                public void displayMessage(){
                        System.out.println(message);
                        }
                }
}
public class nested1 {
                public static void main (String [] args) throws Exception{
                        Outer outer=new Outer();
                        Outer.Inner inner= outer.new Inner();
                        inner.display
                        }
                }
```

Here we have non static inner class and we can make object of this only by making the object of outer class. The output is show below:

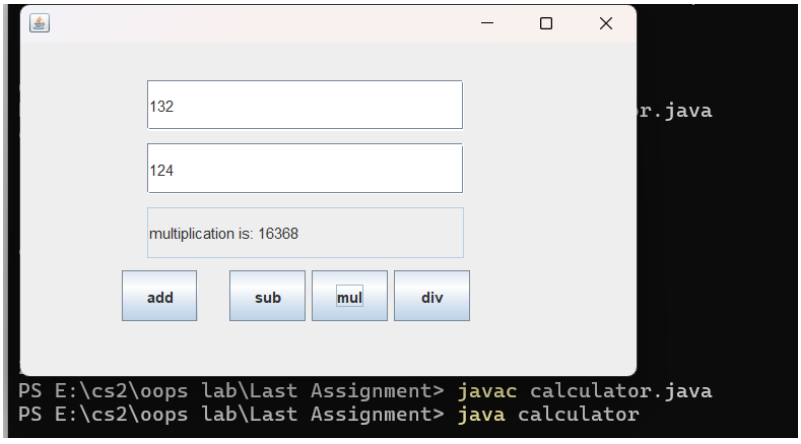



```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class calculatorWindow extends JFrame{
    JTextField tf1;
    JTextField tf2;
    JTextField tf3;
    calculatorWindow(){
        setLayout(null);
        setSize(500,300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        tf1=new JTextField();
        tf1.setBounds(100,30,250,40);
        tf2=new JTextField();
        tf2.setBounds(100,80,250,40);
        tf3=new JTextField();
        tf3.setBounds(100,130,250,40);
        tf3.setEditable(false);

        JButton jb1= new JButton("add");
        jb1.setBounds(80,180,60,40);

        JButton jb2= new JButton("sub");
        jb2.setBounds(165,180,60,40);

        JButton jb3= new JButton("mul");
        jb3.setBounds(230,180,60,40);
```

```
                JButton jb4= new JButton("div");
                jb4.setBounds(295,180,60,40);

                eventHandler evtH= new eventHandler();
                jb1.addActionListener(evtH);
                jb2.addActionListener(evtH);
                jb3.addActionListener(evtH);
                jb4.addActionListener(evtH);
                add(tf1);
                add(tf2);
                add(tf3);
                add(jb4);
                add(jb1);
                add(jb2);
                add(jb3);
                setVisible(true);
                        }
class eventHandler implements ActionListener{
                public void actionPerformed (ActionEvent ae){
                        int value1=Integer.parseInt(tf1.getText());
                        int value2=Integer.parseInt(tf2.getText());

        if(((JButton)ae.getSource()).getText().equalsIgnoreCase("add")){
                int value3=value1+value2;
                tf3.setText("Addition is: "+ value3);
                        }
        else if(((JButton)ae.getSource()).getText().equalsIgnoreCase("sub")){
                int value3=value1-value2;
                tf3.setText("subtraction is: "+ value3);
                                }
        else if(((JButton)ae.getSource()).getText().equalsIgnoreCase("mul")){
                int value3=value1*value2;

                tf3.setText("multiplication is: "+ value3);
```

```
                }
            else if(((JButton)ae.getSource()).getText().equalsIgnoreCase("div")){
                int value3=value1/value2;
                tf3.setText("division is: "+ value3);
                }
        }
    }
}

public class calculator{
    public static void main(String args[]){
        new calculatorWindow();
        }
    }
```

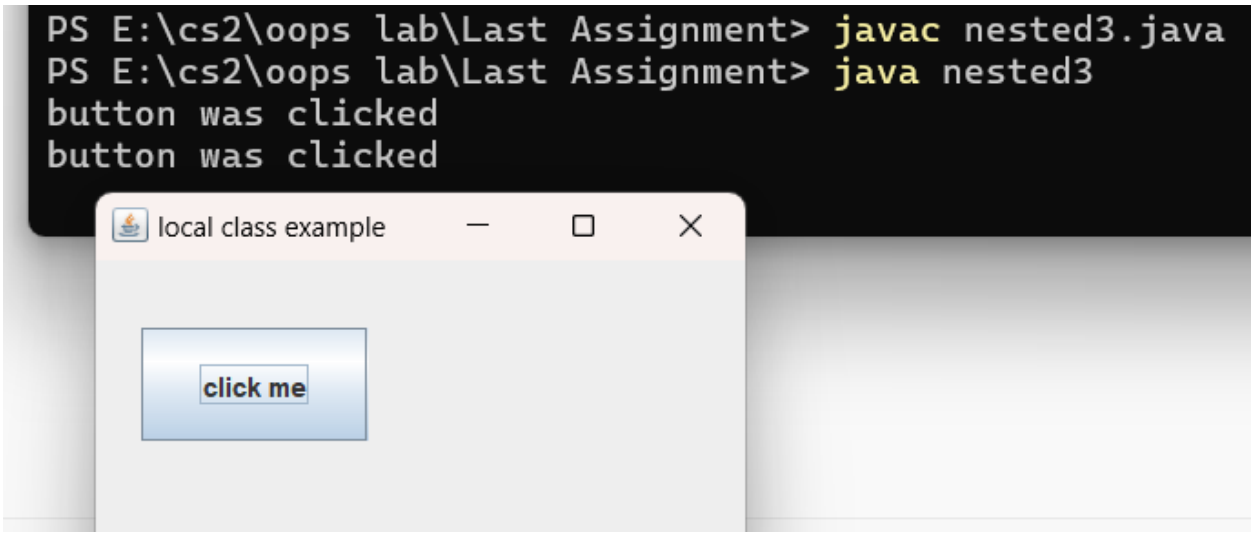
In this we have made simple calculator , we have extended the jframe and imported jbutton for GUI and added actionlistener to jbuttons and rest is simple logic. We have also used nested classes to get the calculator window code to the inner class where rest of the operation are performed .representation is given below




```
import javax.swing.*;
import java.awt.event.*;
public class nested3{
    public static void main(String args[]){
    JFrame frame= new JFrame("local class example");
        frame.setSize(300,200);
        frame.setLayout(null);
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

JButton button = new JButton("click me");

button.setBounds(20,30,100,50);

frame.add(button);


addClickListener(button);

frame.add(button);

frame.setVisible(true);

}

public static void addClickListener(JButton button){

class ButtonClickListener implements ActionListener{

public void actionPerformed(ActionEvent e){

System.out.println("button was clicked");

}

}

button.addActionListener(new ButtonClickListener());

}

}
```

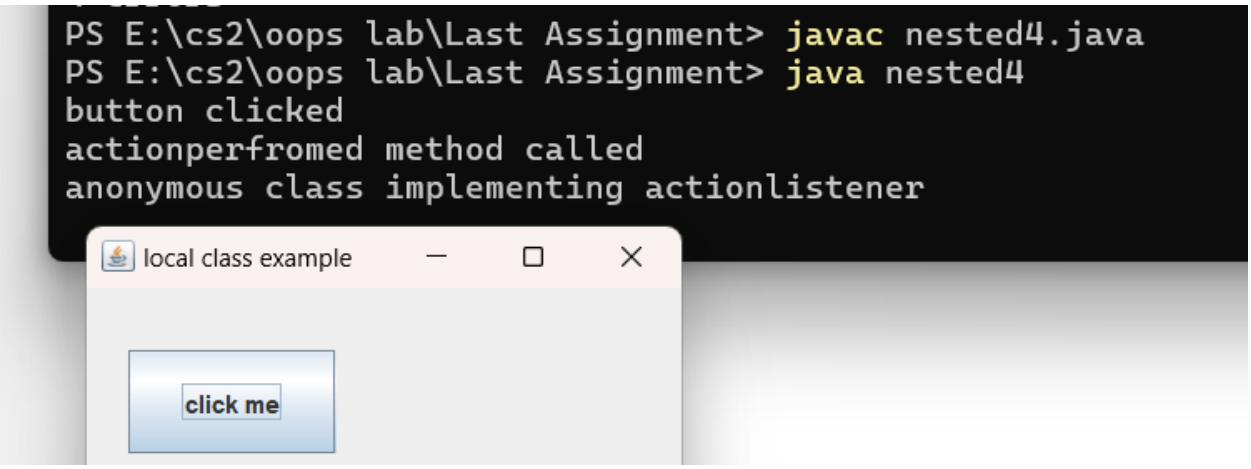
In this code we have made the method in which we pass the jbutton button then it goes to buttonclicklistener which is implementing the actionlistener which registers the click and then we have desired output which is attached below:




```
import javax.swing.*;

import java.awt.event.*;
```

```
public class nested4{
                                                    public static void main(String args[]){
                                                            JFrame frame= new JFrame("local class example");
                                                            frame.setSize(300,200);
                                                            frame.setLayout(null);
                                                            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                                                            JButton button = new JButton("click me");
                                                            button.setBounds(20,30,100,50);
                                                            frame.add(button);

                                                            button.addActionListener(new ActionListener(){
                                                                    public void actionPerformed(ActionEvent e){
                                                                            System.out.println("button clicked ");
                                                                            System.out.println("actionperfromed
method called");
                                                                            System.out.println("anonymous class
implementing actionlistener");

                                                                    }
                                                            });

                                                            frame.setVisible(true);

                                                     }
                                                    }
```

here we have created anonymous nested without a name and through we have sticked actionlistenrer. the output shown below:

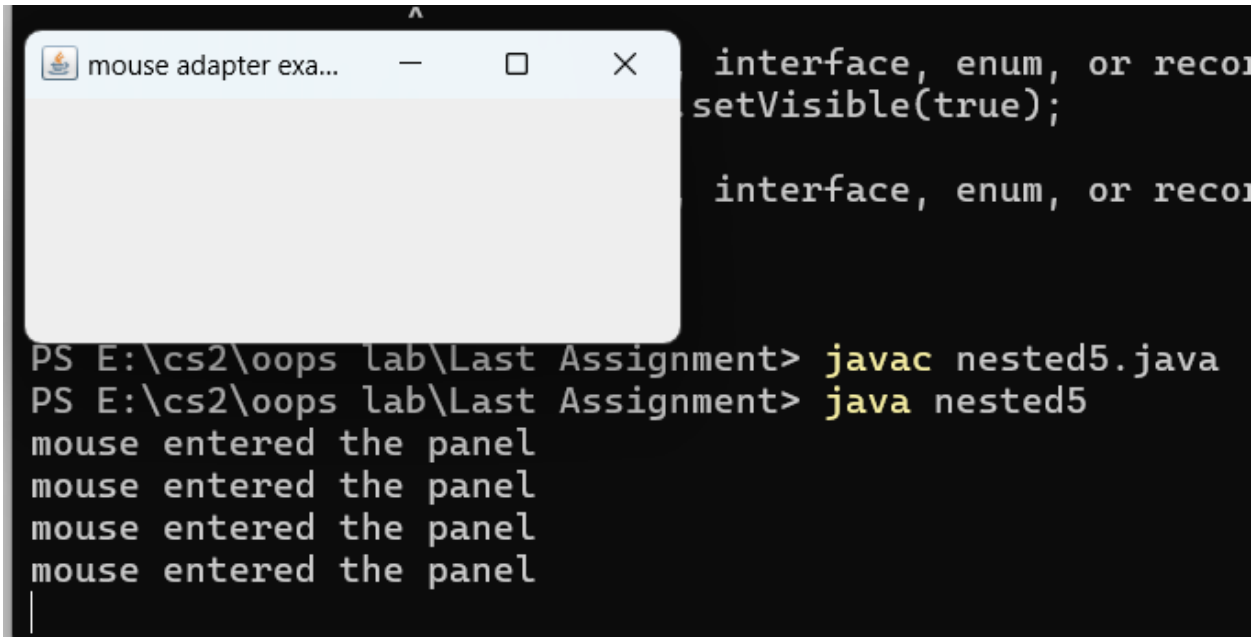



```
import javax.swing.*;
import java.awt.event.*;

public class nested5{
    public static void main (String [] args){
        JFrame frame= new JFrame("mouse adapter example");
        frame.setSize(300,400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel= new JPanel();
        frame.add(panel);

        panel.addMouseListener(new MouseAdapter(){
            public void mouseclicked(MouseEvent e){
            System.out.println("mouse clicked at:"+e.getPoint());
            }
        public void mouseEntered(MouseEvent e){
            System.out.println("mouse entered the panel");
            }

        });
        frame.setVisible(true);
```

}

}

here we have added mouselistener which tracks the movement of mouse here we have one function of mouse which is entering the Jpanel and we have also used anonymous nested class



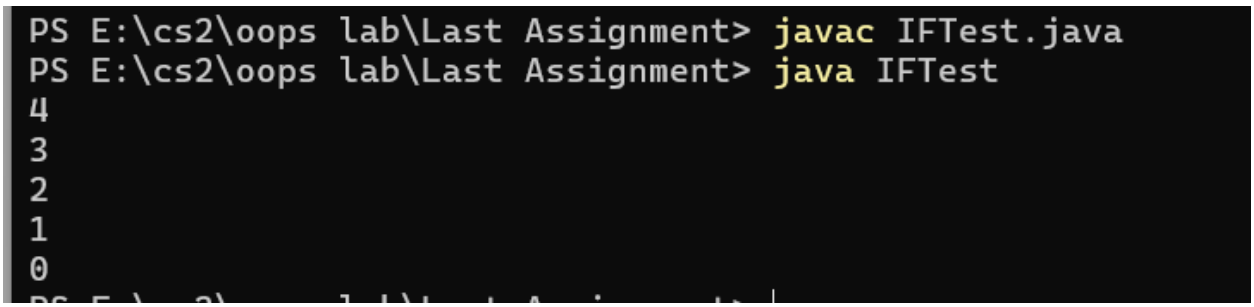

# Lecture 28

```
interface IntStack{
                                        void push(int item );
                                        int pop();
                                        }
class IFTest{
                                        public static void main(String args[]){
                                                FixedStack mystack1= new FixedStack(5);
                                                for(int i=0;i<5;i++)
                                                        mystack1.push(i);
                                                for(int i=0;i<5;i++)
                                                        System.out.println(mystack1.pop());
                                                        }
                                        }
class FixedStack implements IntStack{
                                        private int stck[];
```

```
private int tos;

FixedStack(int size){
        stck= new int[size];
        tos=-1;
    }
public void push(int item){
        if(tos==stck.length-1)
            System.out.println("stack is full");
        else
            stck[++tos]=item;
        }
public int pop(){
    if(tos<0){
        System.out.println("stack underflow.");
        return 0;
                }
    else
        return stck[tos--];
    }
}
```

here we have created an interface where we have created two method

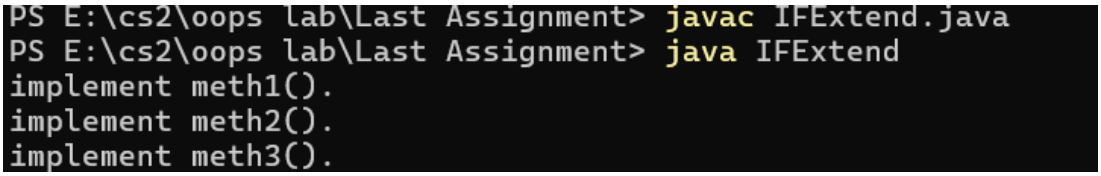
and we are implementing this interface in fixedstack .here we have fixed the size of the stack. the output is shown below:




```
interface A{
void meth1();
void meth2();
```

```
}
interface B extends A {
        void meth3();
        }

class MyClass implements B{
        public void meth1(){
                System.out.println("implement meth1().");
                }
        public void meth2(){
                System.out.println("implement meth2().");
                }
        public void meth3(){
                System.out.println("implement meth3().");
                }
        }
class IFExtend{
    public static void main(String args[]){
                MyClass ob= new MyClass ();

                ob.meth1();
                ob.meth2();
                ob.meth3();
                }
        }
```

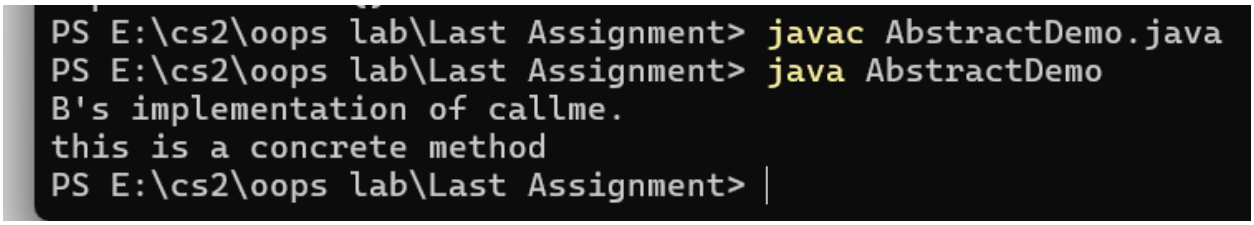
Here we are making one interface with two methods and extending and adding third method. thorugh this we are getting 3 methods in the object of the child of the methods.following is the output shared:




```
    abstract class A{
        abstract void callme();
```

```
    void callmetoo(){
        System.out.println("this is a concrete method");
        }

}
class B extends A{
    void callme(){
        System.out.println("B's implementation of callme.");
        }
}
class AbstractDemo{
    public static void main(String args[]){
        B b= new B();
        b.callme();
        b.callmetoo();
        }
}
```

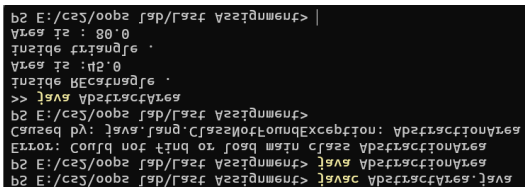
we are accessing an abstract class through the object of it's child class and the abstract method is also accessed thorugh overriding .




```
abstract class Figure {
        double dim1;
        double dim2;
Figure (double a, double b){
            dim1=a;
            dim2=b;
        }
abstract double area();
```

```
}
class Rectangle extends Figure {
        Rectangle(double a, double b){
            super (a,b);
            }
    double area(){
        System.out.println("inside REcatnagle .");
        return dim1*dim2;
        }
    }
class Triangle extends Figure{
    Triangle (double a, double b){
        super(a,b);
        }
    double area(){
        System.out.println("inside triangle .");
        return dim1*dim2;
        }
    }
class AbstractArea{
    public static void main(String args[]){
        Rectangle r= new Rectangle(9,5);
        Triangle t = new Triangle(10,8);
        Figure figref;
        figref=r;
        System.out.println("Area is :"+figref.area());
        figref=t;
        System.out.println("Area is : "+figref.area());
        }
    }
```

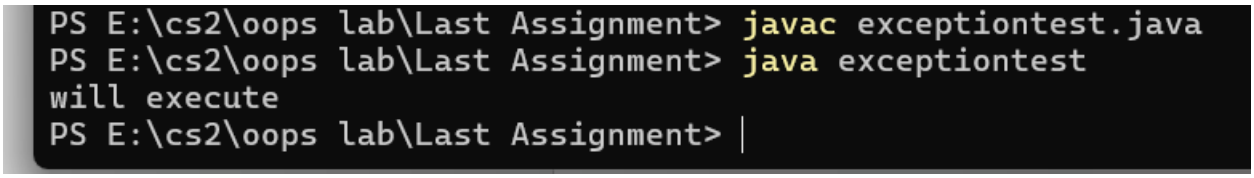
here we can not make the object of the abstract class but we can acces that through the objects of the child class . the output of the program is shown below:



# LECTURE 29-I

```
public class exceptiontest{
                                        public static void main(String args[]){
                                                int a=0;
                                                int b=10;
                                        try{
                                                int c=b/a;
                                                }
                                        catch(ArithmeticException ex){
                                                }
                                        System.out.println("will execute ");
                                        }
                                }
```

here we handling the arithmetic exception throw try and catch method




```
                                                                class Exc1{
                                                                static void subroutine(){
                                                                        int d=0;
                                                                        int a=10/d;
                                                                                }
                                                                public static void main(String args[]){
                                                                                Exc1.subroutine();
                                                                        }
}
```

here we have encountered the error which we handled in the previous code

the error is shown below:

```
PS E:\cs2\oops lab\Last Assignment> javac Exc1.java
PS E:\cs2\oops lab\Last Assignment> java Exc1
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Exc1.subroutine(Exc1.java:4)
        at Exc1.main(Exc1.java:7)
PS E:\cs2\oops lab\Last Assignment>
```

```
public class unchecked{
    public static void main(String args[]){
        int[] ab=new int[2];
        try{
            System.out.println(ab[3]);
            }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
```

we have encountered array out of bound exception.

```
PS E:\cs2\oops lab\Last Assignment> javac unchecked.java
PS E:\cs2\oops lab\Last Assignment> java unchecked
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2
        at unchecked.main(unchecked.java:5)
PS E:\cs2\oops lab\Last Assignment>
```

```
public class unchecked1{
    public static void main(String args[]){
        try{
            int a=Integer.parseInt("abc");
            }
        catch(Exception ex){
            ex.printStackTrace();
            }
    }
```

```
}
```

this exeption occurs when we are extractin the integer values from String which has

alphabets

```
java.lang.NumberFormatException: For input string: "abc"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:588)
        at java.base/java.lang.Integer.parseInt(Integer.java:685)
        at unchecked1.main(unchecked1.java:4)
PS E:\cs2\oops lab\Last Assignment>
```

```
class MultiCatch{
        public static void main(String args[]){
                try{
                        int a=args.length;
                        System.out.println("a= "+a);
                        int b=42/a;
                        int c[]={1};
                        c [42]=99;
                        }
                catch(ArithmeticException e){
                        System.out.println("arithmetic exception");
                        }
                catch(ArrayIndexOutOfBoundsException e){
                        System.out.println("array out of bound exception:");
                                }
                System.out.println("after try and catch block");
        }
}
```

we can mutliple exceptions like here we have got both array and arithmetic exceptions

```
PS E:\cs2\oops lab\Last Assignment> javac MultiCatch.java
PS E:\cs2\oops lab\Last Assignment> java MultiCatch
a=  0
arithmetic exception
after try and catch block
```

```
class SuperSubCatch{
        public static void main(String args[]){
```

```
        try{
                int a=0;
                int b=42/a;
        }
catch(Exception e){
        System.out.println("generic exception catch");
                        }
catch(ArithmeticException e){
        System.out.println("this catch is not reached");
                }
        }
}
```

this denotes that the differt between genric and specific catch is by order which comes first. following note is from compiler

```
PS E:\cs2\oops lab\Last Assignment> javac SuperSubCatch.java
SuperSubCatch.java:10: error: exception ArithmeticException has already been caught
                catch(ArithmeticException e){
                ^
1 error
PS E:\cs2\oops lab\Last Assignment>
```

# LECTURE-29

```
class NestTry {
  public static void main(String args[]) {
    try {
      int a = args.length;
      int b = 42 / a;
      System.out.println("a = " + a);

      try {
        if (a == 1) a = a / (a - a);

        if (a == 2) {
          int c[] = { 1 };
```

```
            c[42] = 99;
          }
        } catch (ArrayIndexOutOfBoundsException e) {
          System.out.println("Array index out-of-bounds: " + e);
        }
      } catch (ArithmeticException e) {
        System.out.println("Divide by 0: " + e);
      }
    }
}
```

the inner try can catch the outer catch statement like it happens here

```
PS E:\cs2\oops lab\Last Assignment> javac NestTry.java
PS E:\cs2\oops lab\Last Assignment> java NestTry
Divide by 0: java.lang.ArithmeticException: / by zero
```

```
public class exceptiontest1 {
  public static void main(String args[]) {
    try {
      int[] numbers = {1, 2, 3};
      System.out.println(numbers[5]);
      String text = null;
      System.out.println(text.length());
      int result = 10 / 0;
    } catch (ArrayIndexOutOfBoundsException | NullPointerException | ArithmeticException ex) {
    }
  }
}
//there is one try and multiple classes of catch.
```

```
PS E:\cs2\oops lab\Last Assignment> javac exceptiontest1.java
PS E:\cs2\oops lab\Last Assignment> java exceptiontest1.java
PS E:\cs2\oops lab\Last Assignment>
```

```
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;

import java.io.IOException;


public class FileDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;

      try {

         fis = new FileInputStream("sample.txt");

         int k;

         while ((k = fis.read()) != -1) {

            System.out.print((char) k);

         }

      } catch (FileNotFoundException e) {

         System.out.println("File not found");

      } catch (IOException e) {

         System.out.println("IOException occurred");

      } finally {

         try {

            if (fis != null) {

               fis.close();

            }

         } catch (IOException e) {

            System.out.println("Error while closing the stream: " + e.getMessage());

         }

      }

   }

}
```

file not found exception , i can't understand the code too

```
PS E:\cs2\oops lab\Last Assignment> javac FileDemo.java
PS E:\cs2\oops lab\Last Assignment> java fileDemo
Error: Could not find or load main class fileDemo
Caused by: java.lang.NoClassDefFoundError: fileDemo (wrong name: FileDemo)
PS E:\cs2\oops lab\Last Assignment> java FileDemo
File not found
PS E:\cs2\oops lab\Last Assignment>
```

```
import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;




public class AutoCloseDemo1 {

  public static void main(String[] args) {

    String filepath = "sample.txt";


    try (FileInputStream fis = new FileInputStream(filepath)) {

      // code to read file

    } catch (IOException e) {

      System.out.println("An error occurred: " + e.getMessage());

    }

  }

}
```

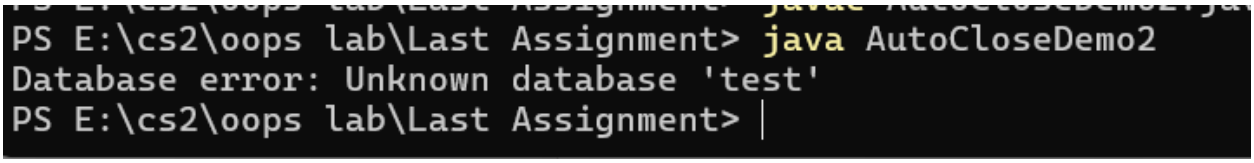
i cant understand what is happening here

```
PS E:\cs2\oops lab\Last Assignment> javac AutoCloseDemo1.java
PS E:\cs2\oops lab\Last Assignment> java AutoCloseDemo1
An error occurred: sample.txt (The system cannot find the file specified)
PS E:\cs2\oops lab\Last Assignment>
```

```
                                    import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;


public class AutoCloseDemo2 {

  public static void main(String[] args) {

    String url = "jdbc:mysql://localhost:3306/test";

    String user = "root";

    String password = "2006";


    try (Connection con = DriverManager.getConnection(url, user, password);
```

```
         Statement stmt = con.createStatement()) {
        String sql = "insert into users(id, name) values(1, 'John')";
        int rows = stmt.executeUpdate(sql);
        System.out.println("Rows inserted: " + rows);
      } catch (SQLException e) {
        System.out.println("Database error: " + e.getMessage());
      }
    }
}
```

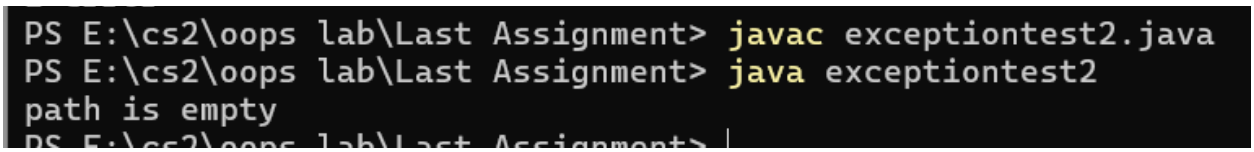
here the database is not availble




```
class InvalidPath extends Exception {
   public InvalidPath(String message) {
      super(message);
   }
}


class MyResource {
   public MyResource(String path) throws InvalidPath {
      if (path == null || path.isEmpty()) {
         throw new InvalidPath("path is empty");
      }
   }
}


public class exceptiontest2 {
   public static void main(String args[]) {
      try {
         MyResource ob = new MyResource(null);
         System.out.println(ob);
      } catch(InvalidPath ex) {
         System.out.println(ex.getMessage());
```

```
    }
  }
}
```

I am not understanding just understanding the exception is nchecked




```
class InvalidPathUnchecked extends RuntimeException {
  public InvalidPathUnchecked(String message) {
    super(message);
  }
}

class MyResource {
  public MyResource(String path) {
    if (path == null || path.isEmpty()) {
      throw new InvalidPathUnchecked("path is empty");
    }
  }
}

public class exceptiontest3 {
  public static void main(String args[]) {
    MyResource ob = new MyResource(null);
    System.out.println(ob);
  }
}
```

Unchecked exception I am unable to understand certain keywords