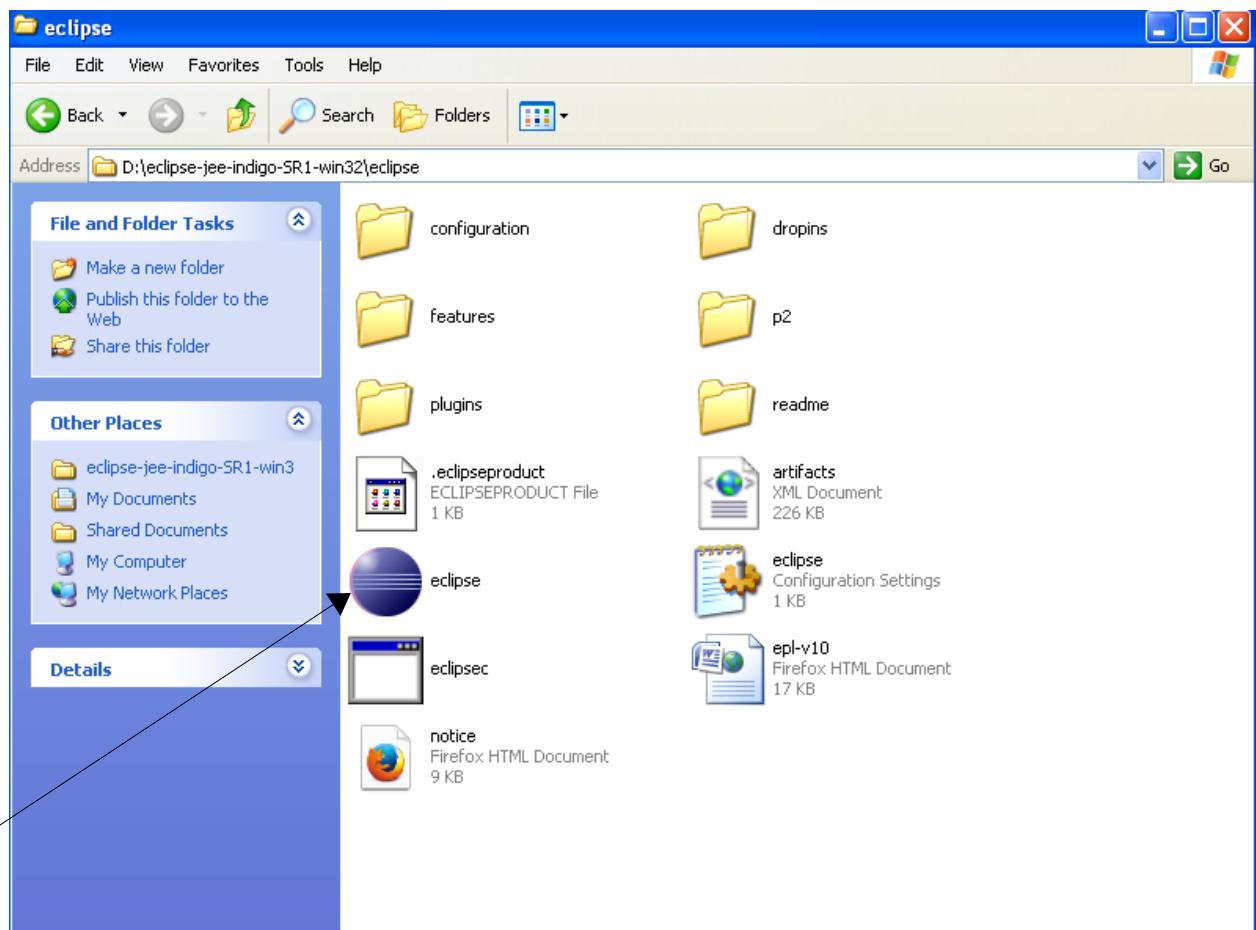
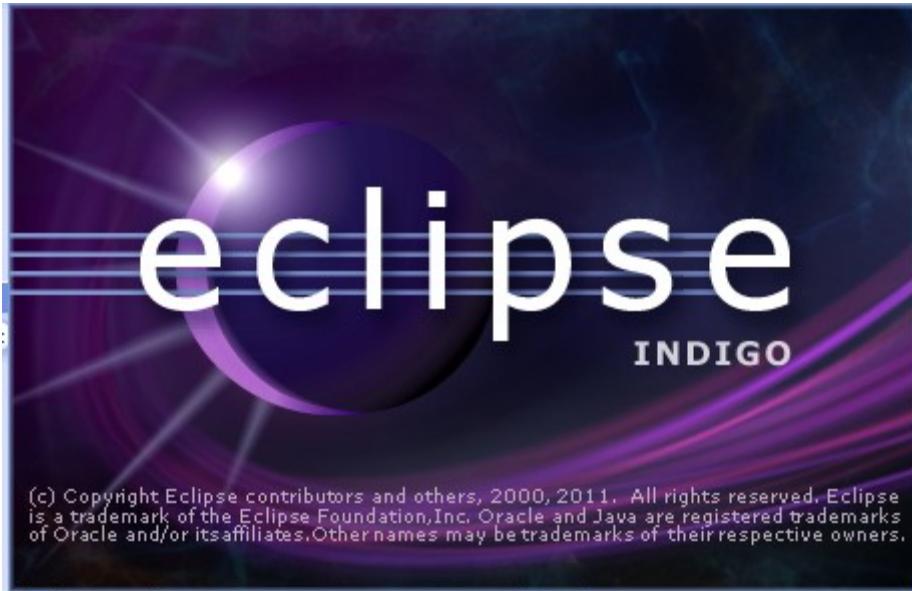


### Introducing Eclipse ID

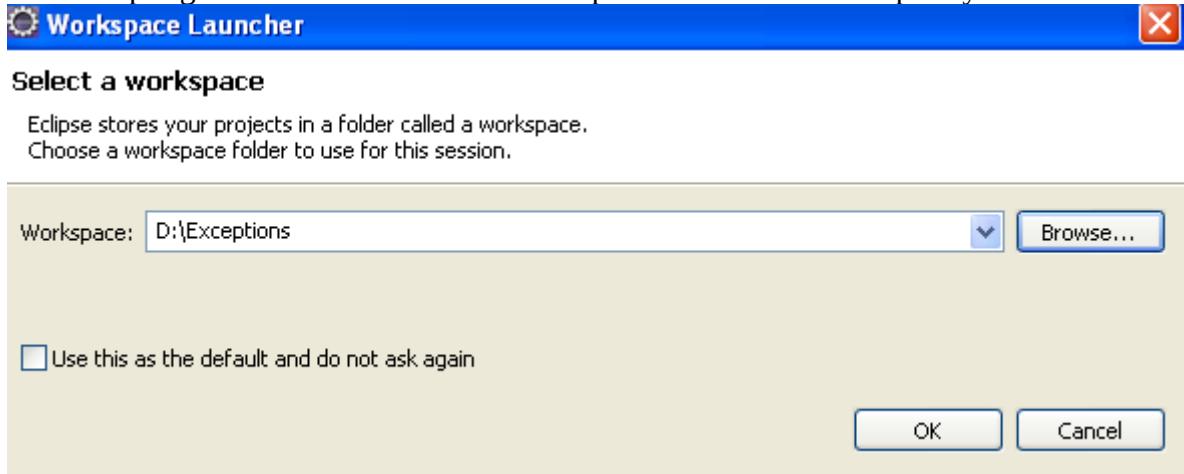
#### Steps:

1. Create a workspace (folder)(for eg in : d drive)
2. Open Eclipse  
Go to eclipse folder and double click on eclipse icon



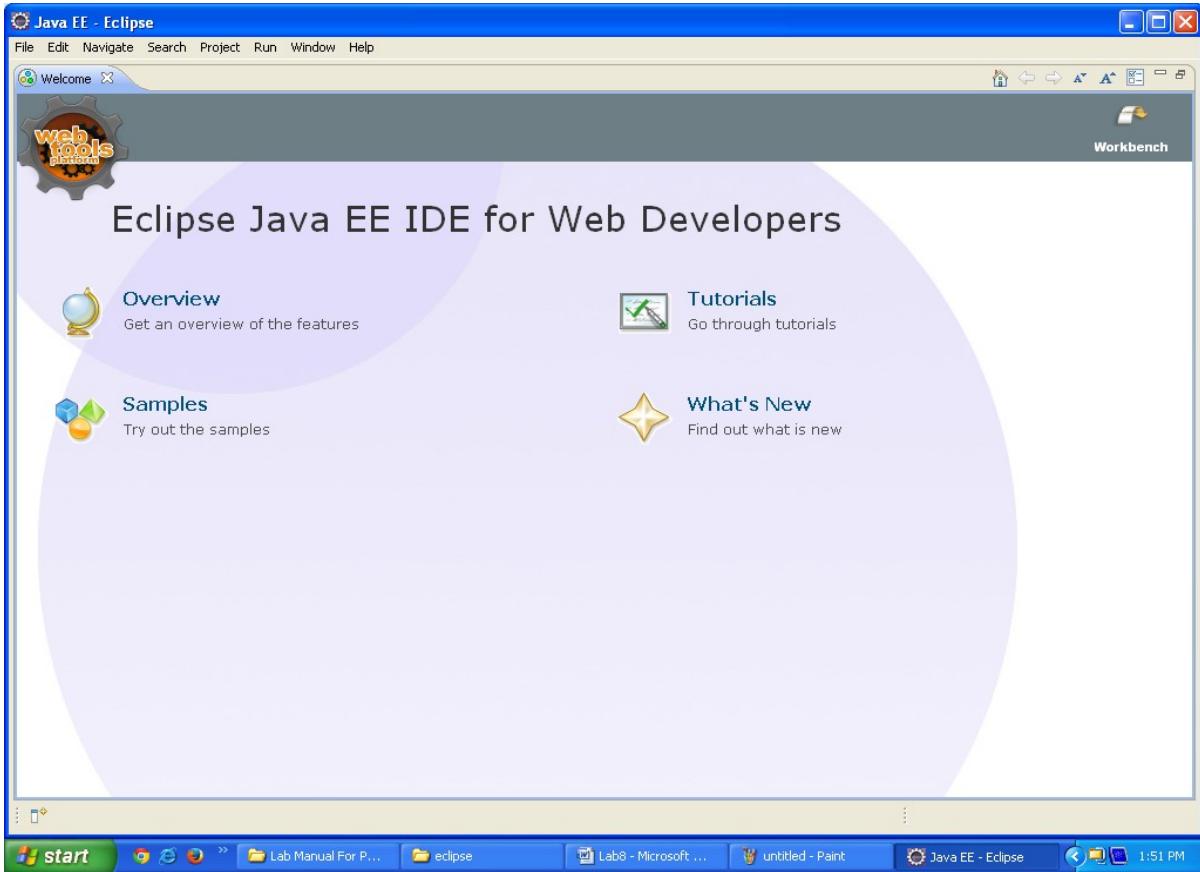


Once eclipse gets started it asks for the workspace. Choose the workspace you have created

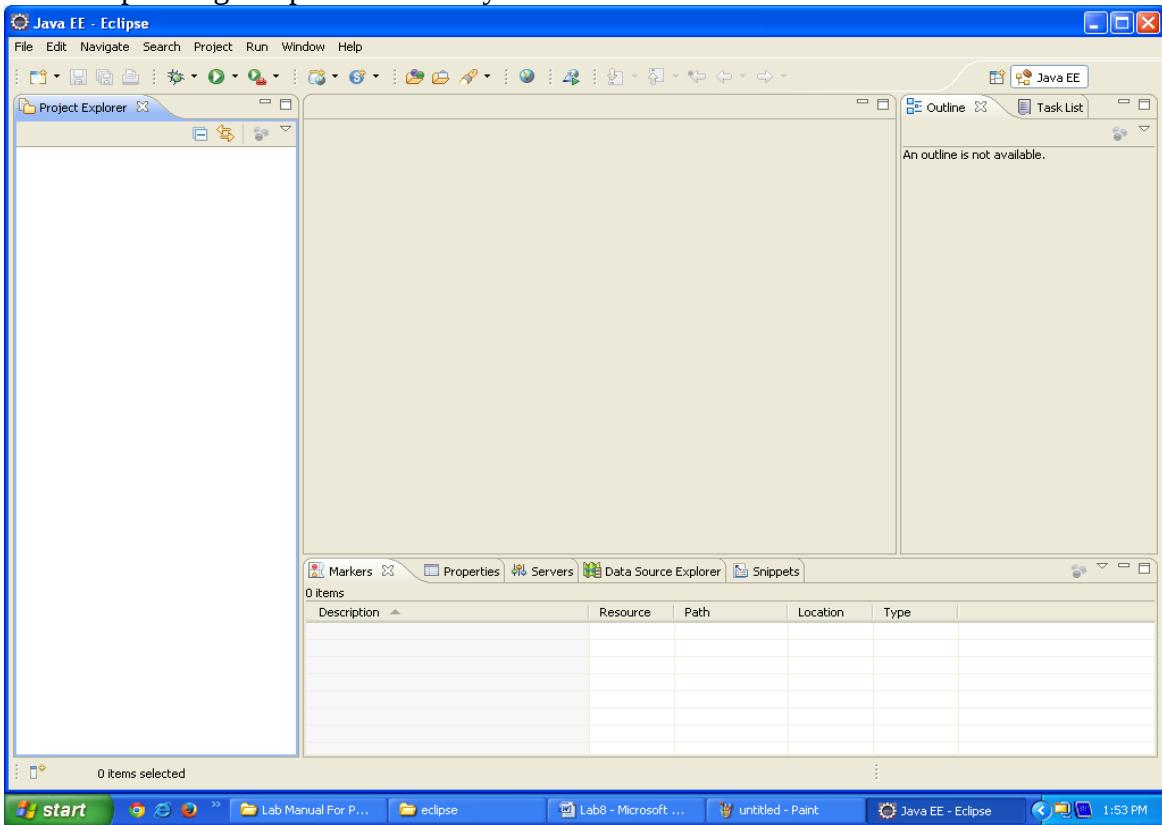


Click ok ...

## Lab Manual



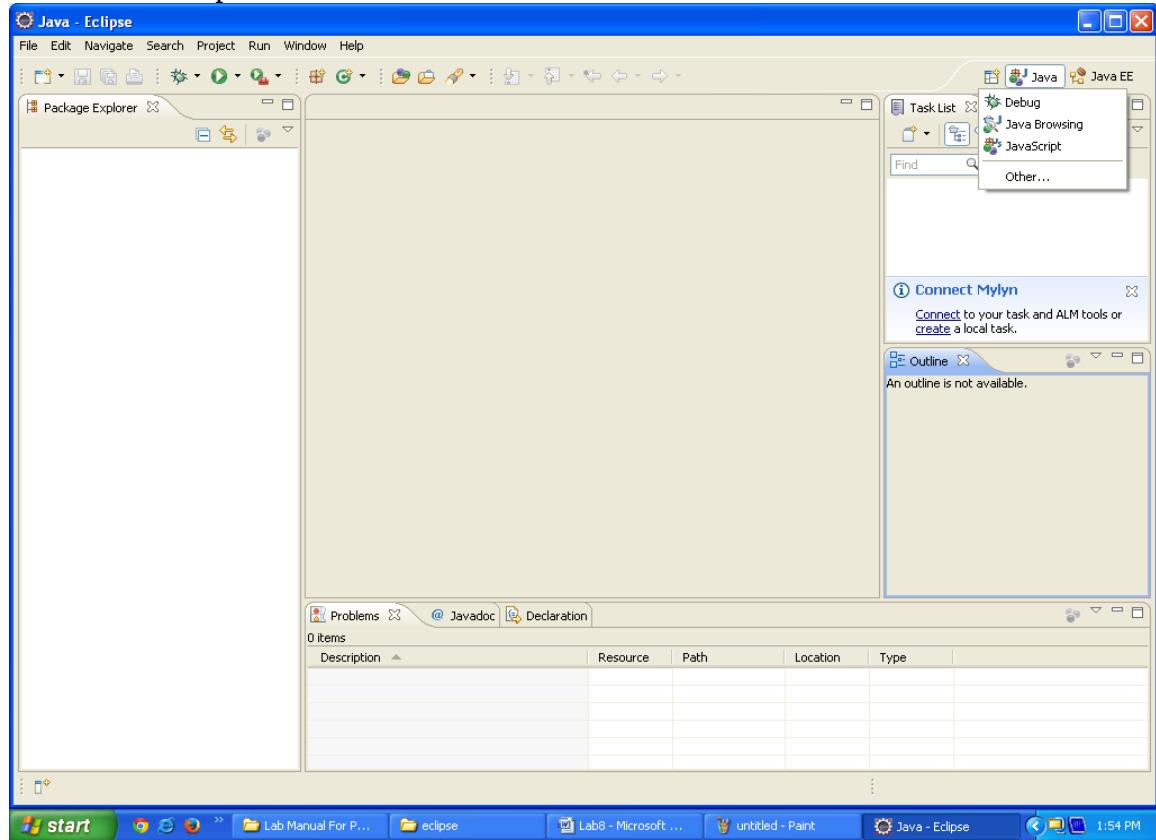
Click on cross button  
Your eclipse ID gets opened and ready



## Lab Manual

---

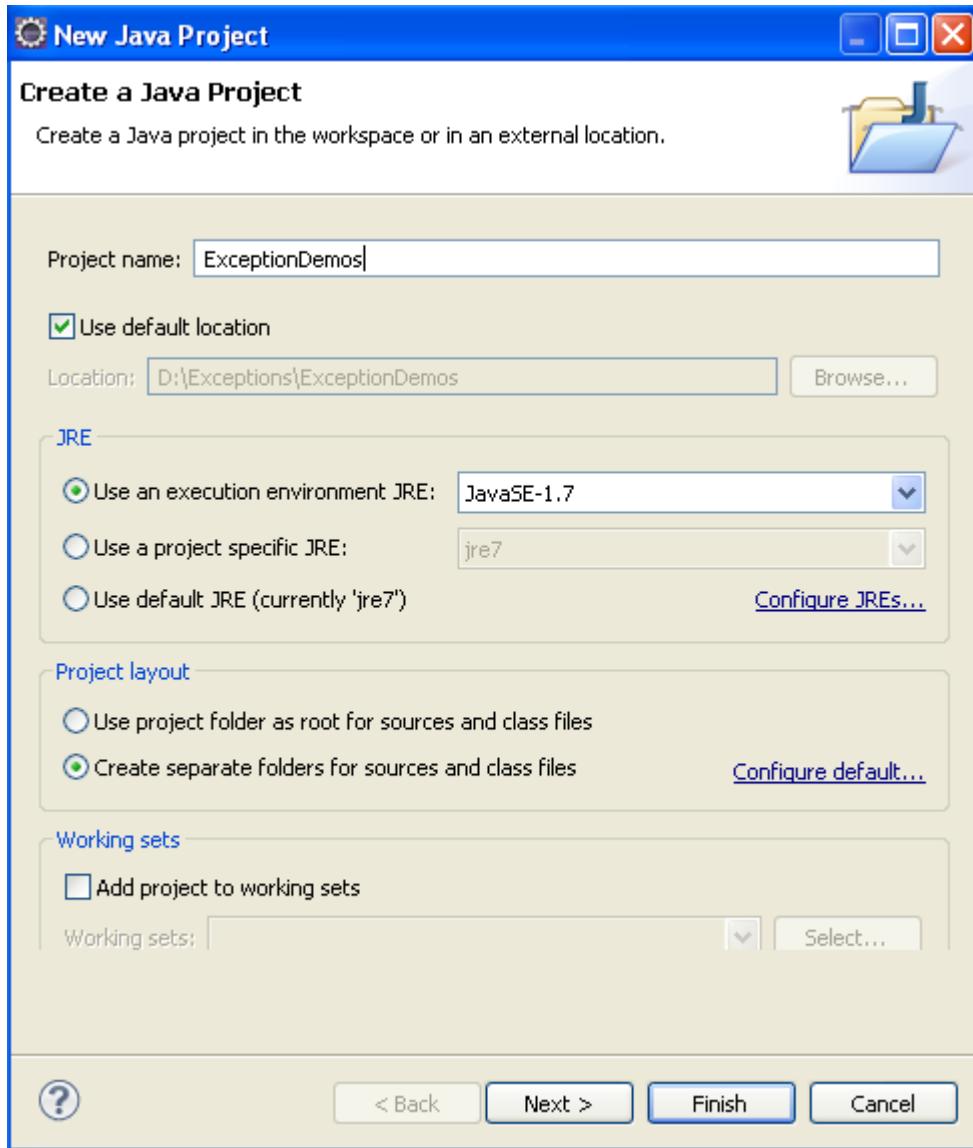
### Select Java Perspective



Create a new Project

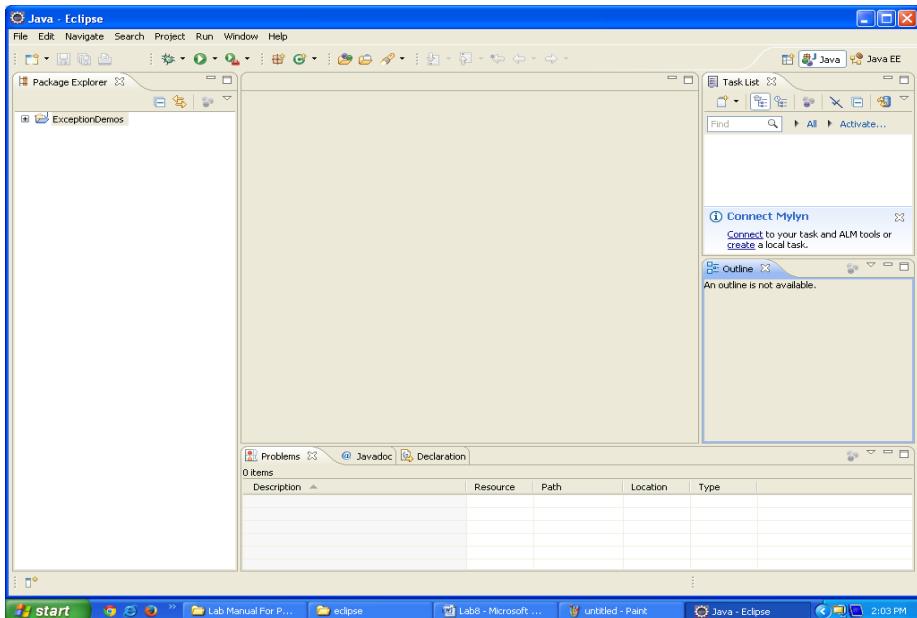
File -> New ->Java project

## Lab Manual

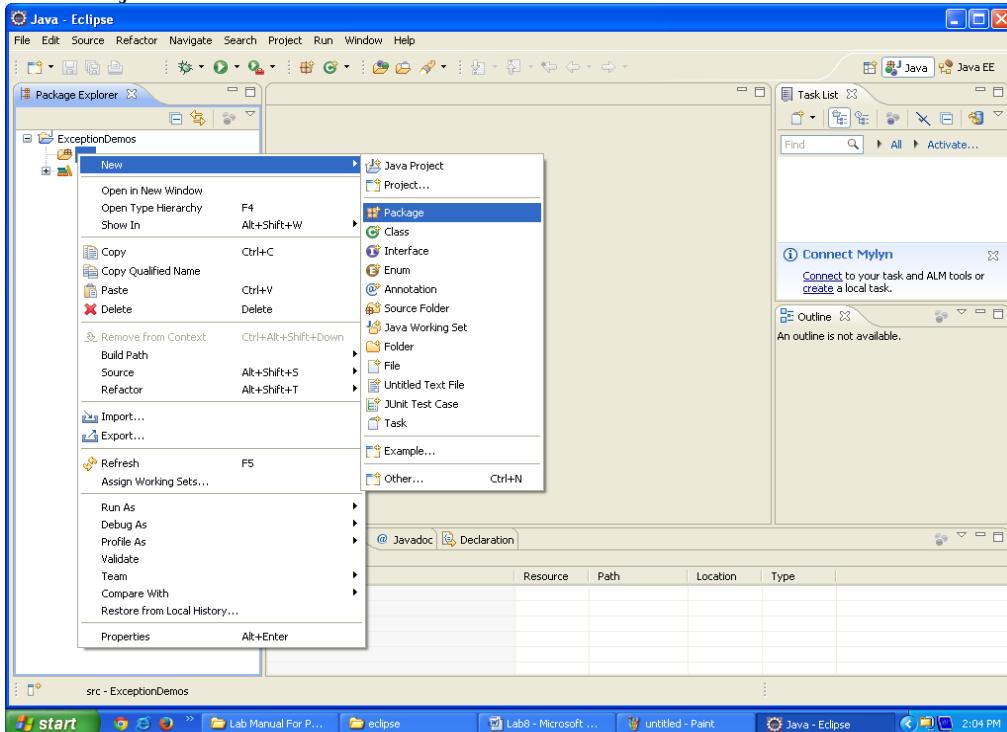


Click on Next- > Finish

## Lab Manual



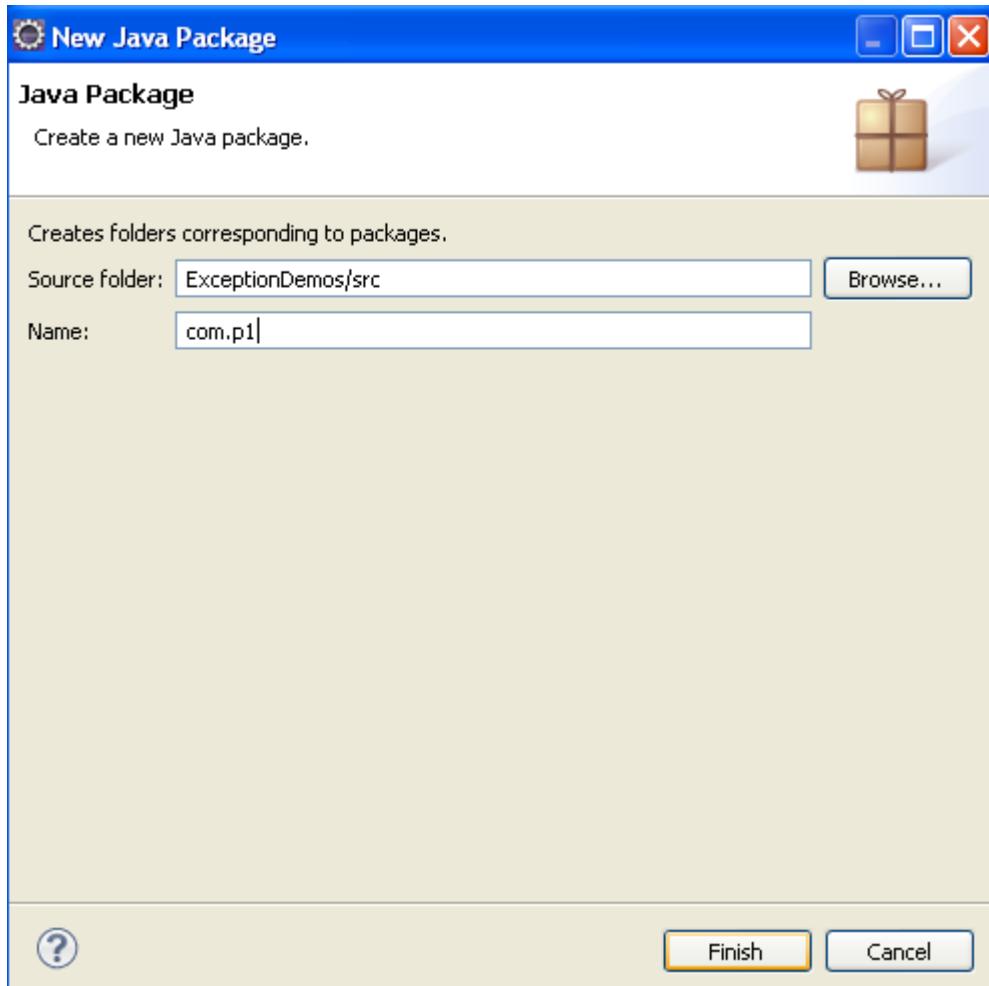
Your Project Gets Created.



Create a package by clicking on  
src -> New ->Package

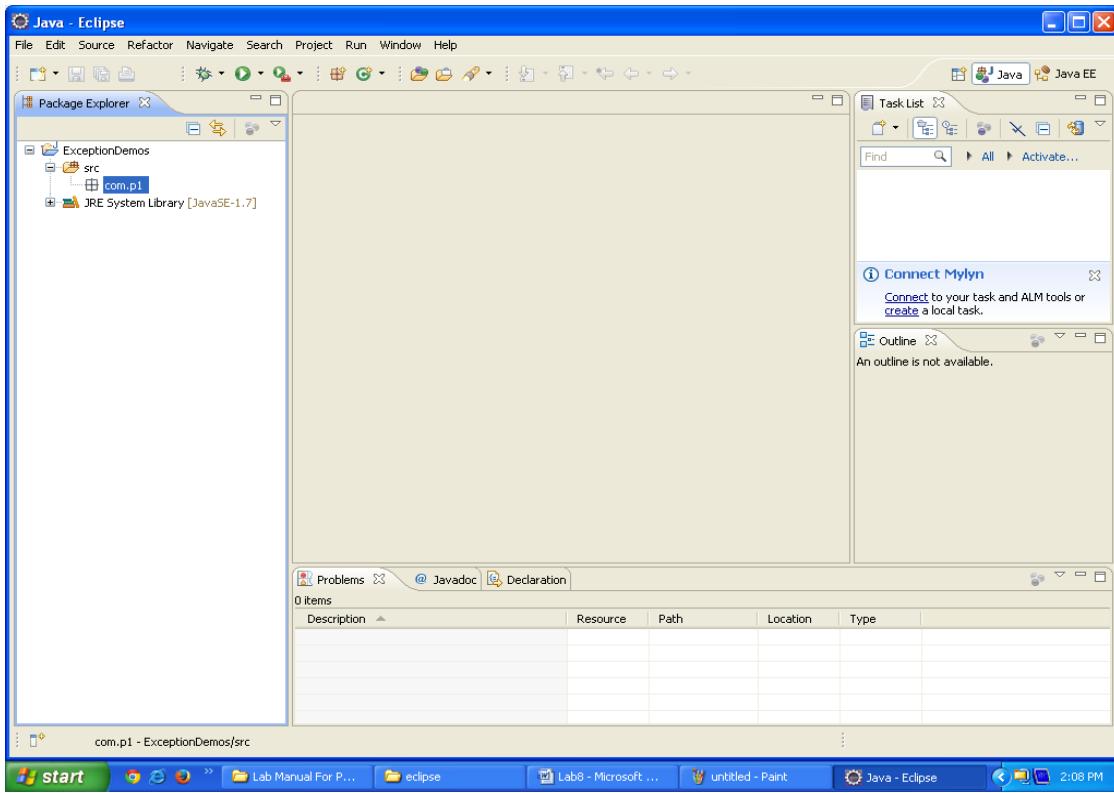
## Lab Manual

---



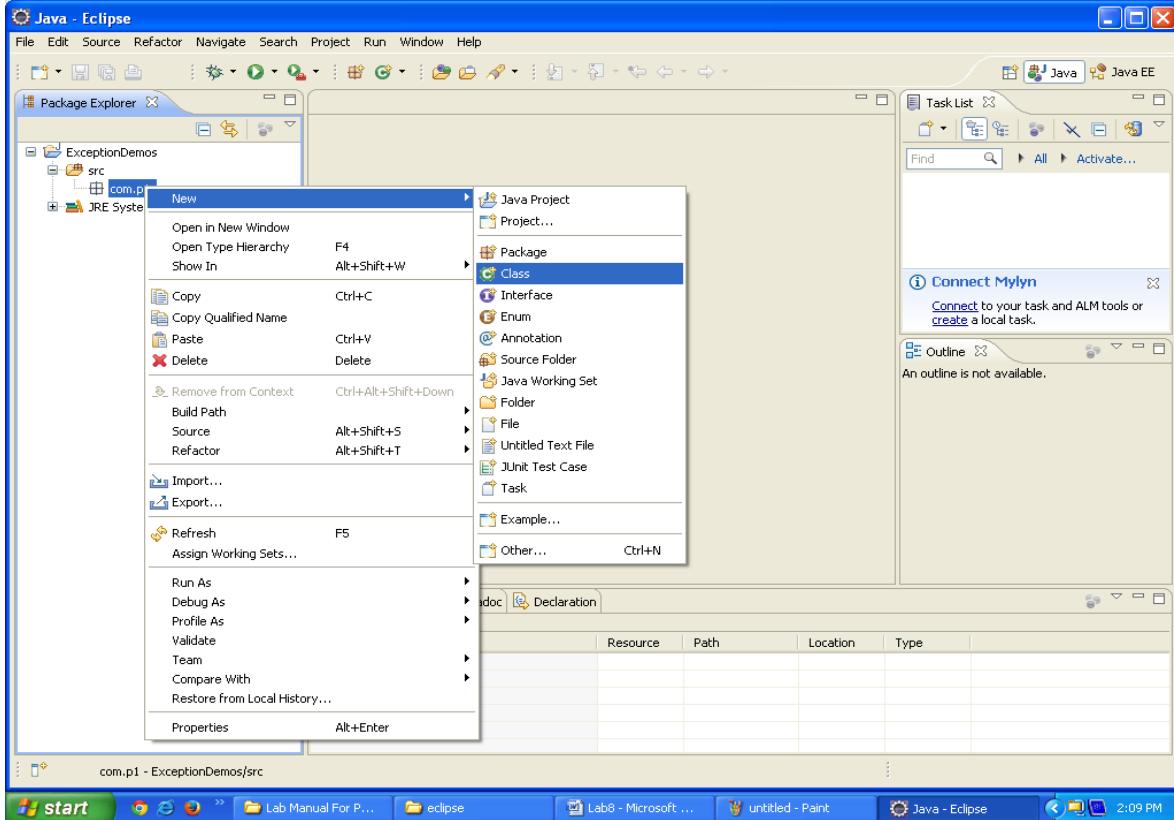
Give name like com.p1 as package name and click on Finish.

## Lab Manual

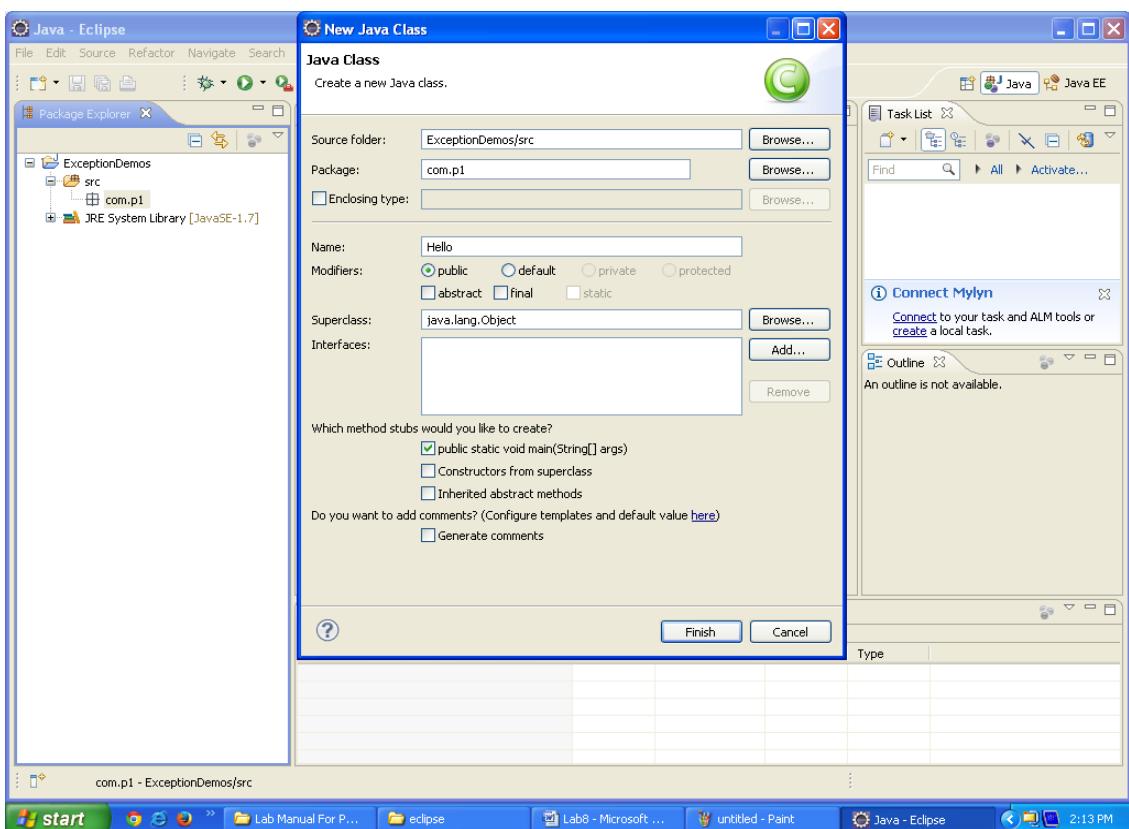
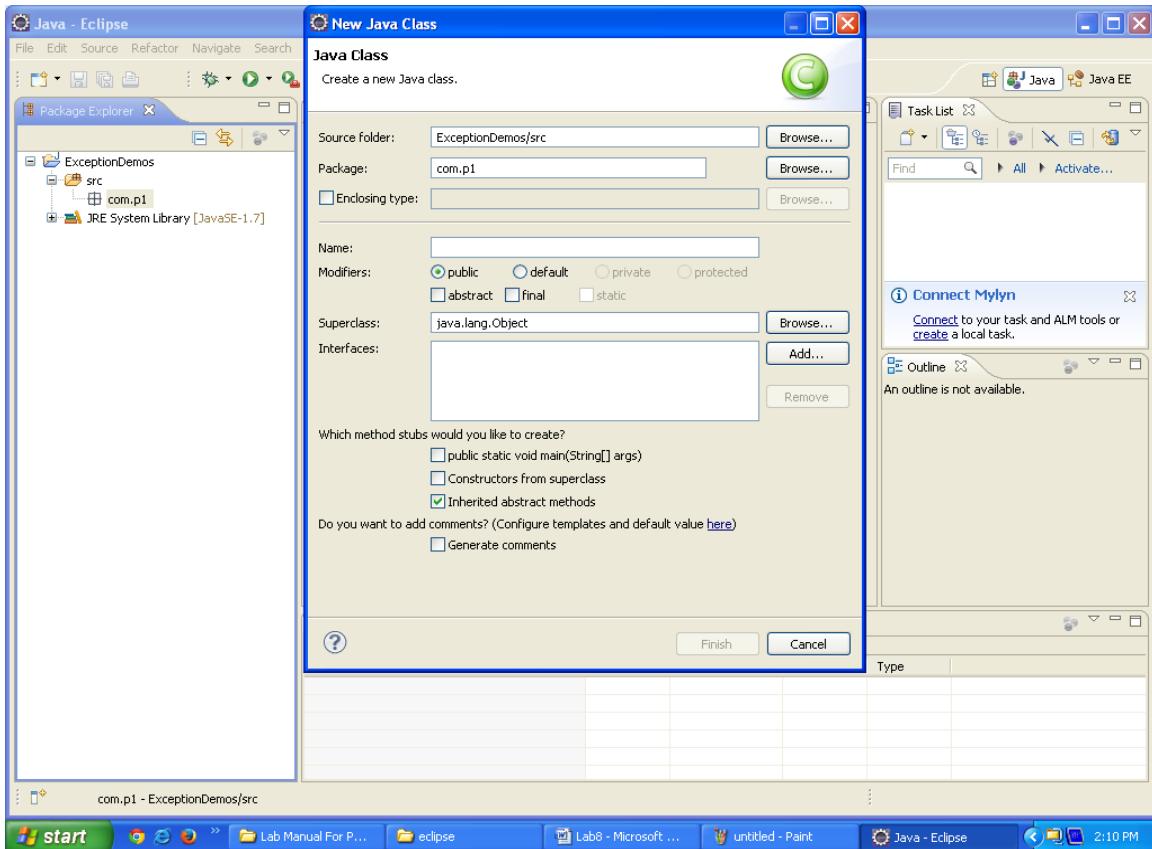


Package got created

Right click on package and create a java class

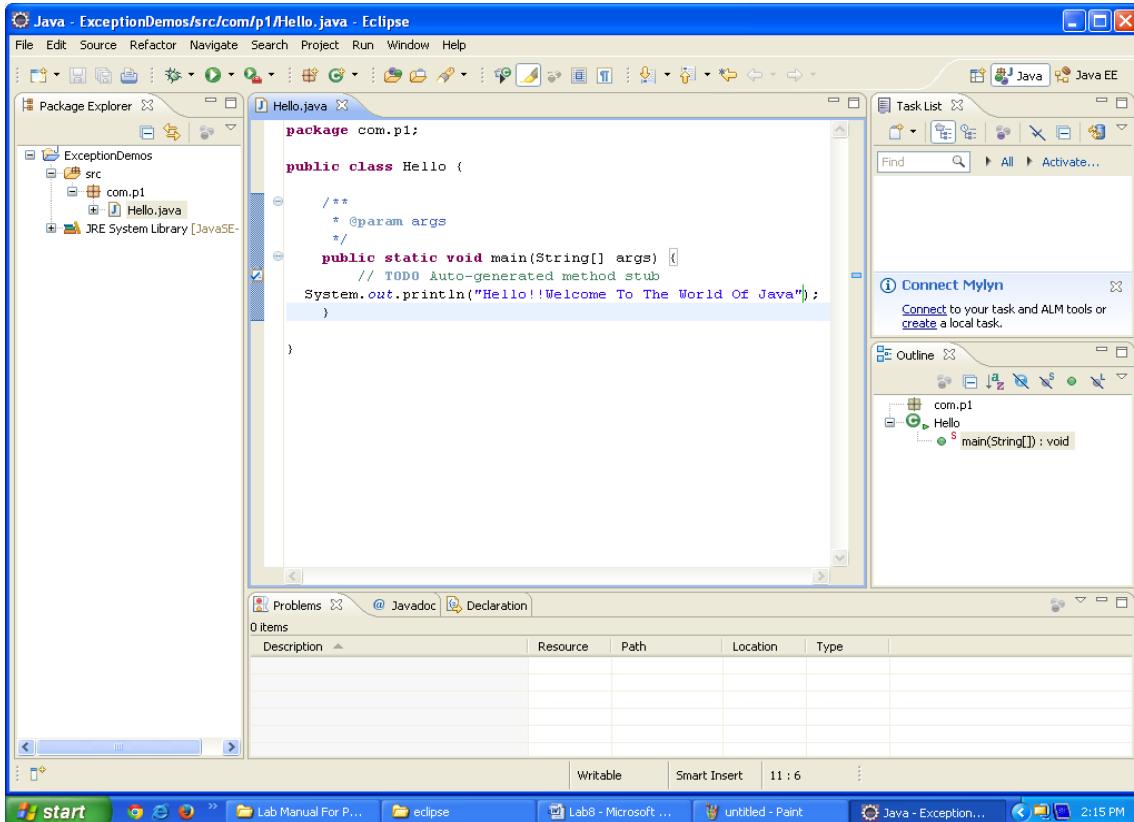


## Lab Manual



Enter the name of the class as Hello and click on Finish

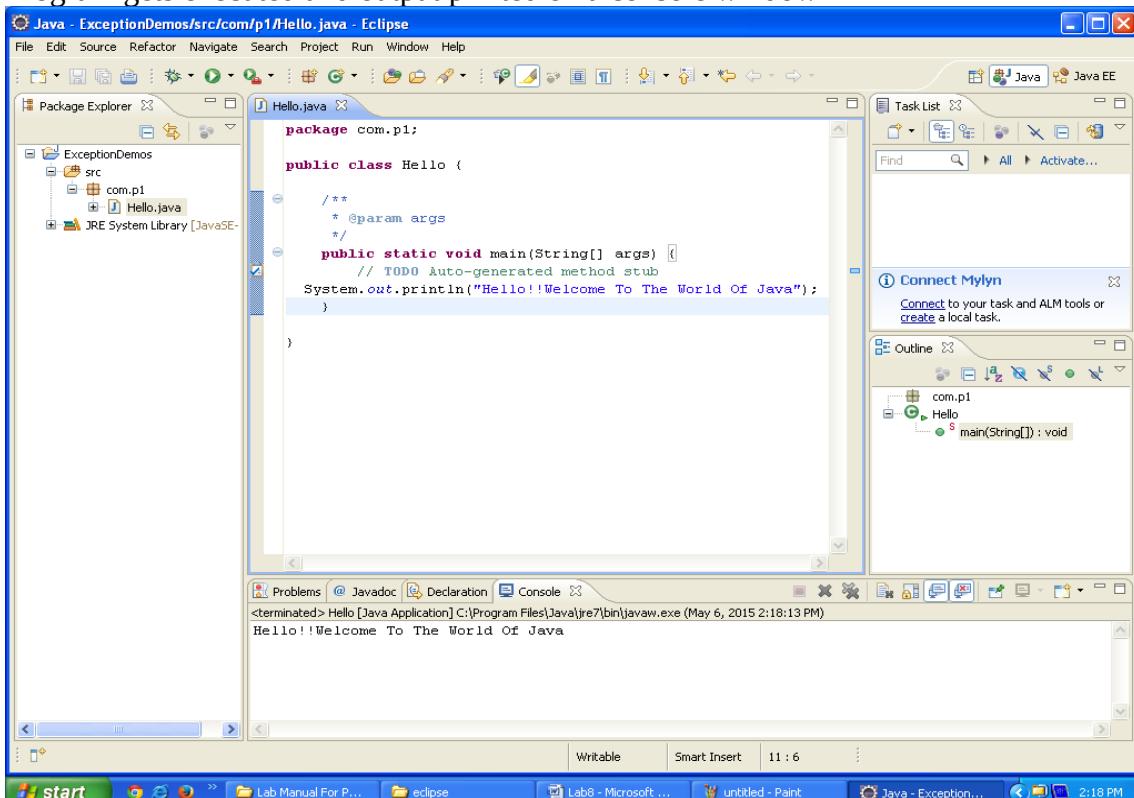
## Lab Manual



Execute and run the program

Run -> Run As -> Java Application

Program gets executed and output printed on a console window

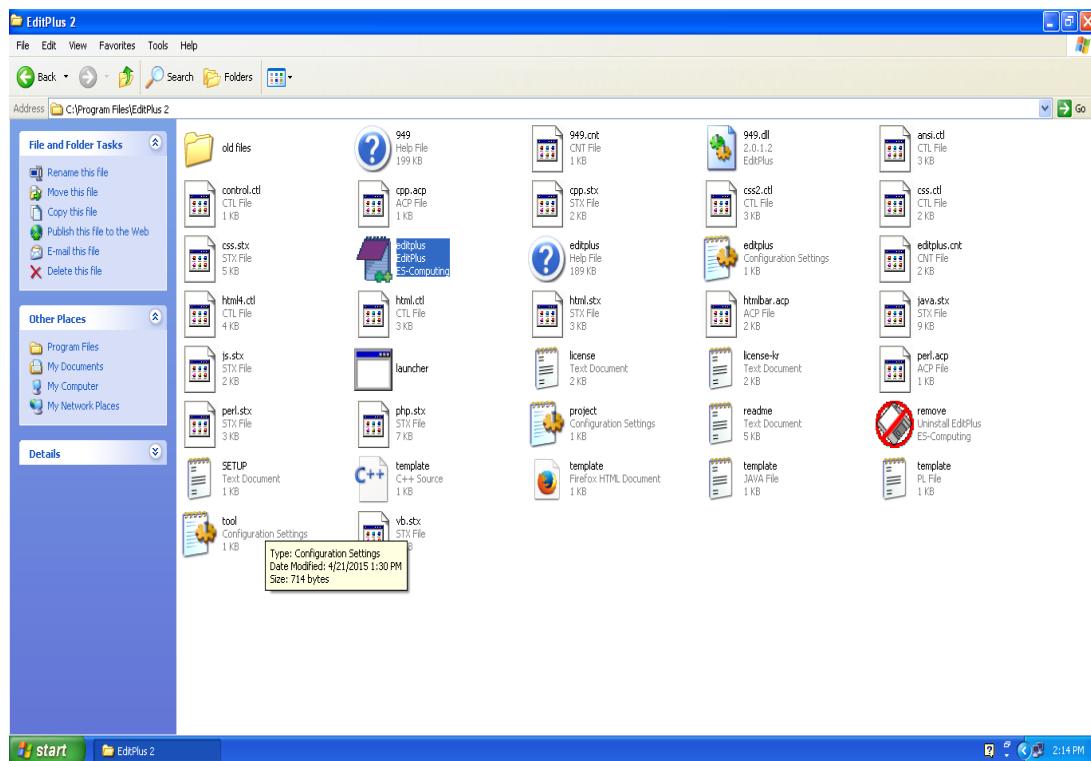


Done with Eclipse....

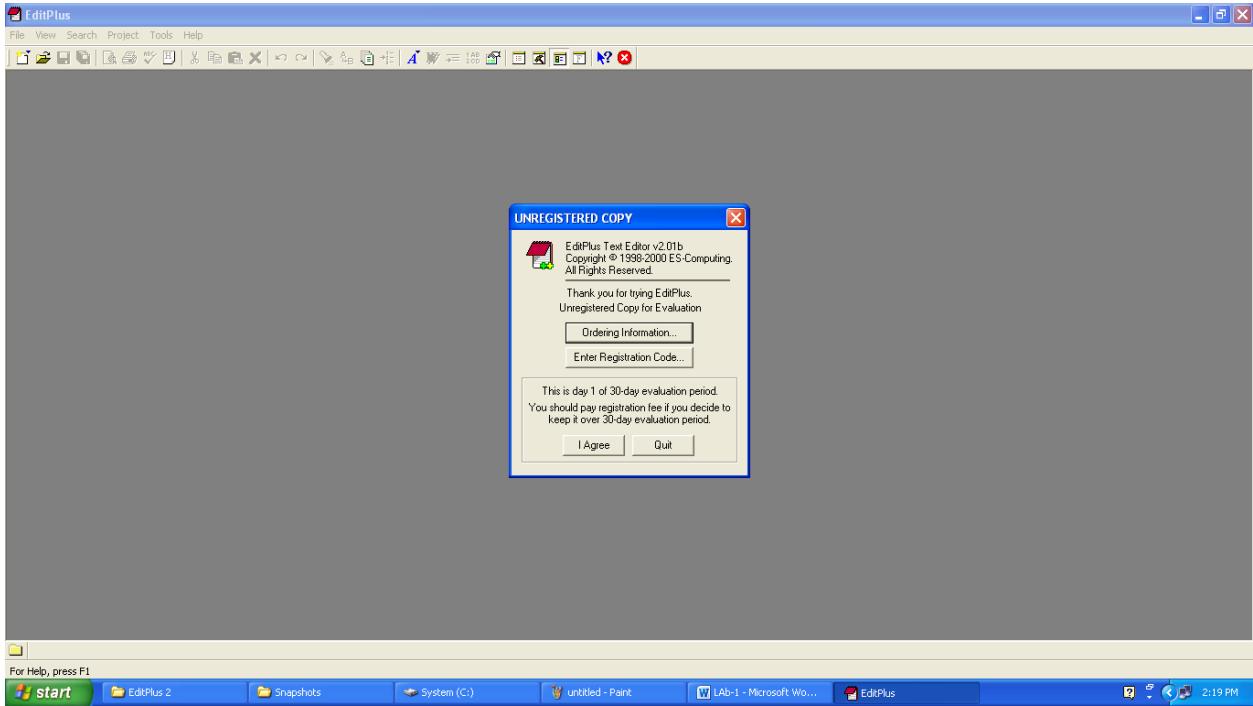
# CORE JAVA

## Lab-1 Assignments Writing First Java Program

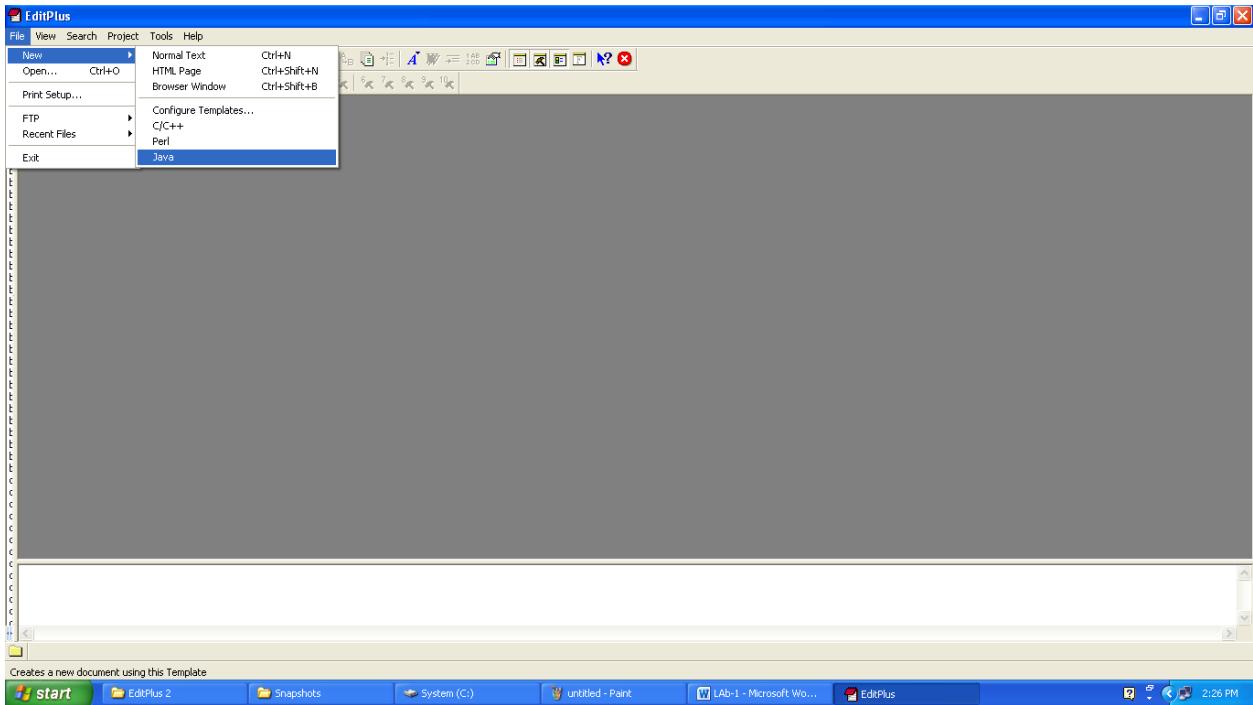
1. Click on start->My Computer-> System ( C ) -> Program Files -> Edit Plus 2..
2. Double click on editplus icon



## Lab Manual



Click on I Agree



Click on  
**File -> New- > Java**  
Write your java program in the editor

## Lab Manual

The screenshot shows a Microsoft Windows XP desktop environment. In the foreground, there is a window titled "EditPlus - [Noname1.java\*]" containing Java code. The code defines a class named "area" with a main method that calculates the area of a circle, rectangle, and square based on user input. A "Save As" dialog box is overlaid on the desktop, prompting the user to save the file to "D:\Data (D:)" with the name "area". The desktop background is the classic Windows XP green field and blue sky. The taskbar at the bottom shows icons for Start, EditPlus 2, Snapshots, System (C:), untitled - Paint, Lab-1 - Microsoft Word, and EditPlus - [Noname1.java\*].

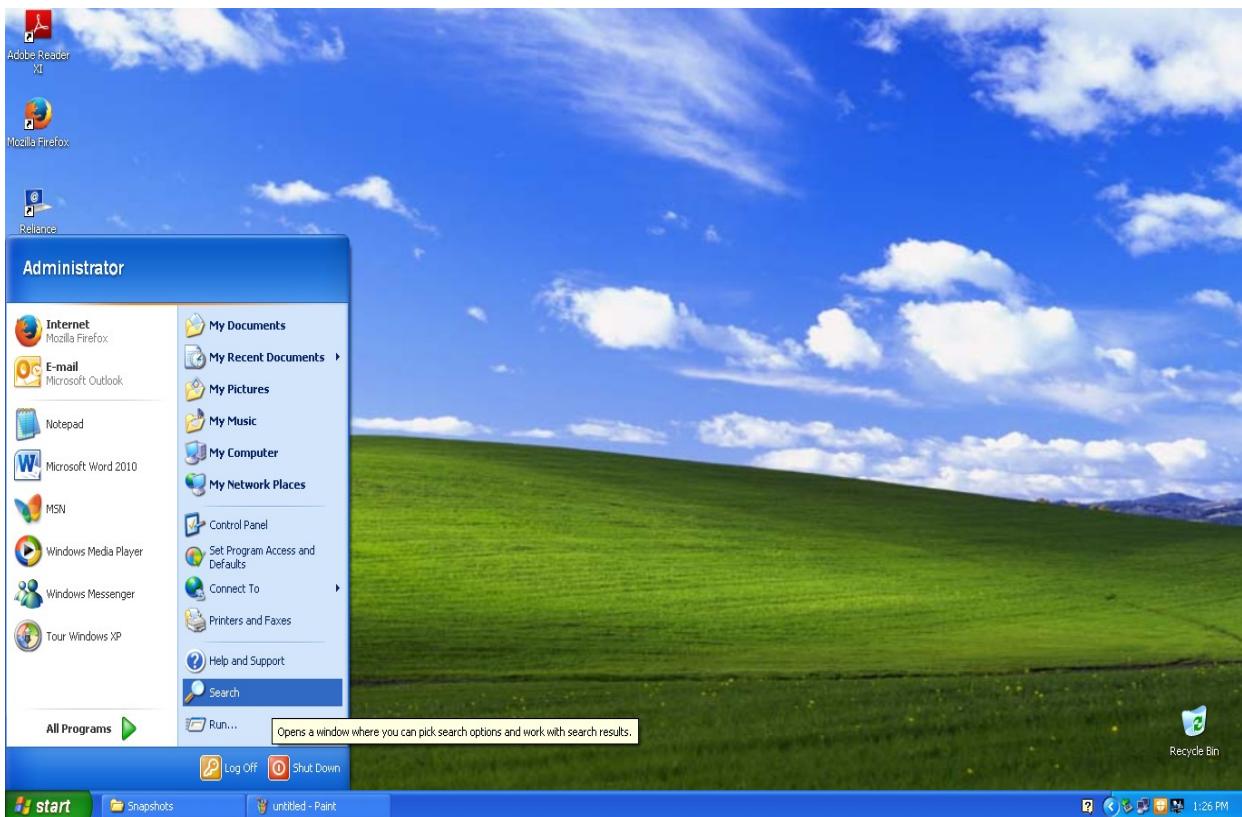
```
1 class area
2 {
3     public static void main(String[] args)
4     {
5         int r,l,b,s;
6         float PI=3.14f;
7         float area;
8
9         r=3;l=5;b=4;s=3;
10        area=PI*r*r ;
11        System.out.println("Area of the circle of radius "+r+" is "+ area);
12        Area = l*b;
13        System.out.println("Area of the rectangle of length"+l+" breadth "+b+" is "+ area);
14        area=s*s;
15        System.out.println("Area of the square of side "+s+" is "+ area);
16    }
17 }
18
```

Save the file as area.java (saved in D drive)

### Compile and Run Java Program on Command Prompt

#### Steps:

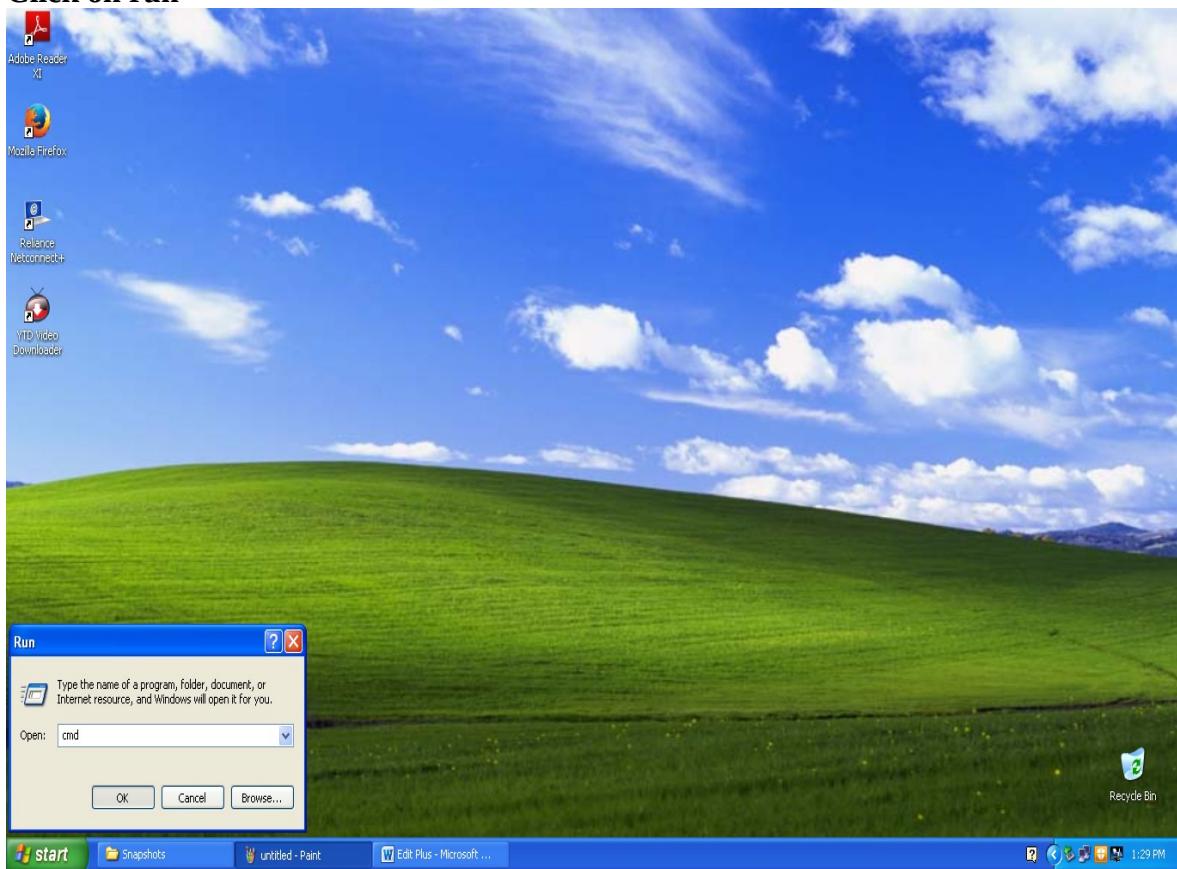
1. Click on Start



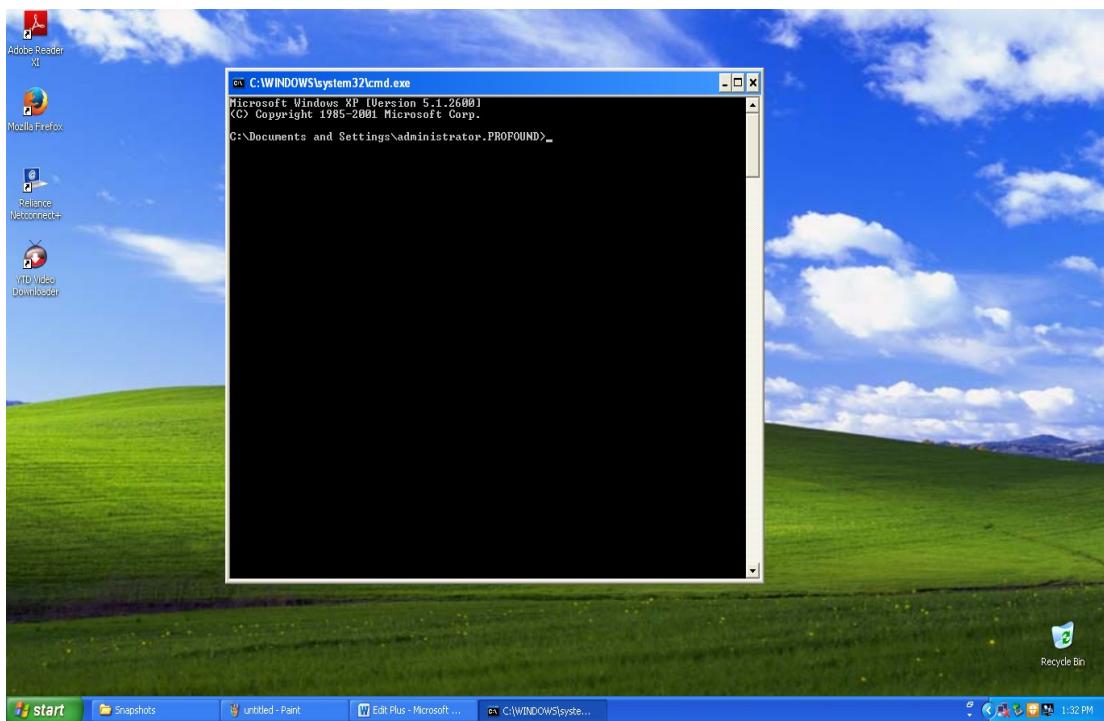
## Lab Manual

---

### 2. Click on run



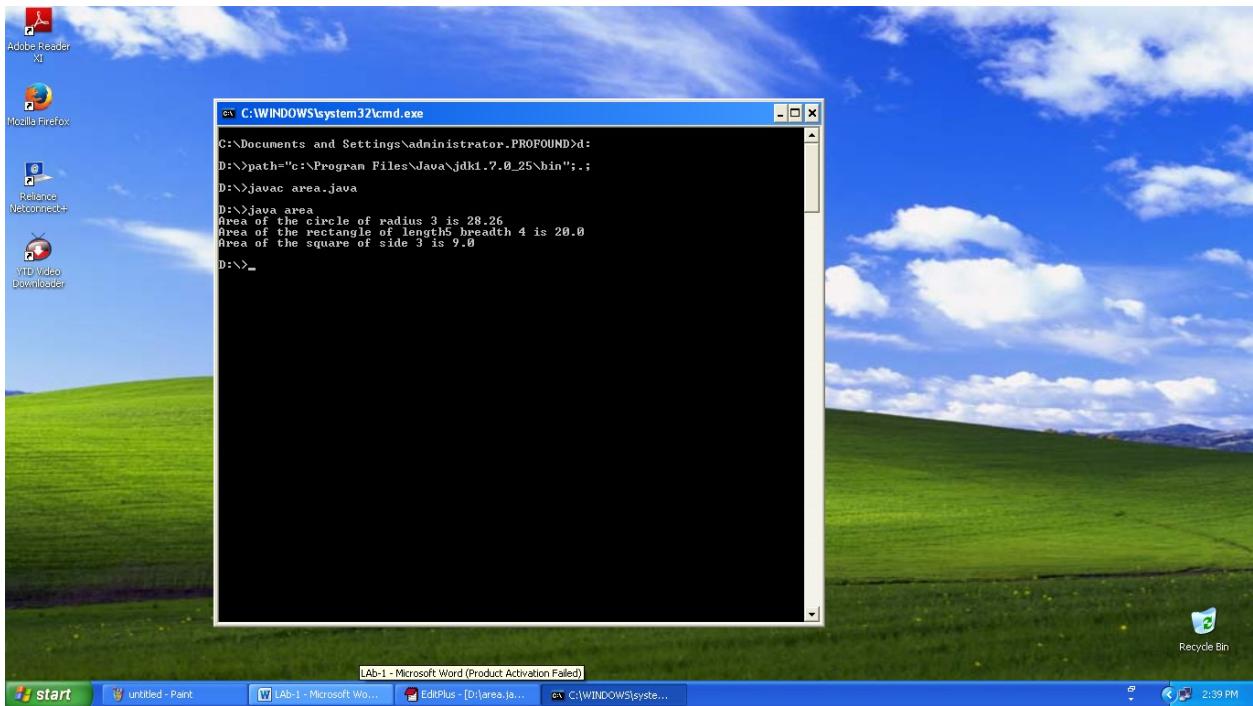
### 3. Type cmd and click on OK Button



4. From the command prompt move to the drive and folder where you have saved your program

like

c:\Documents and Settings\administrator.PROFOUND> d:(press Enter Key)  
d:\>



5: set the path

D:\> path="c:\Program Files\Java\jdk1.7\bin";; // (Press enter key)

6. Compile the source code

D:\>javac area.java

Once compiled generates area.class file

7. Run the .class file

D:\>java area

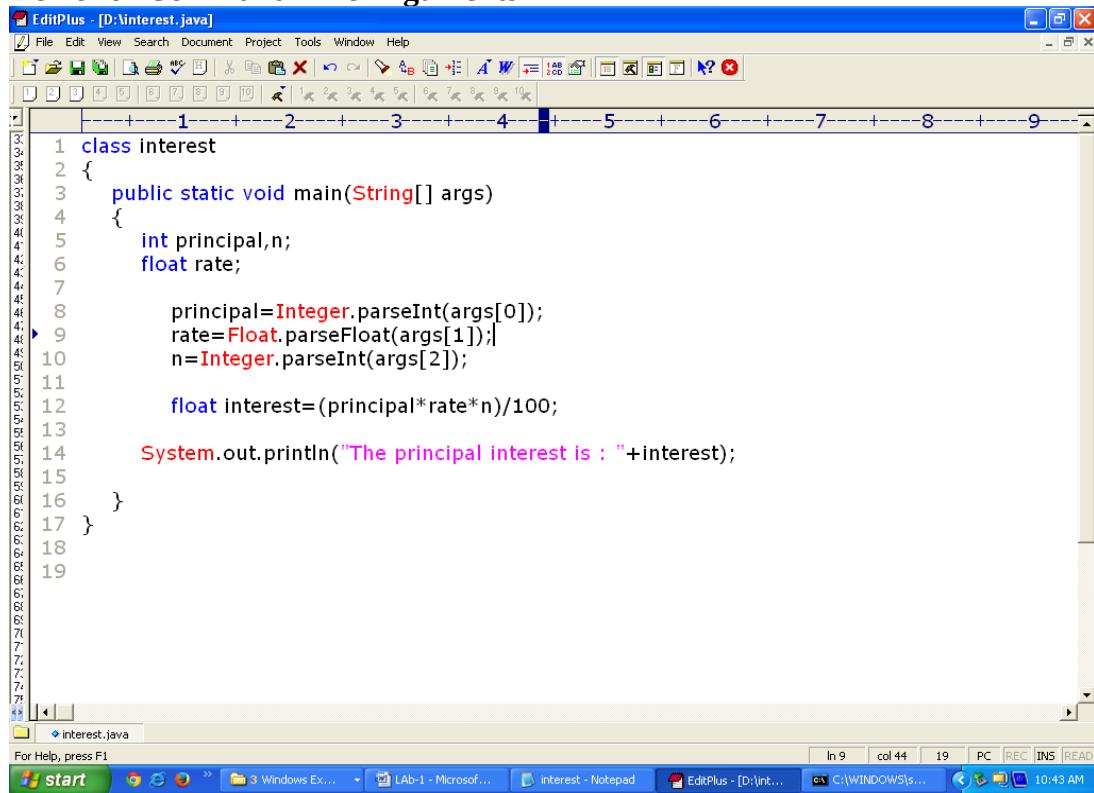
Area of the circle of radius 3 is 28.26

Area of the rectangle of length5 breadth 4 is 20.0

Area of the square of side 3 is 9.0

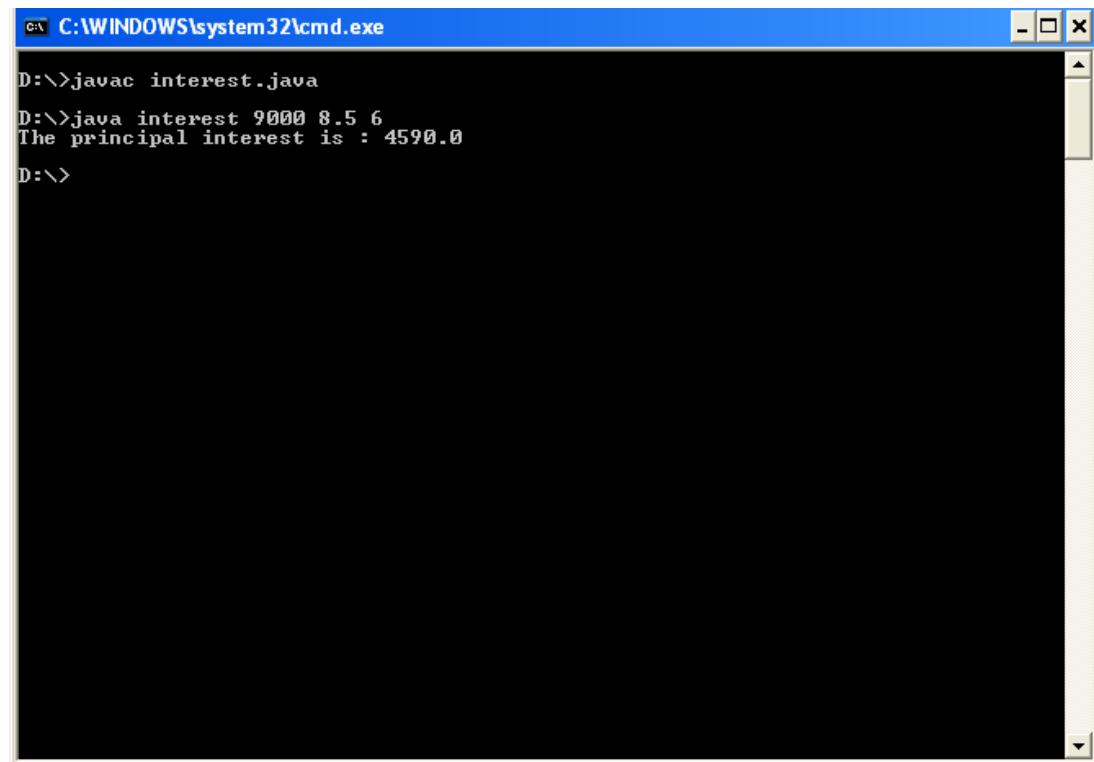
## Lab Manual

### Demo for Command Line Arguments



The screenshot shows the EditPlus text editor with a Java file named "interest.java" open. The code calculates simple interest based on user input for principal, rate, and time.

```
1 class interest
2 {
3     public static void main(String[] args)
4     {
5         int principal,n;
6         float rate;
7
8         principal=Integer.parseInt(args[0]);
9         rate=Float.parseFloat(args[1]);
10        n=Integer.parseInt(args[2]);
11
12        float interest=(principal*rate*n)/100;
13
14        System.out.println("The principal interest is : "+interest);
15
16    }
17 }
```



The screenshot shows a Windows command prompt window titled "cmd.exe" running on the C:\WINDOWS\system32\ directory. The user has typed "javac interest.java" to compile the Java source code, and then "java interest 9000 8.5 6" to run the program with arguments 9000, 8.5, and 6. The output shows the calculated interest as 4590.0.

```
D:\>javac interest.java
D:\>java interest 9000 8.5 6
The principal interest is : 4590.0
D:\>
```

## **Assignments To Solve**

### **\*CLA-----> command Line Argument**

1. accept two integer values via CLA\* and perform all arithmetic operation
2. Modify the above program ,try it with different data types.
3. Accept basic salary via CLA\* and calculate the following

DA=20% of basic

HRA=30% of basic

Calculate gross salary GROSS=BASIC+DA+HRA

Calculate the Income tax (IT) based on the following condition

SAL RANGE	IT
-----------	----

---

Basic >0 and basic < 4000	4%
Basic >=4000 and basic < 10000	6%
Basic >=10000 and basic < 17000	8%
Basic >=17000 and basic < 27000	10%
Basic >=27000 onwards	15%

5. Accept five different values via CLA\* by using for loop and display sum of that values
6. Accept three digit number via CLA\* and calculate the sum of digit
7. Accept values via CLA\* for 1-D array of integer type and display it on screen
8. modify the above program and calculate the sum of number in array

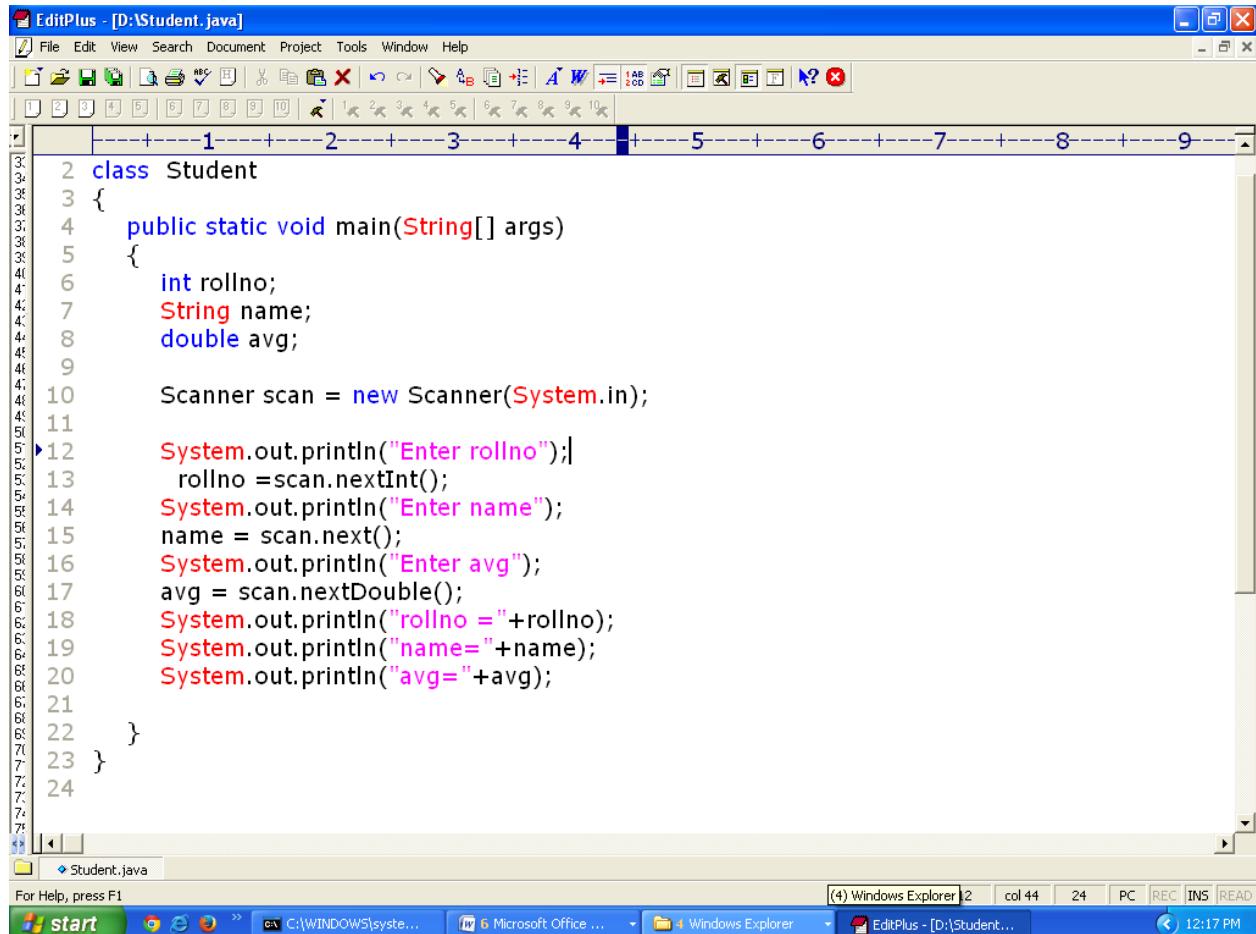
# Lab-2 Assignments

## 1. Classess and Objects

## 2. Using Scanner class

### Introducing Scanner Class To Accept The Values From The User

#### Demo for Scanner Class



The screenshot shows the EditPlus IDE interface with a Java file named "Student.java" open. The code defines a class "Student" with a main method. The main method uses a Scanner object to read input from System.in for rollno, name, and avg, and then prints them back out. The code is as follows:

```
1 class Student
2 {
3     public static void main(String[] args)
4     {
5         int rollno;
6         String name;
7         double avg;
8
9         Scanner scan = new Scanner(System.in);
10
11        System.out.println("Enter rollno");
12        rollno = scan.nextInt();
13        System.out.println("Enter name");
14        name = scan.next();
15        System.out.println("Enter avg");
16        avg = scan.nextDouble();
17        System.out.println("rollno =" + rollno);
18        System.out.println("name=" + name);
19        System.out.println("avg=" + avg);
20
21    }
22 }
23 }
```

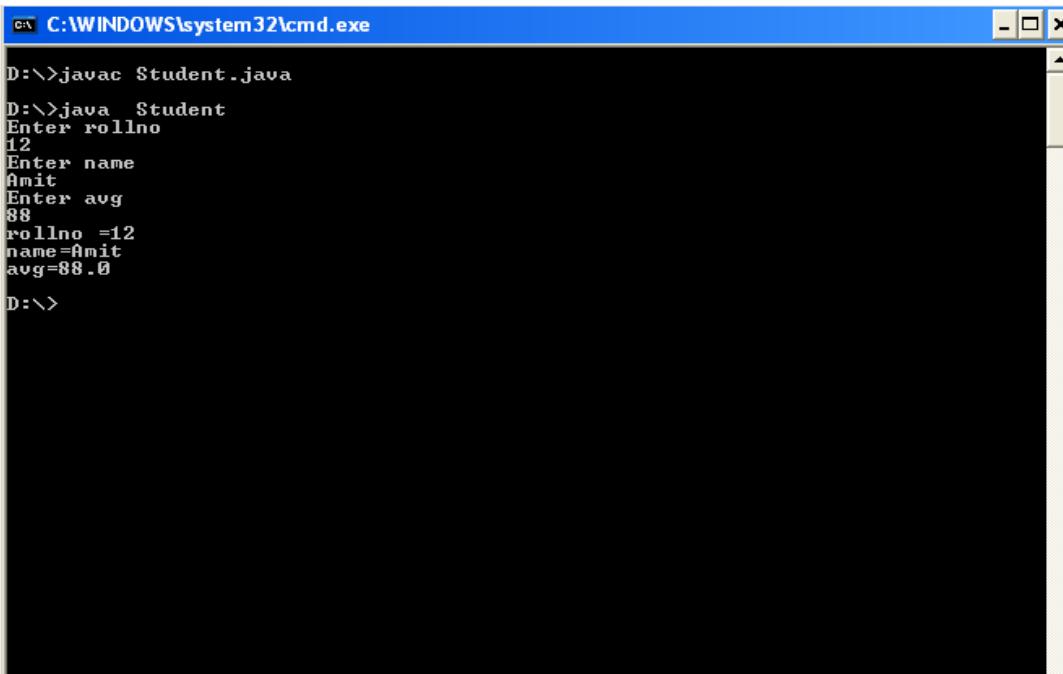
The status bar at the bottom shows the file path "C:\WINDOWS\system32\EditPlus - [D:\Student.java]" and the current time "12:17 PM".

Save as Student.java(in d:\)

Compile and run from Command Prompt

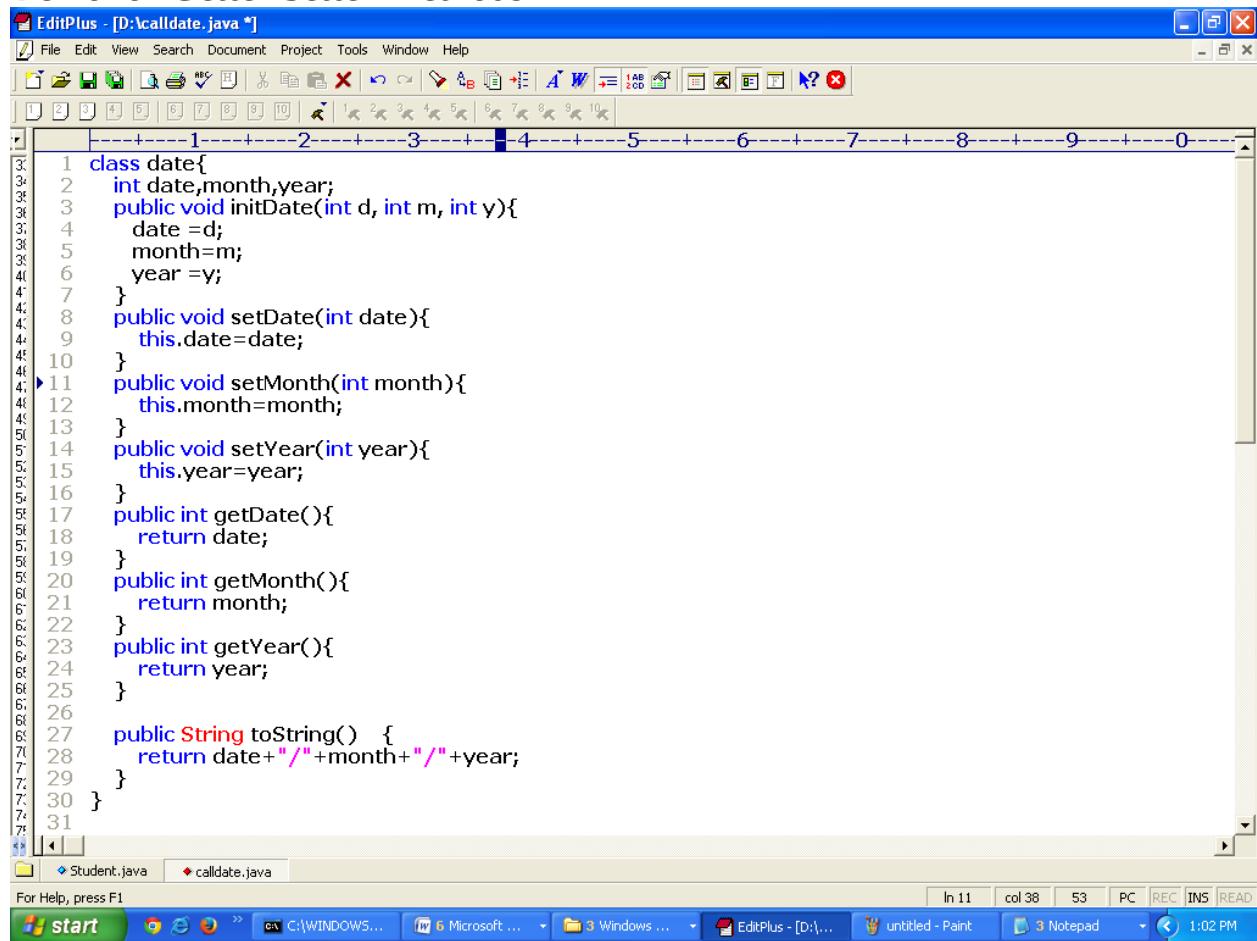
Note: Always set the path if new command prompt is opened( as mentioned in the previous lab demo)

## Lab Manual



```
C:\WINDOWS\system32\cmd.exe
D:\>javac Student.java
D:\>java Student
Enter rollno
12
Enter name
Amit
Enter avg
88
rollno =12
name=Amit
avg=88.0
D:\>
```

### Demo for Getter Setter Methods



```
File Edit View Search Document Project Tools Window Help
EditPlus - [D:\caldate.java *]
1 class date{
2     int date,month,year;
3     public void initDate(int d, int m, int y){
4         date =d;
5         month=m;
6         year =y;
7     }
8     public void setDate(int date){
9         this.date=date;
10    }
11    public void setMonth(int month){
12        this.month=month;
13    }
14    public void setYear(int year){
15        this.year=year;
16    }
17    public int getDate(){
18        return date;
19    }
20    public int getMonth(){
21        return month;
22    }
23    public int getYear(){
24        return year;
25    }
26
27    public String toString() {
28        return date+"/"+month+"/"+year;
29    }
30 }
```

## Lab Manual

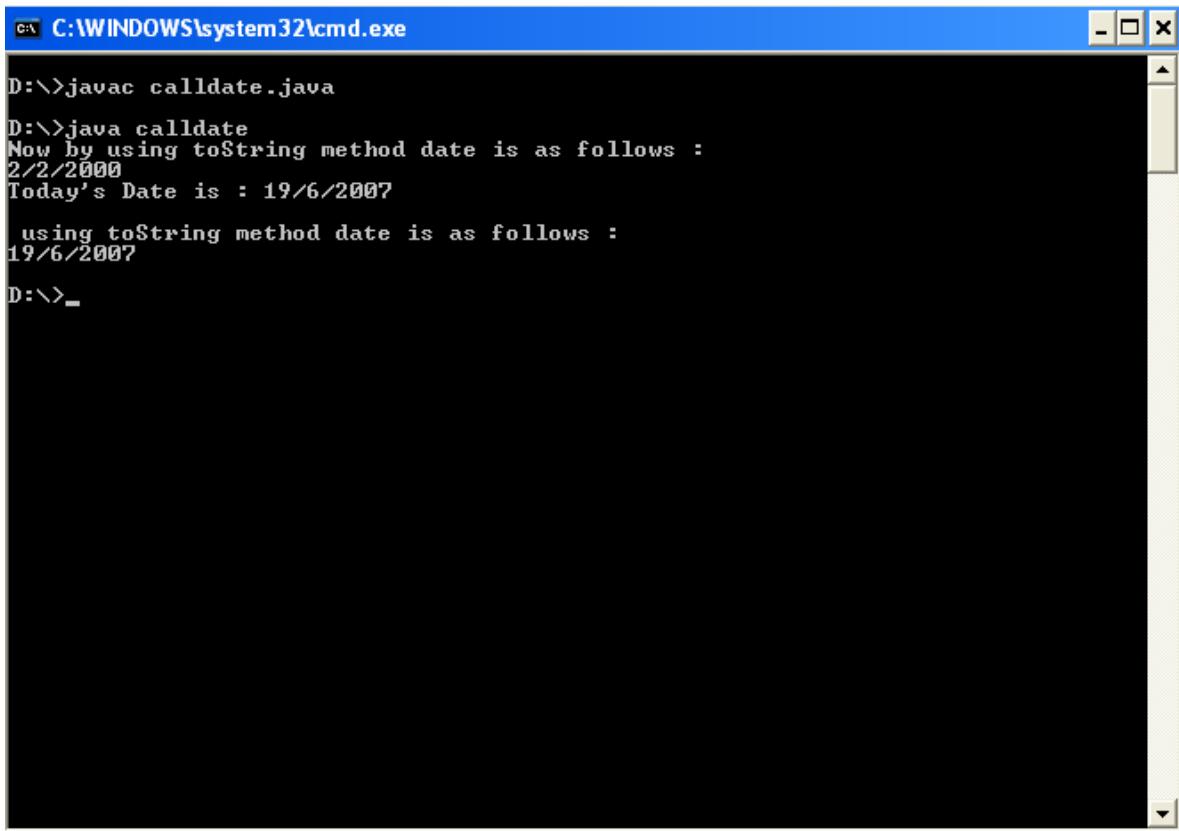
The screenshot shows the EditPlus IDE interface with the file 'calldate.java' open. The code defines a class 'calldate' with a main method. It creates a 'date' object, initializes it to February 2, 2000, prints its string representation, sets the date to June 19, 2007, and then prints the updated date again using both System.out.println and the date's own toString method.

```
60
61 class calldate
62 {
63     public static void main(String[] args)
64     {
65         date d=new date();
66         d.initDate(2,2,2000);
67         System.out.println("Now by using toString method date is as follows : ");
68         System.out.println(d);
69         d.setDate(19);
70         d.setMonth(06);
71         d.setYear(2007);
72
73         System.out.println("Today's Date is : "+d.getDate()+"/"+d.getMonth()+"/"+d.getYear());
74
75         System.out.println();
76         System.out.println(" using toString method date is as follows : ");
77         System.out.println(d);
78     }
79 }
80
```

Save as calldate.java

Compile and run calldate.java

Note: When calldate.java is compiled , date class also gets compiled and date.class and calldate.class files get generated



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window contains the following text:

```
D:>>javac calldate.java
D:>>java calldate
Now by using toString method date is as follows :
2/2/2000
Today's Date is : 19/6/2007
using toString method date is as follows :
19/6/2007
D:>>_
```

### Assignments To Solve :

1. WAP to check whether a person is eligible for voting
2. WAP to check whether a given year is a LEAP year
3. WAP to find the factorial of a number?
4. WAP to find whether a given number is an Armstrong number
5. WAP to reverse the given number
6. Generate the Fib... series 1 1 2 3 5 8 13
7. Create a class Book which describes its Book\_title and Book\_price. Use getter and setter Methods to get & set the Books description. Implement createBook and showBook methods to create and display Book Information. Write a separate class BookInfo to access the Book class

# Lab-3 Assignments

- 1.Introducing Constructor
- 2.OverLoading Constructors
- 3.Overloading Methods/varargs/foreach
- 4.singleton object

The screenshot shows the EditPlus text editor interface with a Java file named 'Addition.java' open. The code defines a class 'Addition' with four overloaded methods: add(int x, int y), add(int x, float y), add(float x, int y), and add(int x, double y, int n). Each method calculates the sum of its parameters and prints the result. The main() method creates an instance of Addition and calls its methods with different argument types and counts.

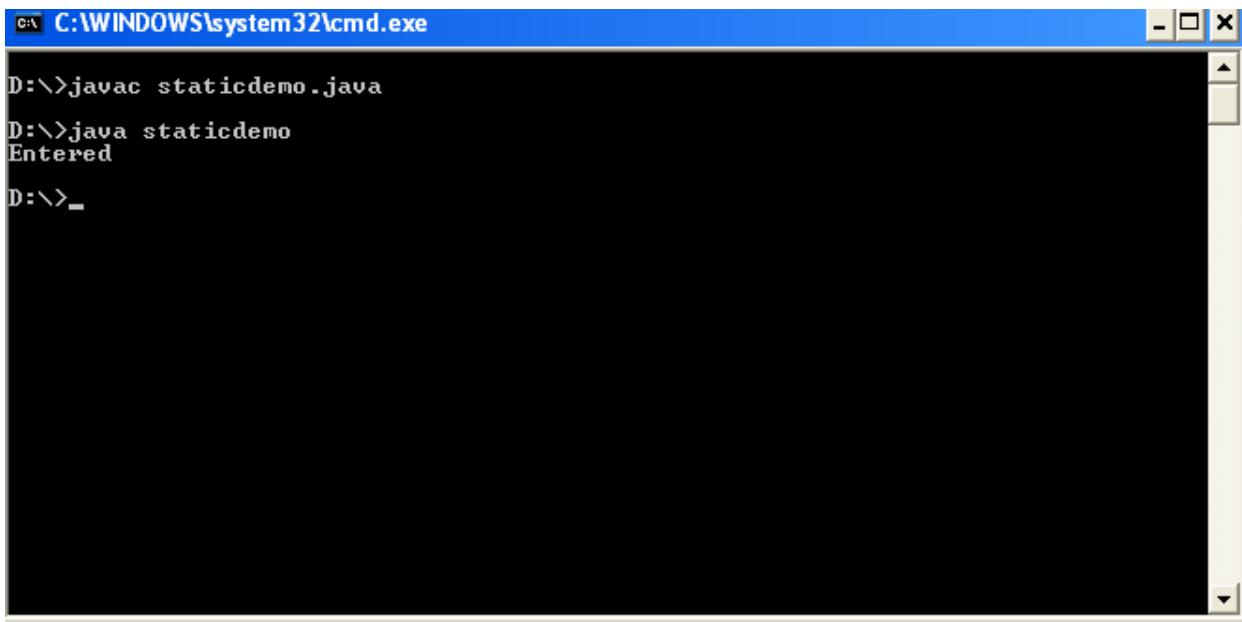
```
1 class Addition
2 {
3     void add(int x,int y)
4     {
5         int z=x+y;
6         System.out.println("adtion of numbers"+ " "+z);
7     }
8     void add(int x,float y)
9     {
10        float z=x+y;
11        System.out.println("adtion of numbers"+ " "+z);
12    }
13    void add(float x,int y)
14    {
15        float z=x+y;
16        System.out.println("adtion of numbers"+ " "+z);
17    }
18    void add(int x,double y,int n)
19    {
20        double z=x+y+n;
21        System.out.println("adtion of numbers"+ " "+z);
22    }
23    public static void main(String[] args)
24    {
25        Addition addition=new Addition();
26        addition.add(2,3);
27        addition.add(2,3.0f);
28        addition.add(2.0f,3);
29        addition.add(2,3,0.6);
30    }
31 }
```

## Lab Manual

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons. Below the taskbar, there is a system tray. In the center, there is a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' showing the execution of 'Addition.java'. Below the command prompt is an 'EditPlus - [D:\staticdemo.java]' window displaying Java code. The code defines two classes: 'xyz' and 'staticdemo'. The 'xyz' class has a static variable 'x' and a constructor that initializes it to a new xyz object. It also has a static method 'getObject()' that prints 'Entered' to the console and returns the static variable 'x'. The 'staticdemo' class has a main method that creates an xyz object using the 'getObject()' method.

```
D:\>javac Addition.java
D:\>java Addition
adtion of numbers 5
adtion of numbers 5.0
adtion of numbers 5.0
adtion of numbers 11.0
D:\>_

-----+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8---+---9---+---0-----
1 class xyz
2 {
3     static xyz x;
4
5     private xyz()
6     {
7     }
8
9     static
10    {
11        x = new xyz();
12    }
13
14     static xyz getObject()
15    {
16         System.out.println("Entered");
17         return x;
18    }
19 }
20 class staticdemo
21 {
22     public static void main(String args[])
23     {
24         xyz p = xyz.getObject();
25     }
26 }
```



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the following text:  
D:\>javac staticdemo.java  
D:\>java staticdemo  
Entered  
D:\>\_

## **Assignments To Solve :**

1. Write an Account class with default constructor, parameterised constructor and methods `toString()`, `deposit(int amt)`, `withdraw(int amt)`. `withdraw` method should take care of insufficient balance. Accept the values from the user..Display the details of various Accounts
2. Write a class `Addition2` with add method (overloading) using varargs and enhanced for loop
3. Declare three array of integer types ,Accept values for two array.Perform arithmetic operation on both array values and store corresponding values in third array. e.g: `c[0]=a[0]+b[0]`
4. Write a class stack. Use methods push and pop to store and retrieve elements from stack(hint: create an array of int to store the elements)
5. Write a singleton class(Singleton class is a class which has only one object)
6. Consider you are working for Xyz Company and developing a payroll software

Construct a class employee with following member using private access specifier:

1. `employeeId` integer
2. `employeeName` string
3. `basicSalary` double

## Lab Manual

---

4. hra	double
5. medical	double
6. pf	double
7. pt	double
8. netSalary	double
9. grossSalary	double

Please use following expressions for calculations.

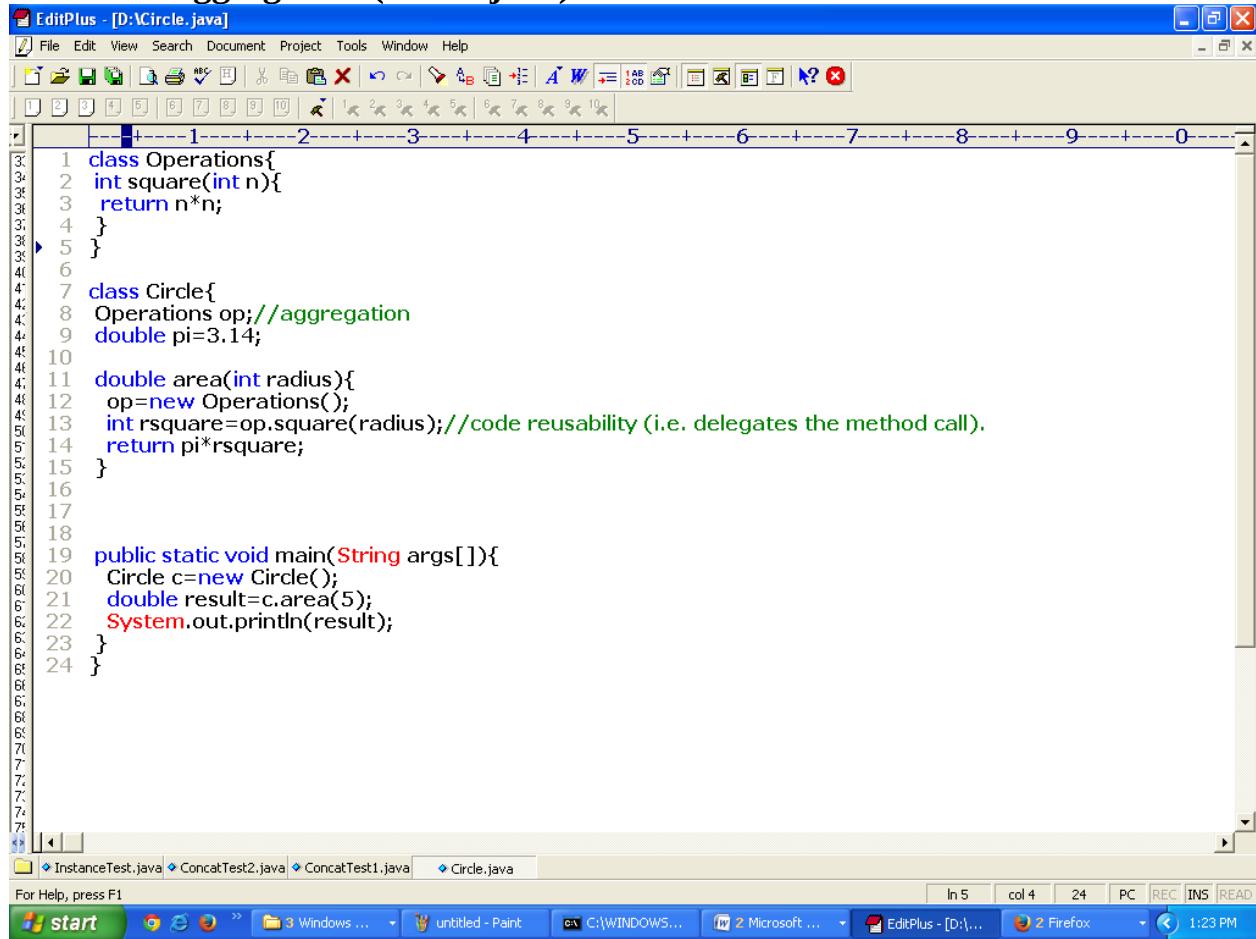
- a. hra=50% of basic salary
- b. pf=12% of basic salary
- c. pt=Rs.200

1. Write no arguments constructor and parameterized constructor to initialize objects.
2. Write properties for employeeId, employeeName, basic salary, netSalary and grossSalary
3. Write methods to display the details of an employee and calculate the gross and the net salary.  
gross salary=basic salary+hra+medical  
net salary=gross salary-(pt+pf)

# Lab-4 Assignments

1. Aggregation/Containment
2. String ,StringBuffer,StringBuilder
3. Math and Wrapper classes
4. Enum

## Demo for Aggregation (Circle.java)



The screenshot shows the Java code for the Circle class. The code uses aggregation to reuse the square method from the Operations class. The code is as follows:

```
1 class Operations{  
2     int square(int n){  
3         return n*n;  
4     }  
5 }  
6  
7 class Circle{  
8     Operations op;//aggregation  
9     double pi=3.14;  
10  
11     double area(int radius){  
12         op=new Operations();  
13         int rsquare=op.square(radius);//code reusability (i.e. delegates the method call).  
14         return pi*rsquare;  
15     }  
16  
17  
18  
19     public static void main(String args[]){  
20         Circle c=new Circle();  
21         double result=c.area(5);  
22         System.out.println(result);  
23     }  
24 }
```

The code editor interface includes tabs for InstanceTest.java, ConcatTest2.java, ConcatTest1.java, and Circle.java. The status bar at the bottom shows the file path C:\WINDOWS..., the line number In 5, column col 4, and the word count 24.

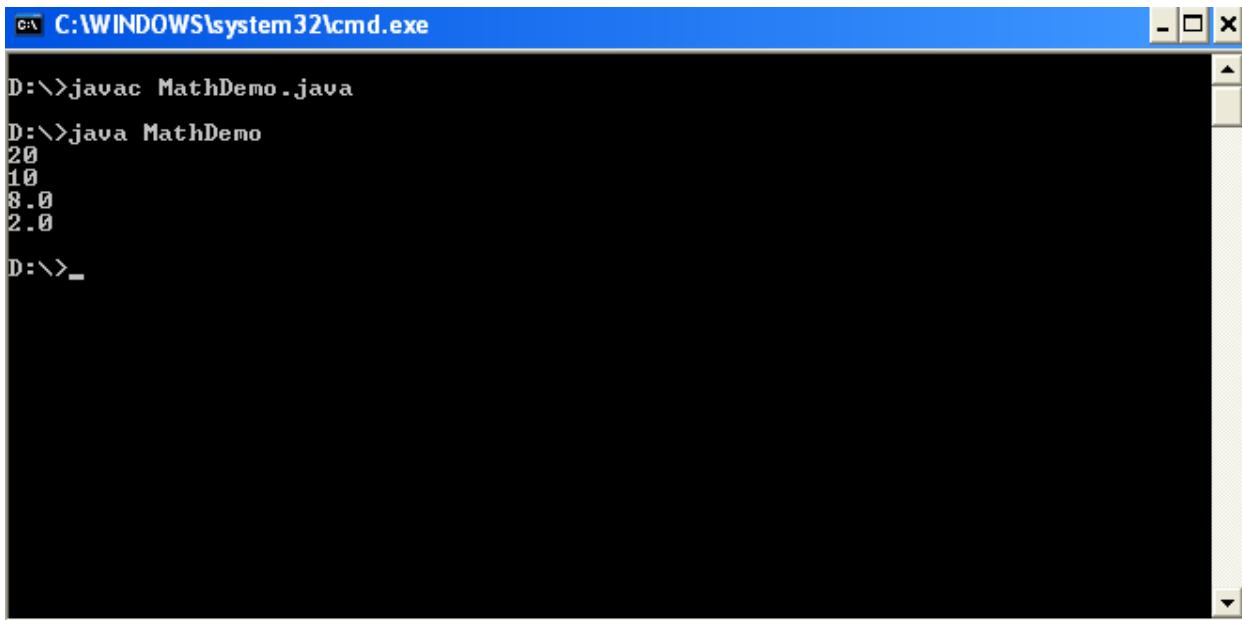
## Lab Manual

```
C:\WINDOWS\system32\cmd.exe
D:\>javac Circle.java
D:\>java Circle
78.5
D:\>_
```

### Demo for using Readymade class methods( Math class)

```
File Edit View Search Document Project Tools Window Help
EditPlus - [D:\MathDemo.java]
D:\>EditPlus - [D:\MathDemo.java]
1 class MathDemo
2 {
3     public static void main(String[] args)
4     {
5         System.out.println(Math.max(10,20));
6         System.out.println(Math.min(10,20));
7         System.out.println(Math.pow(2,3));
8         System.out.println(Math.sqrt(4));
9     }
10 }
11
12 }
```

Note: Use some more methods of Math class to do math operations



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the following command-line session:

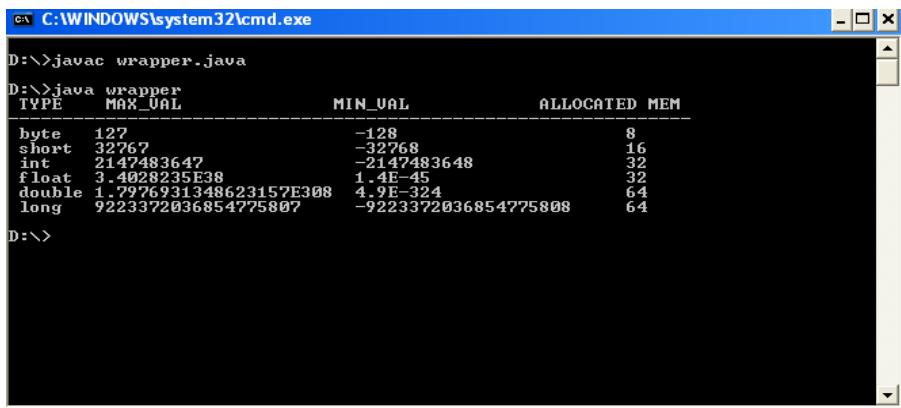
```
D:\>javac MathDemo.java
D:\>java MathDemo
20
10
8.0
2.0
D:\>_
```

## **Assignments To Solve :**

1. Write an Employee class with id, name and dob(Contained Object.. Containment)with Default and parameterised constructor & toString()..
2. Write program to create array of employee object, and print all employee information
3. Use the String Class Methods like length, hashCode>equals, replace, trim, subString, concatinate , compare , charAt, subString etc.. for a given String(s)
4. practice StringBuffer methods such as length(), capacity(),append(),insert(),delete() etc
5. Write a java program to split string by “blank space” and “-” patterns
6. Create string by using below ways and compare using “==” and equals() method:
  - a. String s1=”abc”; //literal way
  - b. String s2=”abc”;
  - c. String s3=new String(“abc”); //using new keyword
  - d. String s4=new String(“abc”);
7. Write a Java program to return the sum of the digits present in the given string. If there is no digits the sum return is 0.
  - a. String s=”14abc9kj”;
  - b. O/p: 1+4+9=14
8. Use Wrapper class methods to print MaxValue, MinValue and Size for various primitive types

## Lab Manual

---



A screenshot of a Windows command prompt window titled "cmd.exe C:\WINDOWS\system32\cmd.exe". The window shows the following text:

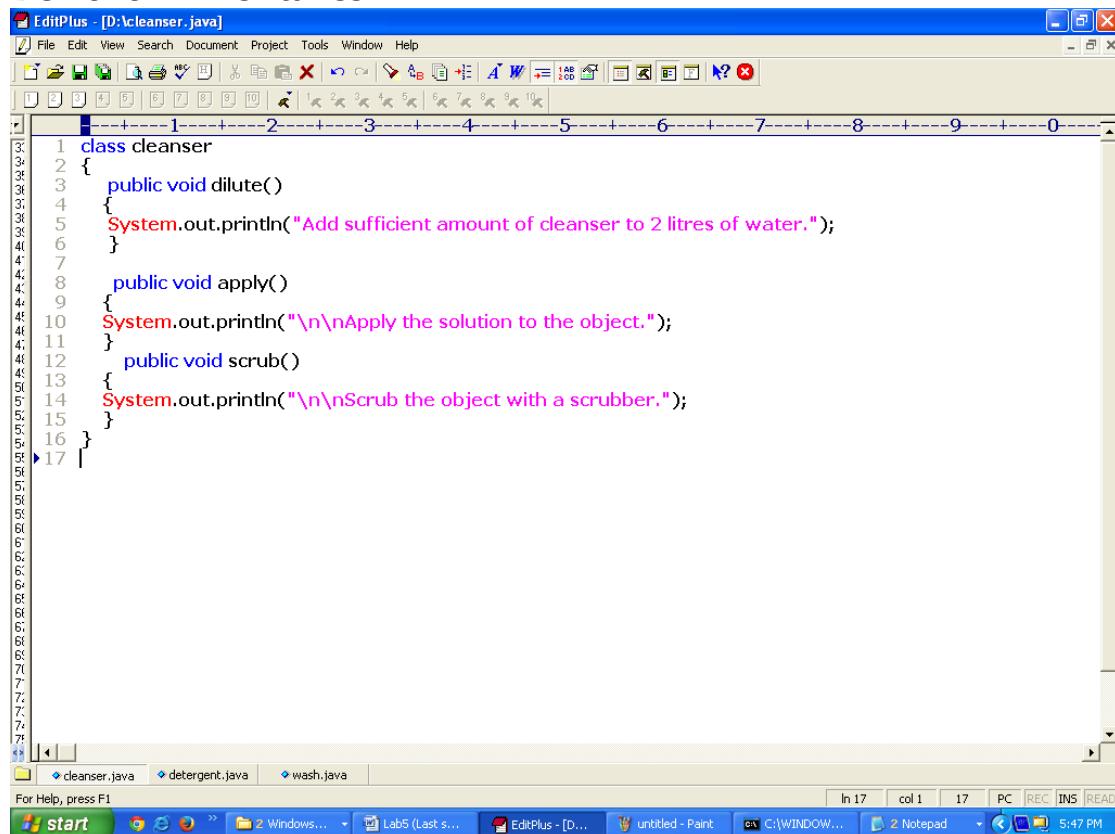
```
D:>javac wrapper.java
D:>java wrapper
      TYPE      MAX_VAL      MIN_VAL      ALLOCATED MEM
byte      127          -128          8
short     32767        -32768        16
int       2147483647    -2147483648   32
float      3.4028235E38  1.4E-45      32
double     1.7976931348623157E308  4.9E-324    64
long      9223372036854775807  -9223372036854775808  64
D:>
```

9. Accept Integer values from user and display it's equivalent Binary, Hexadecimal, Octal values.
10. Declare enum of weekday: Print number with day  
Hint: public enum day{SUNDAY (0), MONDAY (1)}  
Print: 0 SUNDAY 1 MONDAY

# Lab-5 Assignments

## Introducing Inheritance (is a a type of relationship)

### Demo for Inheritance



The screenshot shows the 'EditPlus - [D:\cleanser.java]' window. The code is as follows:

```
1 class cleanser
2 {
3     public void dilute()
4     {
5         System.out.println("Add sufficient amount of cleanser to 2 litres of water.");
6     }
7
8     public void apply()
9     {
10    System.out.println("\n\nApply the solution to the object.");
11 }
12     public void scrub()
13     {
14     System.out.println("\n\nScrub the object with a scrubber.");
15 }
16 }
```

The code defines a class named 'cleanser' with three methods: 'dilute', 'apply', and 'scrub'. Each method prints a specific instruction to the console.

Save as cleanser.java

## Lab Manual

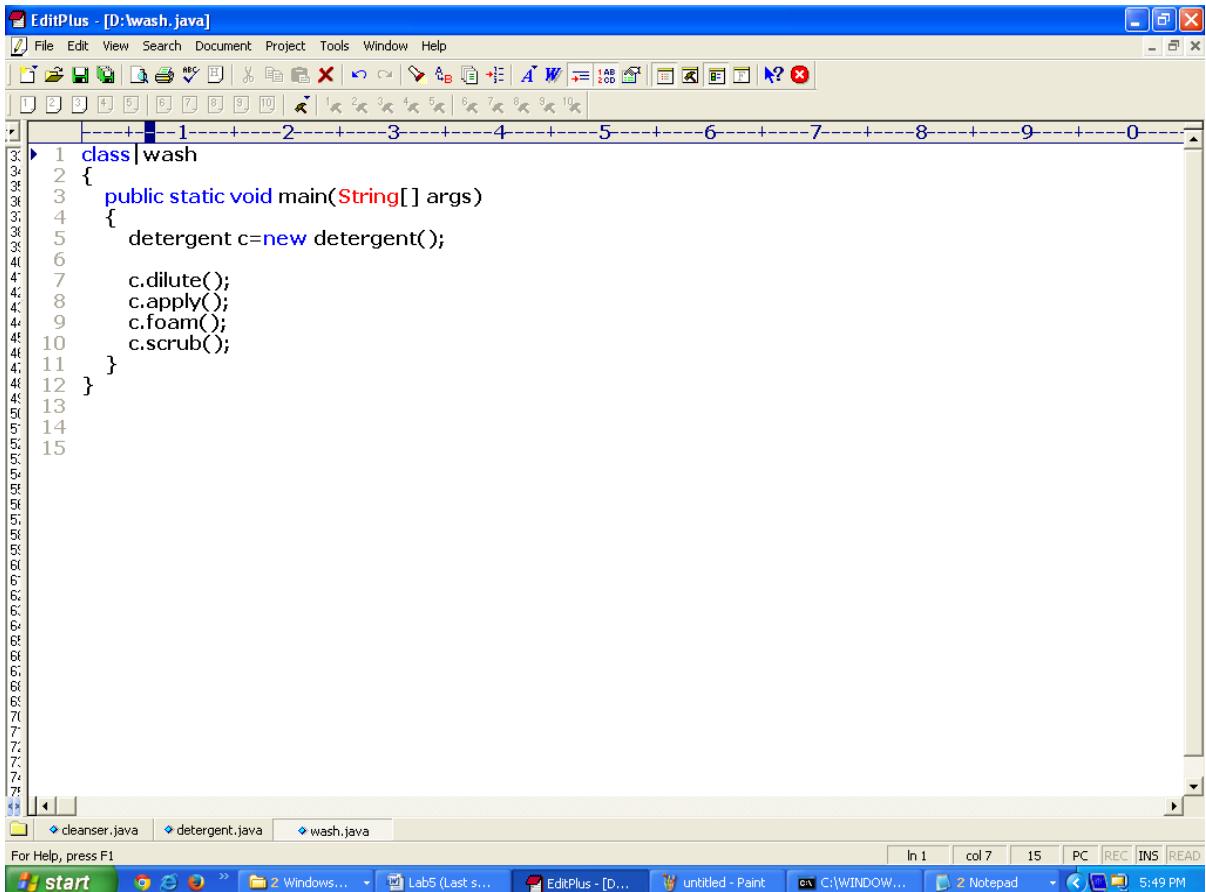
---

The screenshot shows the EditPlus text editor interface. The title bar reads "EditPlus - [D:\detergent.java]". The menu bar includes File, Edit, View, Search, Document, Project, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Find, Copy, Paste, etc. The status bar at the bottom shows "For Help, press F1", "In 16 col 1 17 PC REC INS READ", and the system tray shows icons for Start, Taskbar, and system status.

```
1 class detergent extends cleanser
2 {
3
4     public void foam()
5     {
6         System.out.println("\n\nCheck whether sufficient foam is formed.");
7     }
8
9     public void scrub()
10    {
11        super.scrub();
12        System.out.println("\n\nScrub the area of stains properly. ");
13    }
14
15 }
16
17
```

Save as detergent.java

## Lab Manual

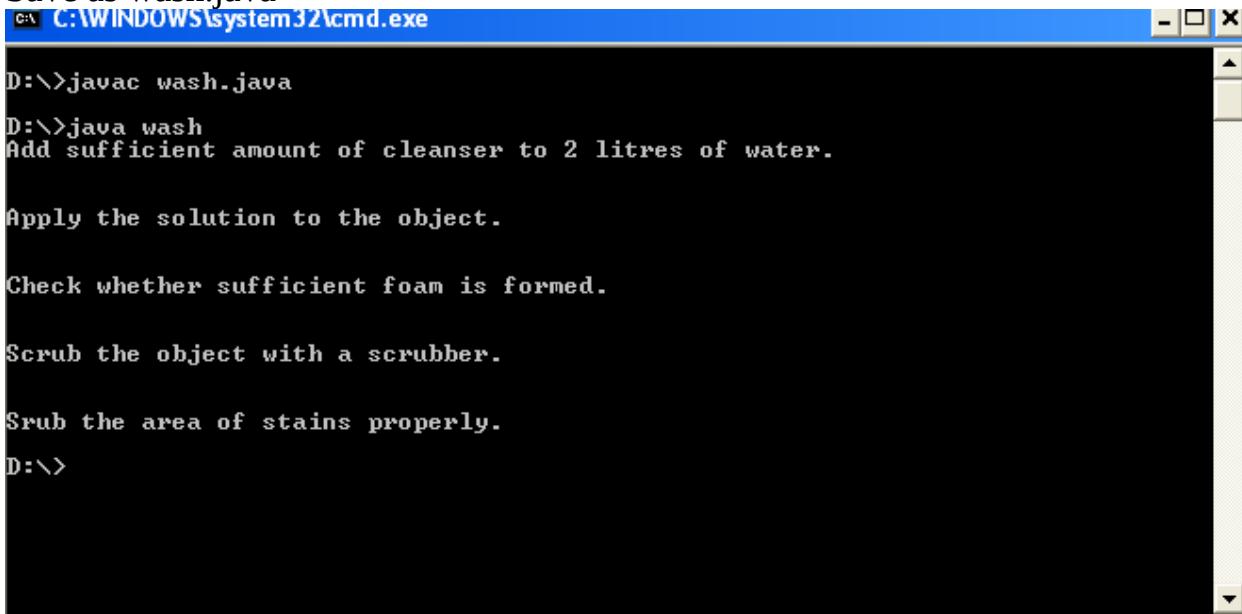


The screenshot shows the 'EditPlus - [D:wash.java]' window. The code in the editor is:

```
1 class wash
2 {
3     public static void main(String[] args)
4     {
5         detergent c=new detergent();
6
7         c.dilute();
8         c.apply();
9         c.foam();
10        c.scrub();
11    }
12 }
```

The status bar at the bottom shows: For Help, press F1, In 1, col 7, 15, PC, REC, INS, READ.

Save as wash.java



The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command entered is 'javac wash.java'. The output is:

```
D:>javac wash.java
D:>java wash
Add sufficient amount of cleanser to 2 litres of water.

Apply the solution to the object.

Check whether sufficient foam is formed.

Scrub the object with a scrubber.

Scrub the area of stains properly.

D:>
```

Demo2

## Lab Manual

The screenshot shows the EditPlus text editor with the file 'C.java' open. The code defines three classes: A, B, and C. Class A has a constructor that prints a message. Class B also has a constructor that prints a message. Class C extends A and overrides its constructor. It creates an instance of B and then prints its own message. The code is as follows:

```
1 class A
2 {
3     A()
4     {System.out.println("\n\nInside A's default constructor.");}
5 }
6
7 class B
8 {
9
10    B()
11    {System.out.println("\n\nInside B's default constructor.");}
12 }
13
14 class C extends A
15 {
16     B b1=new B();
17
18     public static void main(String args[])
19     {
20
21         C c1=new C();
22
23     }
24
25 }
26
```

The status bar at the bottom shows the current file is 'C.java'. The toolbar above the status bar includes icons for file operations like Open, Save, Print, and Find.

Save as C.java

The screenshot shows a Windows command prompt window titled 'cmd.exe' with the path 'C:\WINDOWS\system32'. The user runs 'javac C.java' and then 'java C'. The output shows the program's execution:

```
D:\>javac C.java
D:\>java C

Inside A's default constructor.

Inside B's default constructor.

D:\>
```

### Assignments To Solve :

1. Write class game{ }

```
class boardgame extends game{ }
class chess extends boardgame{ }
class gamedemo{
public static void main(String[] args) {
chess ch=new chess();  }
}
```

(Note: Write only default constructors in each class with specific information . See the order of constructors invoked when chess object is created)

2. Create class **WageEmployee** extending **Employee** class with attributes as hrs (int)and rate(int) and method computeSalary() to calculate the salary.Print the salary and details of WageEmployee.  
(Note: Use the previous date and Employee classes. Accept the values from the user..Default, Parameterised Constructor and toString() to be written in all the classes)
3. Create **SalesPerson** class extending **WageEmployee** with attributes as sales(int) and commission (int). Overide the ComputeSalary()  
in Salesperson class and print the salary and details of SalesPerson
4. Create **Manager** class extending **Employee** class with attributes as fixedsalary(int) and incentives(int) and method computeSalary() to calculate the salary of Manager .Print the salary and details of Manager
5. Write a **TestEmployee** class to print the details of all types of employees  
(use array[] of Employee Objects)

# Lab-6 Assignments

## Introducing Abstract class

### Demo for Abstract class

```
1 abstract class animal {
2     animal()
3     {
4         System.out.println("\nAnimal is the super class");
5     }
6     public abstract void sound();
7 }
8 abstract class herbivores extends animal {
9     herbivores()
10    {
11        System.out.println(" \nherbivorous animals depend on plants for their food");  }}
12
13 abstract class carnivores extends animal{
14     carnivores()
15     {
16         System.out.println(" \ncarnivorous animals depend on other animals for their food");  }
17
18 class pig extends herbivores {
19     pig()
20     {
21         System.out.println("\nPig is a herbivorous animal.");
22     }
23     public void sound()
24     {
25         System.out.println("\nPig grunts.");
26     }
27
28 class tiger extends carnivores {
29     tiger()
30     {
31         System.out.println("\ntiger is a carnivorous animal.");
32     }
33     public void sound()
34     {
35         System.out.println("\n Tiger roars.");  } }
```

The screenshot shows the Java code for an inheritance hierarchy. It starts with an abstract class `animal` which has a constructor and an abstract method `sound()`. Two subclasses, `herbivores` and `carnivores`, inherit from `animal`. The `herbivores` class has a constructor and prints a message about herbivorous animals. The `carnivores` class has a constructor and prints a message about carnivorous animals. The `pig` class inherits from `herbivores` and overrides the `sound()` method to print that pigs grunt. The `tiger` class inherits from `carnivores` and overrides the `sound()` method to print that tigers roar.

Save as animal.java

## Lab Manual

The screenshot shows the EditPlus text editor window with the title "EditPlus - [D:\animaldemo.java]". The code in the editor is:

```
1 class animaldemo
2 {
3     public static void main(String[] args)
4     {
5         animal a;
6
7         a=new pig();
8         a.sound();
9
10        a=new tiger();
11        a.sound();
12    }
13 }
14
15 }
```

The status bar at the bottom shows "For Help, press F1" and "In 15 col 1 15 PC REC INS READ". The taskbar below the window lists several icons including "start", "day7-1", "Lab Manual ...", "Lab6 - Micro...", "animal - Not...", "EditPlus - [D...", "untitled - Paint", and the system clock "10:45 AM".

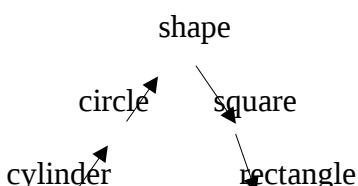
Sava as animaldemo.java

The screenshot shows a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The command history and output are as follows:

```
D:\>javac animal.java
D:\>javac animaldemo.java
D:\>java animaldemo
Animal is the super class
herbivorous animals depend on plants for their food
Pig is a herbivorous animal.
Pig grunts.
Animal is the super class
carnivorous animals depend on other animals for their food
tiger is a carnivorous animal.
Tiger roars.
D:\>_
```

## **Assignments To Solve**

1. Create an abstract class Instrument which is having the abstract function play. Create three more sub classes from Instrument which is Piano, Flute, Guitar. Override the play method inside all three classes printing a message .  
“Piano is playing tan tan tan tan” for Piano class  
“Flute is playing toot toot toot toot” for Flute class  
“Guitar is playing tin tin tin” for Guitar class  
You must not allow the user to declare an object of Instrument class.  
Create an array of 10 Instruments.  
Assign different type of instrument to Instrument reference.  
Check for the polymorphic behavior of play method.  
Use the instanceof operator to print that which object stored at which index of instrument array.
  
2. Create a class Medicine to represent a drug manufactured by a pharmaceutical company. Provide a function displayLabel() in this class to print Name and address of the company.  
Derive Tablet, Syrup and Ointment classes from the Medicine class. Override the displayLabel() function in each of these classes to print additional information suitable to the type of medicine. For example, in case of tablets, it could be “store in a cool dry place”, in case of ointments it could be “for external use only” etc.  
Create a class TestMedicine. Write main function to do the following:  
Declare an array of Medicine references of size 10  
Create a medicine object of the type as decided by a randomly generated integer in the range 1 to 3.  
Refer Java API Documentation to find out random generation feature.  
Check the polymorphic behavior of the displayLabel() method.
  
3. Create an abstract class shape with abstract method  
void area();  
create 4 more classes circle, cylinder, square and rectangle



Override the area() in all the classes  
Create an array of references of type shape in TestShape class and print the area of different types of shapes.

# Lab-7 Assignments

## Introducing Interfaces and Packages

The screenshot shows the EditPlus text editor interface with the file 'printer.java' open. The code implements the Printer interface, which defines methods for printing lines and stars. It also implements the Testable interface, which defines a method for testing coke. The main method creates a Printer object and calls its methods.

```
1 interface printable{
2     public void printLine();
3     public void printStar();
4 }
5
6 interface testable{
7     public void testCoke();
8 }
9
10 class printer implements printable,testable{
11     public void printLine() {
12         System.out.println("-----");
13     }
14
15     public void printStar() {
16         System.out.println("*****");
17     }
18
19     public void testCoke() {
20         System.out.println("Testing....please wait");
21     }
22
23     public static void main(String[] args) {
24         printer print=new printer();
25         print.printLine();
26         print.testCoke();
27         print.printStar();
28     }
29 }
```

Save as printer.java

Note: three .class files get created

printable.class

testable.class

printer.class

## Lab Manual

```
C:\WINDOWS\system32\cmd.exe
D:\>javac printer.java
D:\>java printer
Testing....please wait
*****
D:\>_
```

### Demo2

```
1 class nestedinterface {
2     int a;
3
4     nestedinterface(int a) {
5         this.a=a;
6     }
7
8     interface print {
9         void print(String str);
10    }
11
12 class nesteddemo extends nestedinterface implements nestedinterface.print{
13     int b;
14     nesteddemo(int a,int b)
15     {
16         super(a);
17         this.b=b;
18     }
19
20     public void print(String str){
21         System.out.println(str+a+" & "+b+".");
22     }
23
24     String getdata(){
25         return "The value of a & b : ";
26     }
27
28     public static void main(String args[]){
29         nesteddemo nd=new nesteddemo(20,53);
30         nd.print(nd.getdata()); }}
```

Save as nesteddemo.java

```
cmd C:\WINDOWS\system32\cmd.exe
D:\>javac nesteddemo.java
D:\>java nesteddemo
The value of a & b : 20 & 53.
D:\>
```

## Assignments To Solve

1. Create an interface relational  
interface relational

```
{  
void greaterThan();  
void lessThan();  
void greaterThanEq();  
void lessThanEq();  
}
```

Write an implementing class implRel which implements relational  
class implRel implements relational

```
{  
int a, b;  
implRel(int a,int b)  
{  
this.a=a;  
this.b=b;  
}  
- - -  
}
```

Create a class relationdemo .In main create an object of implRel (implementing class) and invoke all the methods ...

2. Create an interface  
interface MyMath

```
{  
double sqr(double a);  
double cube(double a);  
double cosine(double a);  
double sine(double a);  
}
```

Write a class **implMath** which implements **MyMath** interface.

Write another class **MathDemo** ..In main create a reference of **MyMath** pointing to **implMath** class and invoke the methods..Accept the values from the user..

3. Create 2 classes **Student** and **Batch**, Student class is in pack1n and Batch class is in pack2. Write a **PrintData** class to print Student and Batch information.
4. Use the Student and Batch classes created earlier. it should contain public, protected, private and default attributes and using Batch class, Check accessibility of their attributes.
5. Create a package com.user .

Now create a Greeter class in this package having the following features:

Attributes:

name string //indicates name of the person to be greeted

Member functions:

Greeter(aName) //constructor to initialize the name of the  
//person to be greeted by this greeter.

sayHello() //returns a hello message with the name of the  
//person initialized earlier.

sayGoodBye() //bids goodbye to the person named earlier.

Create another class in the same package called Advisor

Attributes:

message string[5] //contains five advice messages

Member functions:

Advisor() //default constructor to initialize an array of  
//strings with atleast five advice messages

getAdvice() //randomly selects an advice from the available  
//list of messages and returns it to the caller of this method

Outside the package, from your working directory, create a class GreeterTest that constructs Greeter objects for all command-line arguments and prints out the results of calling sayHello().

The program should then display an advice and finally bid goodbye to each of the persons/entities in reverse order of the names entered at the command line.

java GreeterTest Mars Venus

then the program should print

Hello, Mars!

Hello, Venus!

Advice: Never say No

Goodbye Venus!

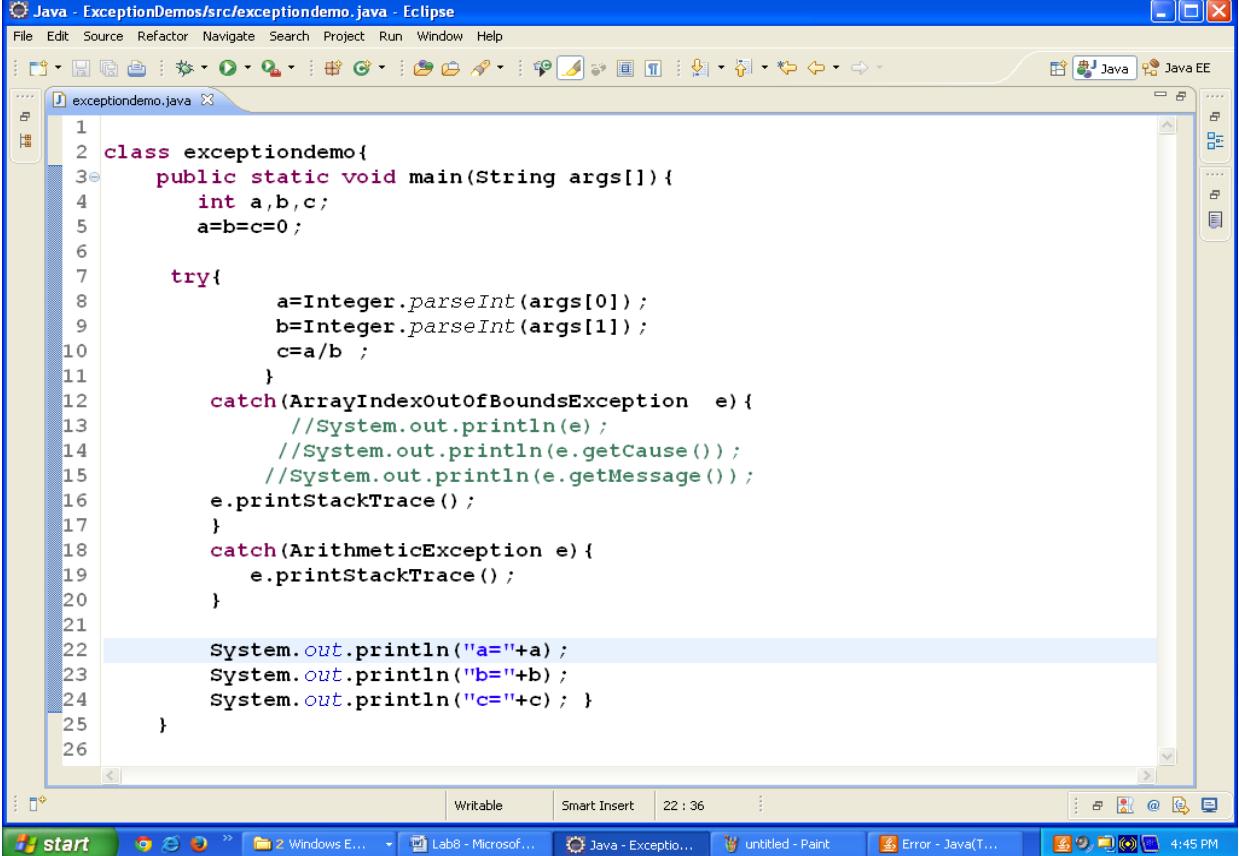
Goodbye Mars!

# Lab-8 Assignments

## Exception Handling

### Demo1

create exceptiondemo class in Eclipse ID



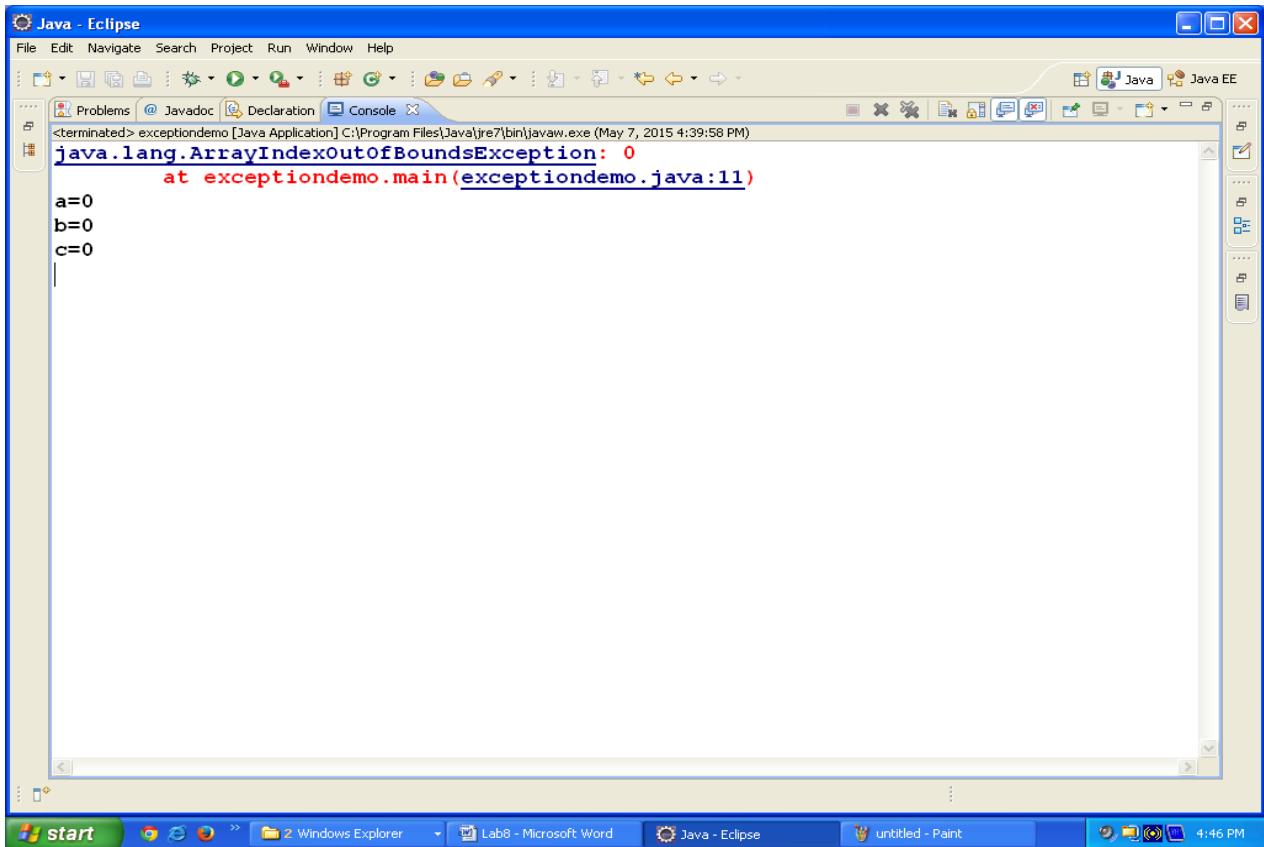
The screenshot shows the Eclipse IDE interface with the file `exceptiondemo.java` open. The code implements exception handling for division by zero. It reads two integers from command-line arguments, performs division, and prints the results. If an `ArrayIndexOutOfBoundsException` occurs (due to empty arguments), or an `ArithmaticException` occurs (due to division by zero), it prints the cause and stack trace.

```
Java - ExceptionDemos/src/exceptiondemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
exceptiondemo.java
1
2 class exceptiondemo{
3     public static void main(String args[]){
4         int a,b,c;
5         a=b=c=0;
6
7         try{
8             a=Integer.parseInt(args[0]);
9             b=Integer.parseInt(args[1]);
10            c=a/b ;
11        }
12        catch(ArrayIndexOutOfBoundsException e){
13            //System.out.println(e);
14            //System.out.println(e.getCause());
15            //System.out.println(e.getMessage());
16            e.printStackTrace();
17        }
18        catch(ArithmaticException e){
19            e.printStackTrace();
20        }
21
22        System.out.println("a="+a);
23        System.out.println("b="+b);
24        System.out.println("c="+c); }
25    }
26
```

Run as -> Java Application

## Lab Manual

---



### Demo2

create userdefined exception class **myError** and **myExceptiondemoclass**

## Lab Manual

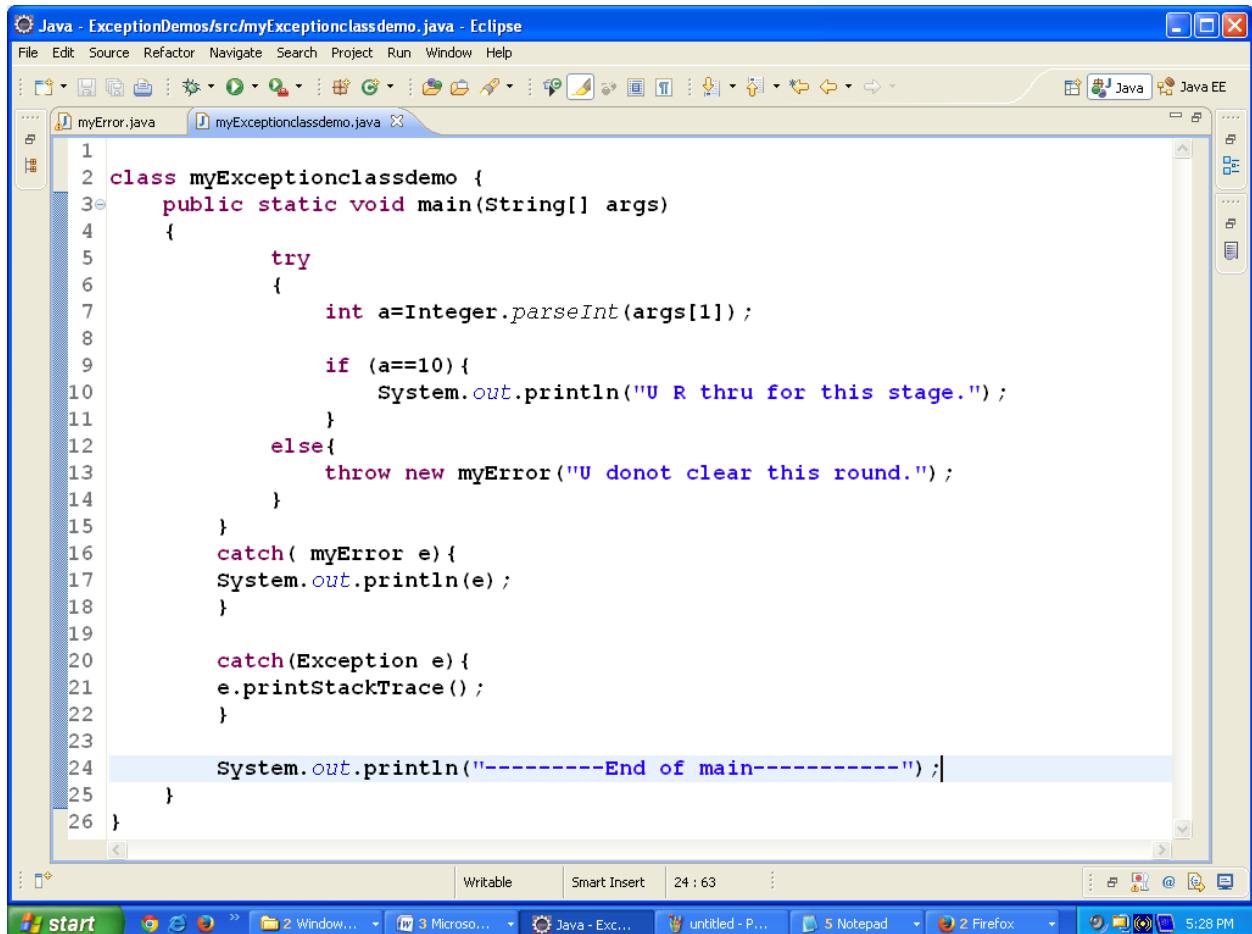
---

The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/myError.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, and Find. The left sidebar shows the project structure with "myError.java" selected. The main editor window contains the following Java code:

```
1  class myError extends Exception
2
3  {
4      myError(String str)
5      {
6          super(str);
7          System.out.println("Oh! u entered the wrong number.");
8      }
9
10 }
11
```

The status bar at the bottom shows "Writable", "Smart Insert", and "11 : 1". Below the status bar is the Windows taskbar with icons for Start, Internet Explorer, Microsoft Word, Java - Exc..., Notepad, and Firefox, along with the system clock showing 5:25 PM.

## Lab Manual



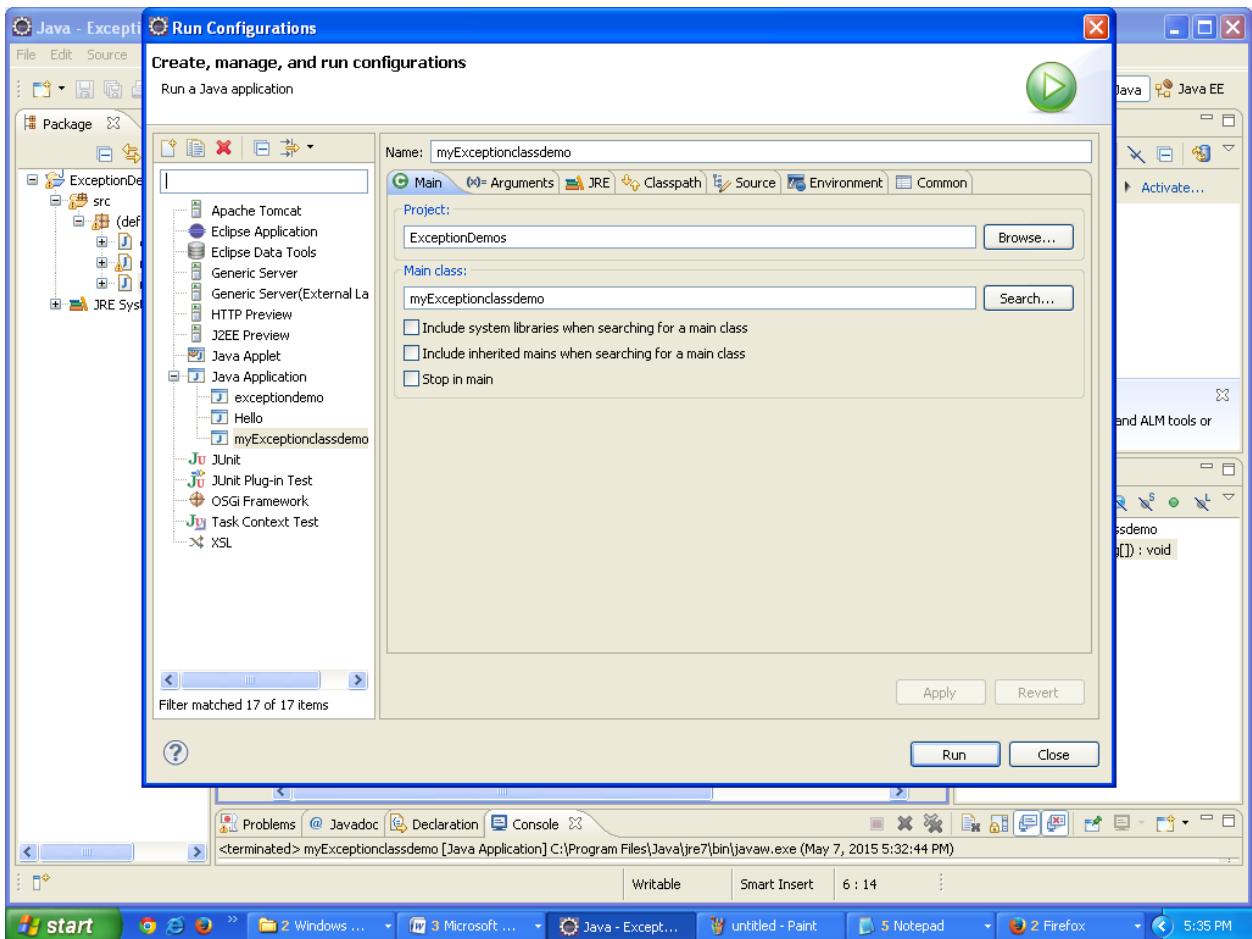
The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/myExceptionclassdemo.java - Eclipse". The main window displays Java code for a class named "myExceptionclassdemo". The code includes a main method that attempts to parse a command-line argument into an integer. If the value is 10, it prints a message. Otherwise, it throws a custom exception "myError" with a specific error message. The code then catches both "myError" and the general "Exception" to print the stack trace. Finally, it prints a message indicating the end of the main method. The code is as follows:

```
1  class myExceptionclassdemo {  
2      public static void main(String[] args)  
3      {  
4          try  
5          {  
6              int a=Integer.parseInt(args[1]);  
7  
8              if (a==10){  
9                  System.out.println("U R thru for this stage.");  
10             }  
11             else{  
12                 throw new myError("U donot clear this round.");  
13             }  
14         }  
15         catch( myError e){  
16             System.out.println(e);  
17         }  
18  
19         catch(Exception e){  
20             e.printStackTrace();  
21         }  
22  
23         System.out.println("-----End of main-----");  
24     }  
25 }  
26 }
```

Steps for passing CLA(command Line Arguments in eclipse)

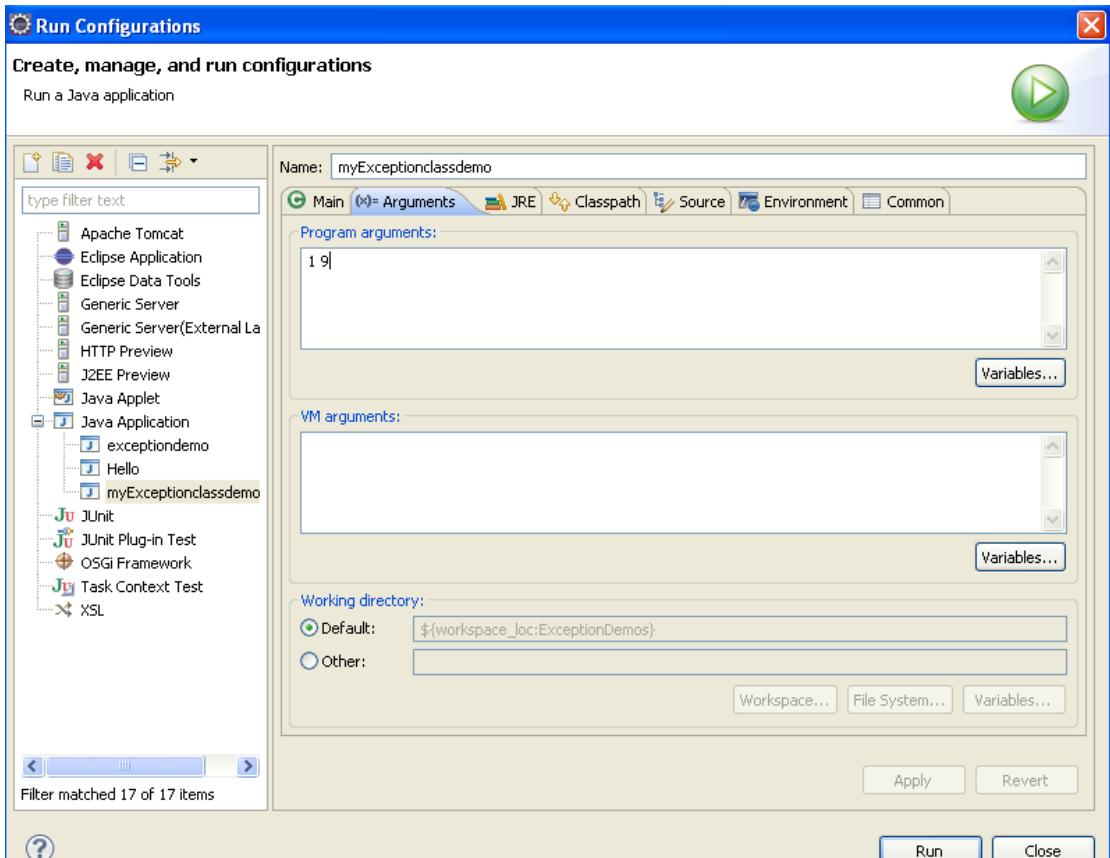
Run as -> Run Configurations...

## Lab Manual



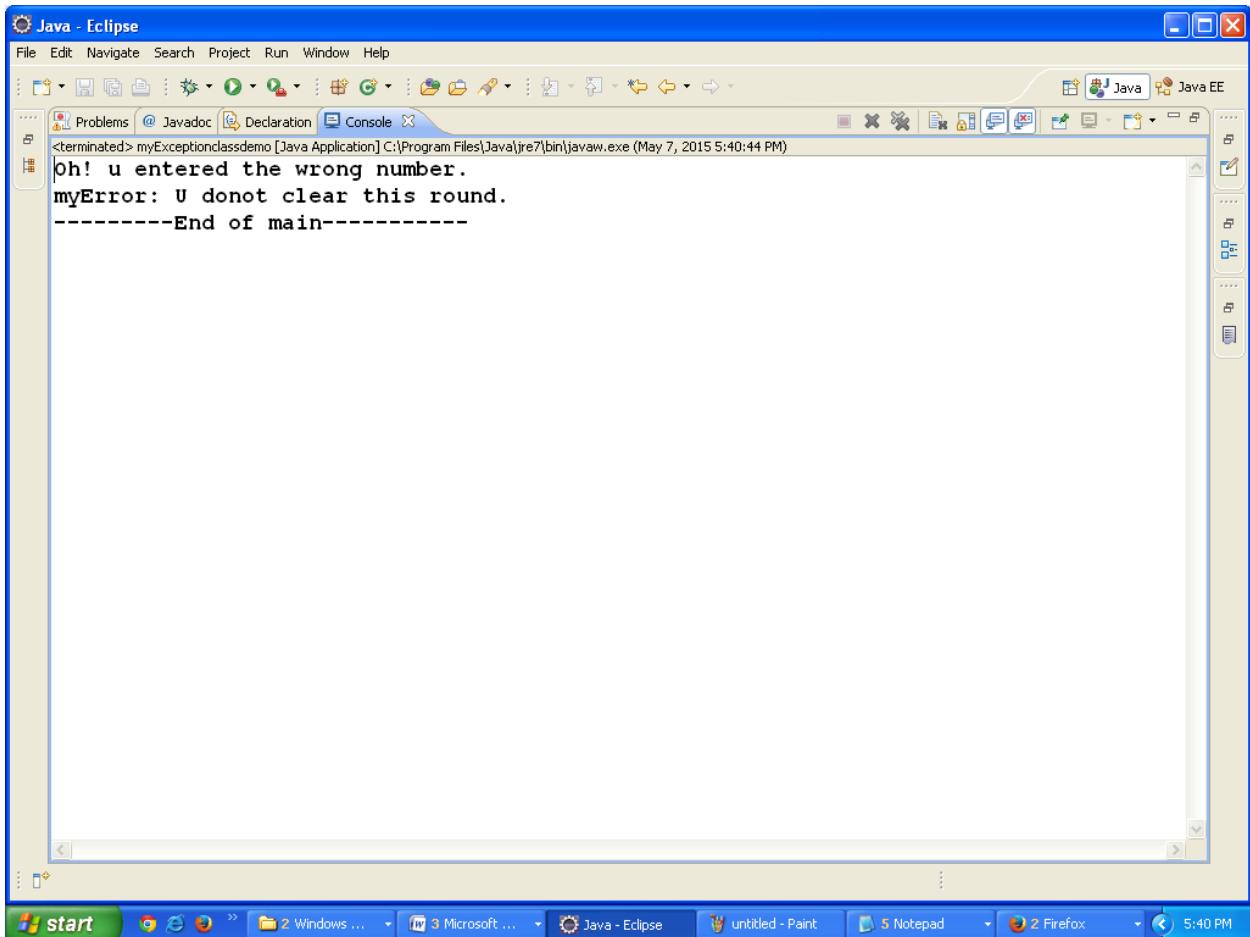
Click on Arguments

## Lab Manual



Enter Arguments and click on Run

## Lab Manual



### Assignments To Solve:

1. Write a class **TestException** to test different types of exceptions, such as NumberFormatException, ArrayIndexOutOfBoundsException, ArithmeticException, NullPointerException, ClassNotFoundException etc
2. Create a class called **CalcAverage** that has the following method:(use throw keyword)  
public double avgFirstN(int N)

This method receives an integer as a parameter and calculates the average of first N natural numbers. If N is not a natural number, throw an exception IllegalArgumentException with an appropriate message.

3. Create a class **Number** having the following features:(use throw keyword)

Attributes

int                first number

int	second number	
result	double	stores result of math operations performed
on a & b		
Member functions(Methods)		

Number(x, y)	constructor to initialize the values of a and b
add()	stores the sum of a and b in result
sub()	stores difference of a and b in result
mul()	stores product in result
div()	stores a divided by b in result

Test to see if b is 0 and throw an appropriate exception since division by zero is undefined.

Display a menu to the user to perform the above four arithmetic operations.

4. Create a user defined Exception class called **InsufficientFundsException**

(Note: Use the existing **Account** Class (created in lab3))

Handle an exception for **Withdraw(int amt)** in Account class

**Withdraw(int amt)** should throw **InsufficientFundsException** if the amount to be withdrawn is greater than the balance.

Accept the values from user for creating Account object and amt to withdraw

5. Create a userdefined Exception classes **IncorrectAgeException** & **IncorrectNationalityException**

Write a class **Voter** with constructors and methods **toString()**, **check(String name, int age)**

The check (...) should check for Nationality and age for voting and throw appropriate Exception

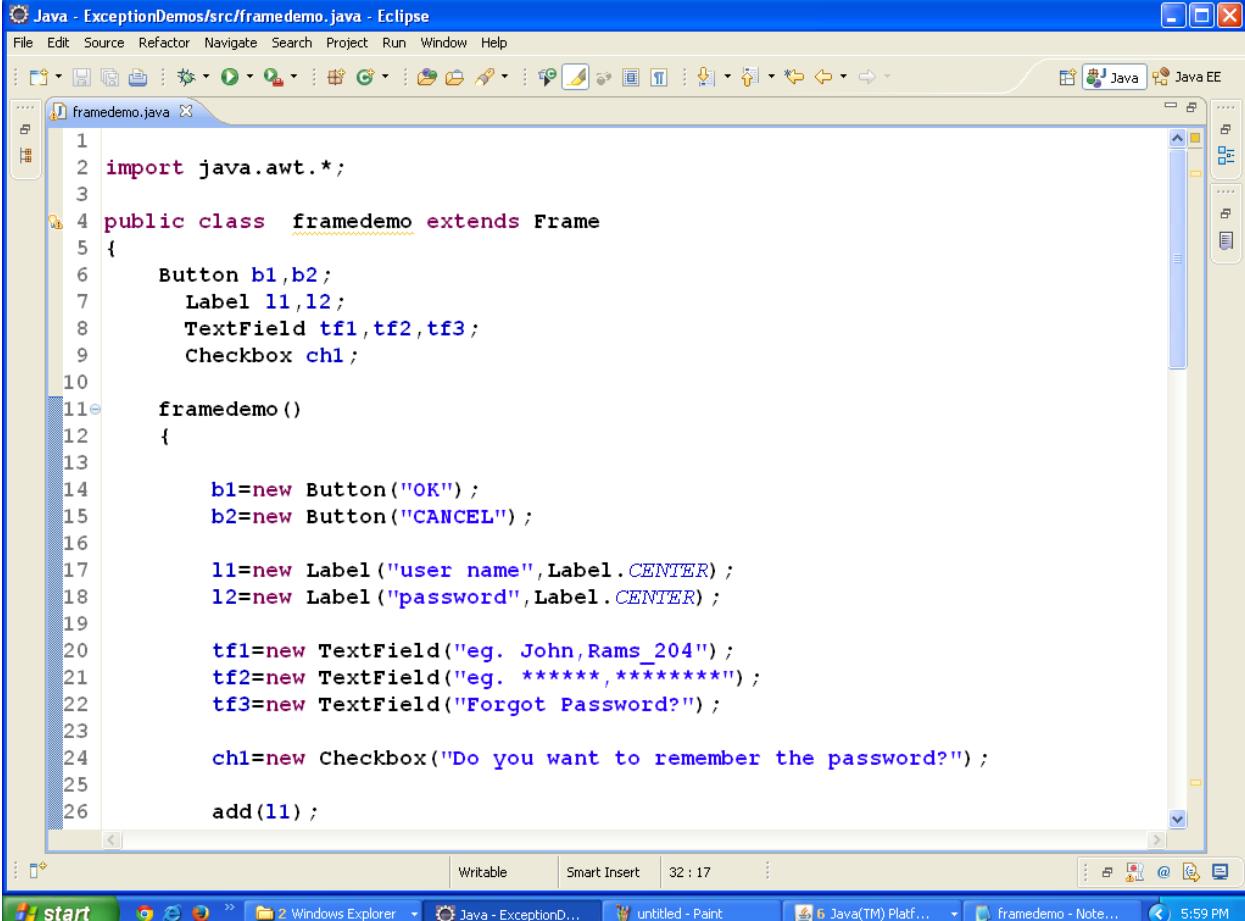
Accept the Name, Nationality & age from user

# Lab-9 Assignments

## AWT(Abstract Window ToolKit)

### 1.Design frames using AWT classes

#### Demo1: To Show the use of various AWT Components



The screenshot shows the Eclipse IDE interface with a Java file named "framedemo.java" open in the editor. The code defines a class "framedemo" that extends the "Frame" class. It contains declarations for several AWT components: two buttons ("b1", "b2"), two labels ("l1", "l2"), three text fields ("tf1", "tf2", "tf3"), and one checkbox ("ch1"). The constructor "framedemo()" initializes these components and adds them to the frame's content pane.

```
Java - ExceptionDemos/src/framedemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
framedemo.java
1 import java.awt.*;
2
3 public class framedemo extends Frame
4 {
5     Button b1,b2;
6     Label l1,l2;
7     TextField tf1,tf2,tf3;
8     Checkbox ch1;
9
10
11    framedemo()
12    {
13
14        b1=new Button("OK");
15        b2=new Button("CANCEL");
16
17        l1=new Label("user name",Label.CENTER);
18        l2=new Label("password",Label.CENTER);
19
20        tf1=new TextField("eg. John_Rams_204");
21        tf2=new TextField("eg. *****,*****");
22        tf3=new TextField("Forgot Password?");
23
24        ch1=new Checkbox("Do you want to remember the password?");
25
26        add(l1);
```

Continued...

## Lab Manual

The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/framedemo.java - Eclipse". The main window displays the Java code for "framedemo.java". The code defines a class with various methods and variable declarations. A status bar at the bottom indicates "Line: 44". The taskbar at the bottom shows multiple open windows, including "Lab Manual F...", "Java - Except...", "untitled - Paint", "Java(TM) ...", "framedemo ~ ...", and "Lab9 - Micros...".

```
27     add(tf1);
28     add(l2);
29     add(tf2);
30     add(tf3);
31     add(b1);
32     add(b2);
33     add(ch1);

34

35     setLayout(null);
36     setBackground(Color.pink);

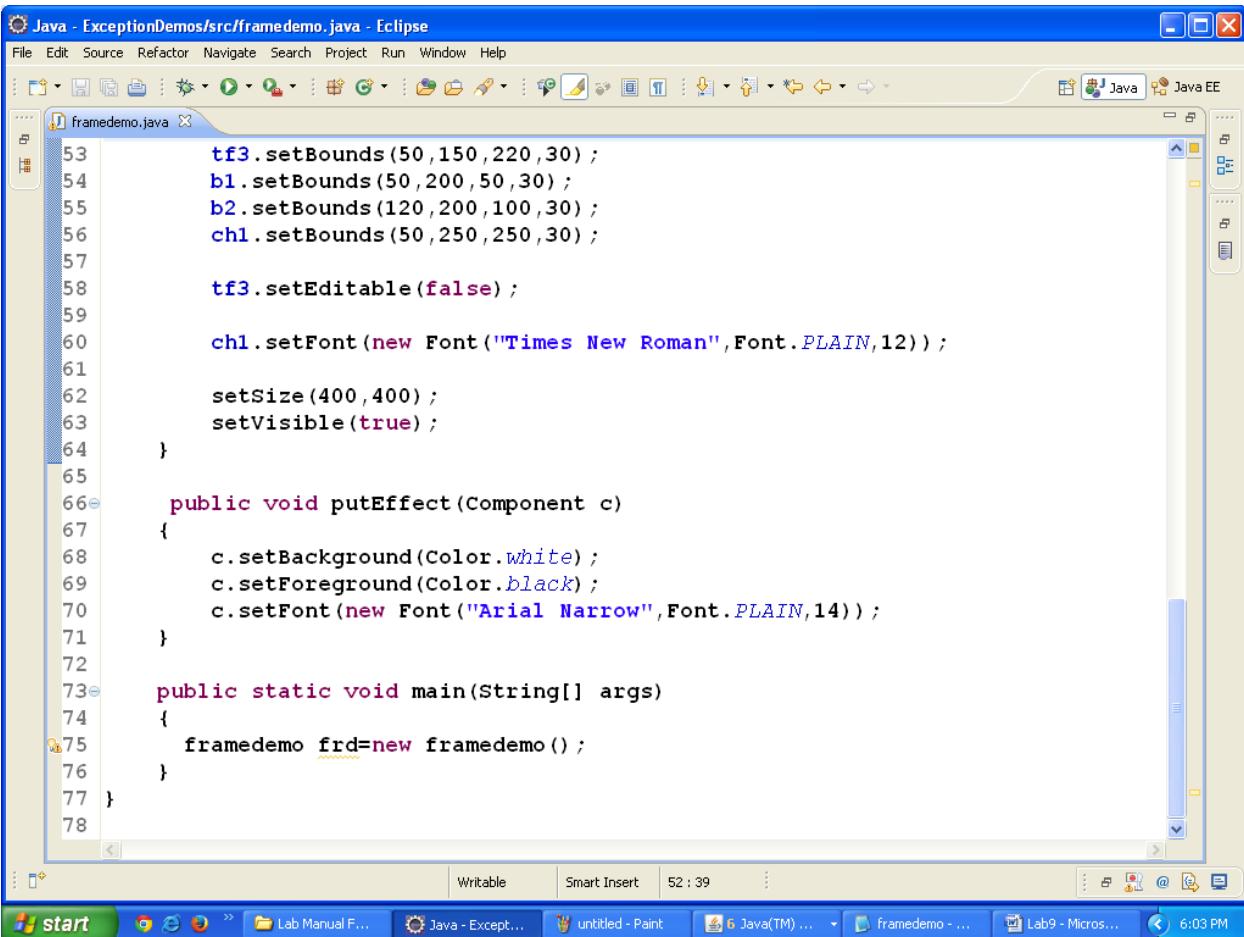
37     putEffect(b1);
38     putEffect(b2);
39     putEffect(l1);
40     putEffect(l2);
41     putEffect(tf1);
42     putEffect(tf2);
43     putEffect(tf3);
44     putEffect(ch1);

45

46     l1.setBounds(50,50,120,30);
47     l2.setBounds(50,100,120,30);
48     tf1.setBounds(200,50,120,30);
49     tf2.setBounds(200,100,120,30);
```

Continued..

## Lab Manual

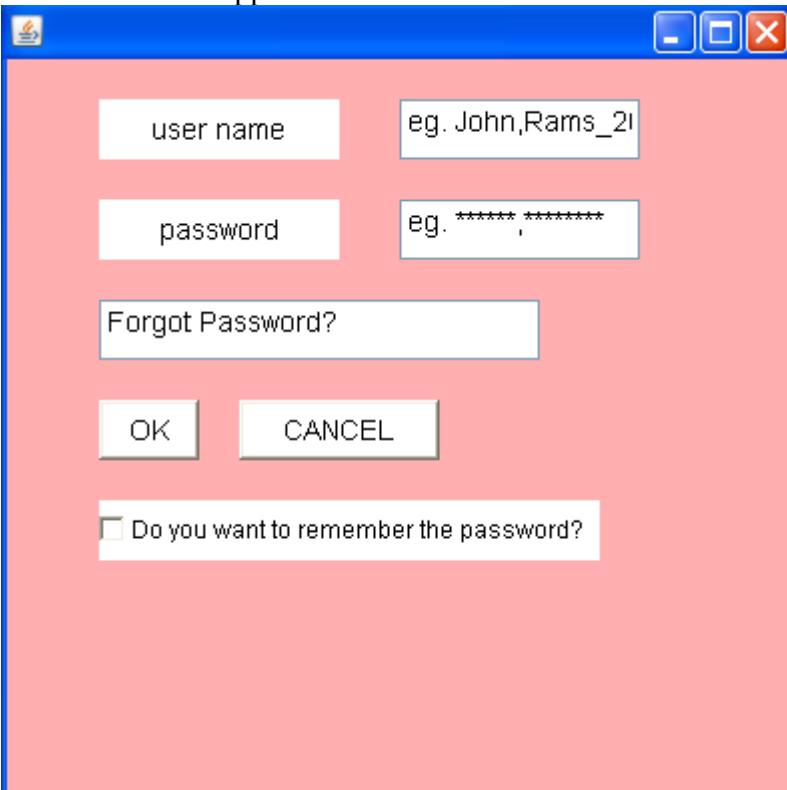


The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/framedemo.java - Eclipse". The main window displays the following Java code:

```
53     tf3.setBounds(50,150,220,30);
54     b1.setBounds(50,200,50,30);
55     b2.setBounds(120,200,100,30);
56     ch1.setBounds(50,250,250,30);
57
58     tf3.setEditable(false);
59
60     ch1.setFont(new Font("Times New Roman",Font.PLAIN,12));
61
62     setSize(400,400);
63     setVisible(true);
64 }
65
66 public void putEffect(Component c)
67 {
68     c.setBackground(Color.white);
69     c.setForeground(Color.black);
70     c.setFont(new Font("Arial Narrow",Font.PLAIN,14));
71 }
72
73 public static void main(String[] args)
74 {
75     framedemo frd=new framedemo();
76 }
77 }
```

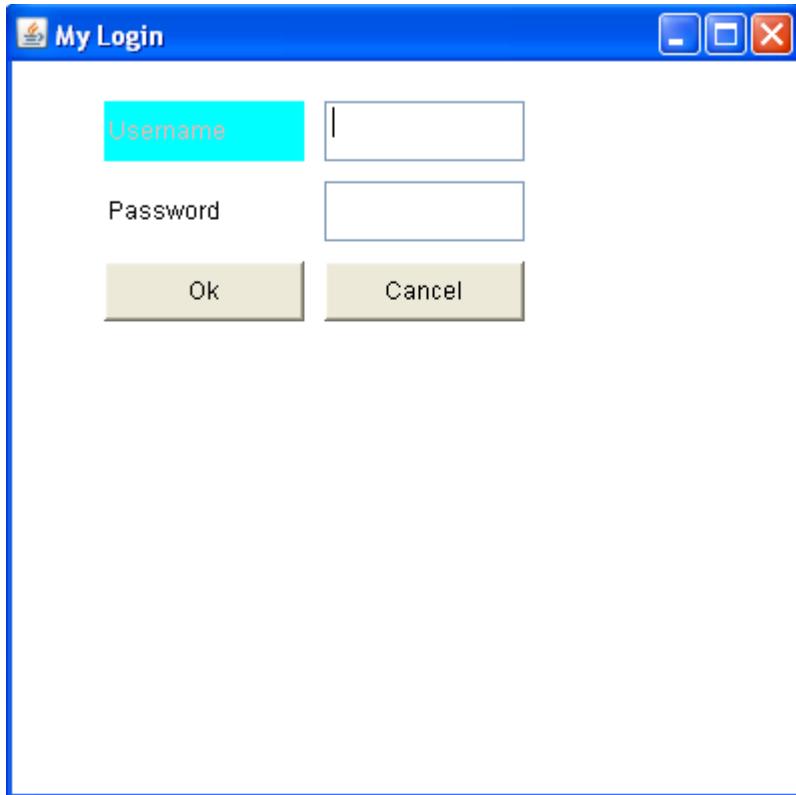
The code defines a Java application named `framedemo`. It sets up a frame with three buttons and one text field, changes the component's font, and makes it visible. The `main` method creates an instance of the `framedemo` class.

Runa as -> Java Application



## **Assignments To Solve**

3. Generate the Login Page as shown below. (Event for which should be handled in the next lab..)



4. Generate the below page.. Draw 3 Scrollbars with backgrounds.(Event for which should be handled in the next lab..)

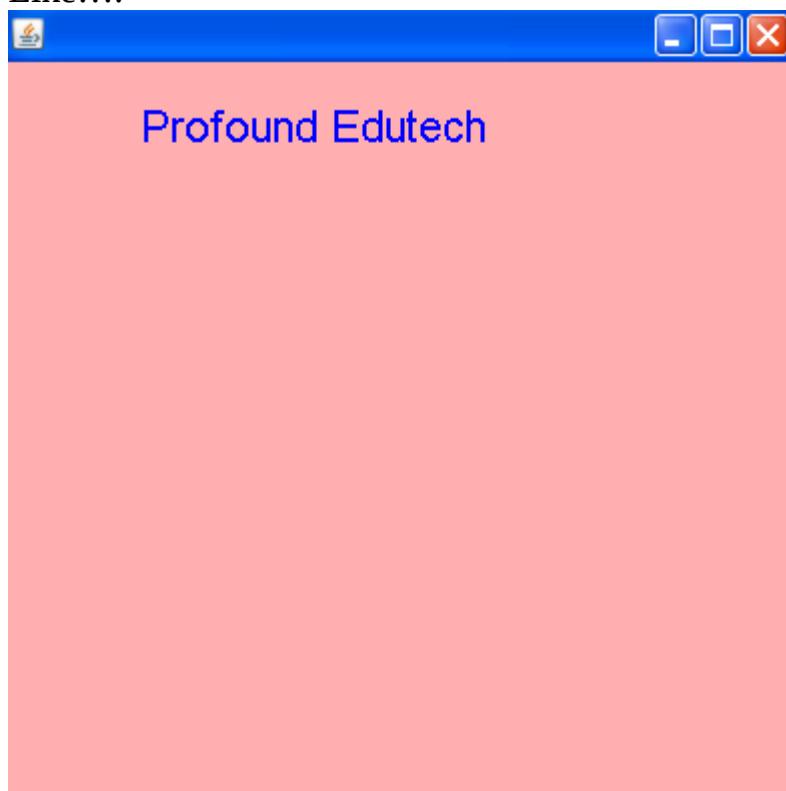
## Lab Manual

---



5. WAP to print your name with the particular Font on the Frame

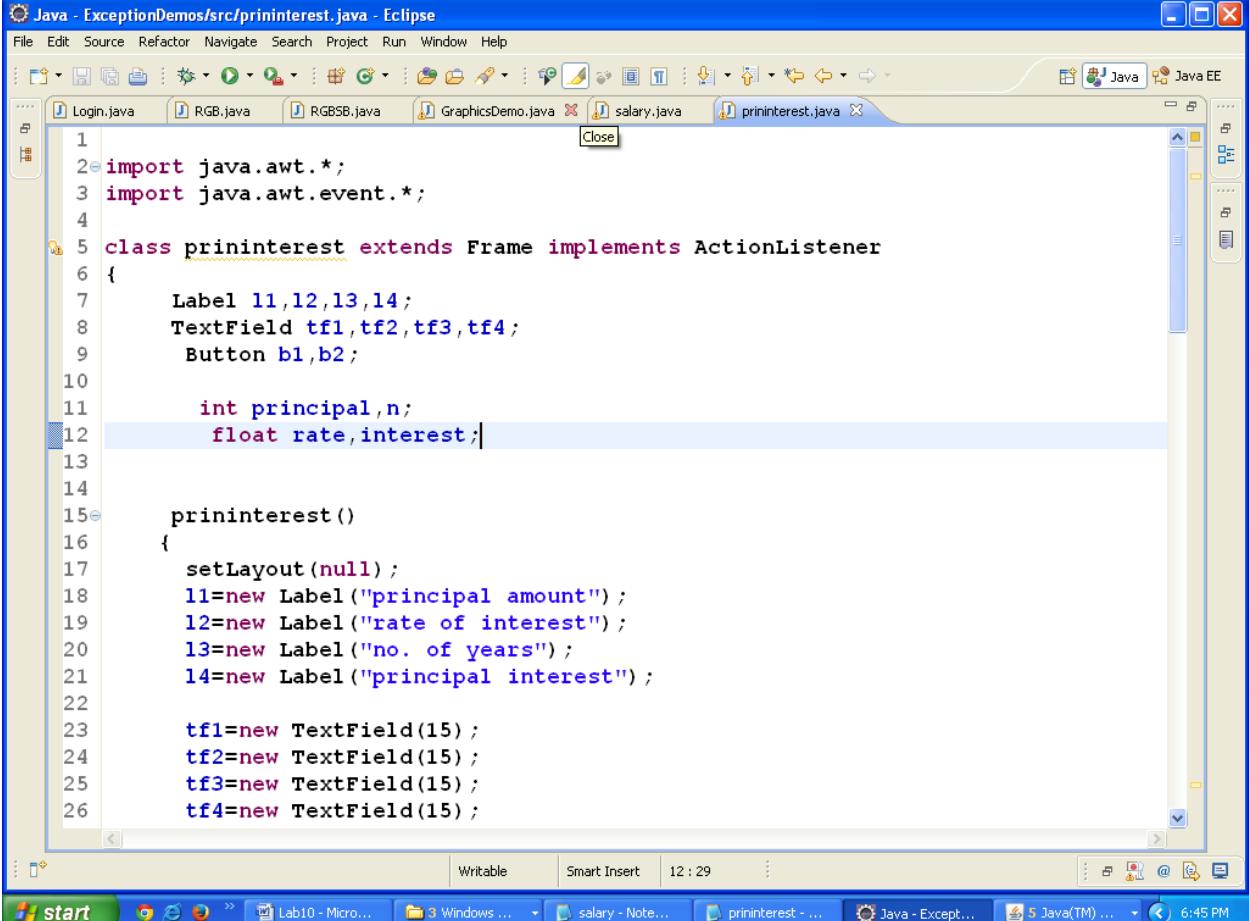
Like....



# Lab-10 Assignments

## Event Handling

Demo 1: Handling an Event for Button to calculate SimpleInterest

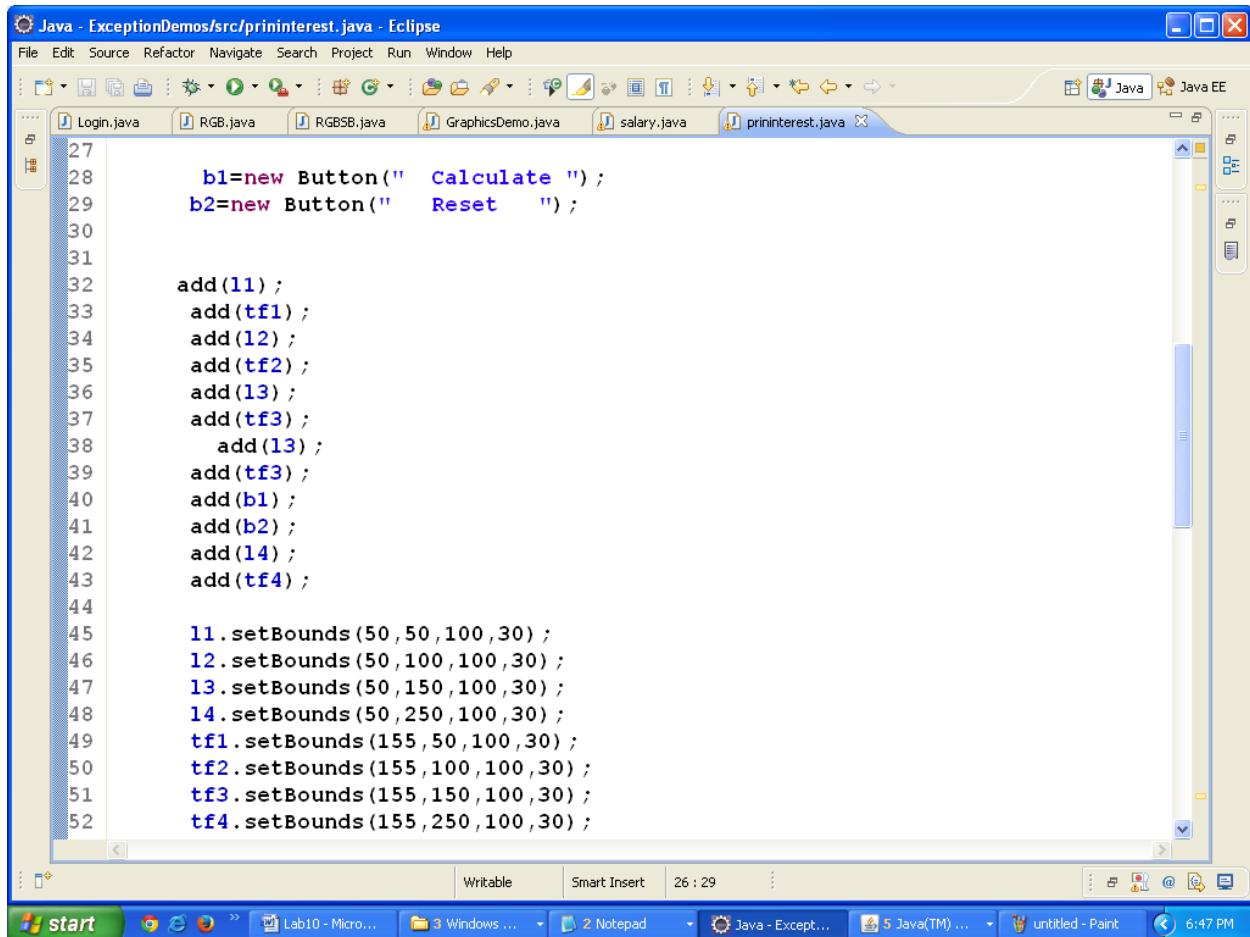


The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/prininterest.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows project files: Login.java, RGB.java, RGBSB.java, GraphicsDemo.java, salary.java, and prininterest.java. The main editor window displays Java code for a frame-based application. The code defines a class `prininterest` that extends `Frame` and implements `ActionListener`. It contains labels for principal amount, rate of interest, no. of years, and principal interest, and four text fields for input. The code is currently at line 12, which defines variables for principal, n, rate, and interest.

```
Java - ExceptionDemos/src/prininterest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Login.java RGB.java RGBSB.java GraphicsDemo.java salary.java prininterest.java
1
2 import java.awt.*;
3 import java.awt.event.*;
4
5 class prininterest extends Frame implements ActionListener
6 {
7     Label l1,l2,l3,l4;
8     TextField tf1,tf2,tf3,tf4;
9     Button b1,b2;
10
11     int principal,n;
12     float rate,interest;
13
14
15     prininterest()
16     {
17         setLayout(null);
18         l1=new Label("principal amount");
19         l2=new Label("rate of interest");
20         l3=new Label("no. of years");
21         l4=new Label("principal interest");
22
23         tf1=new TextField(15);
24         tf2=new TextField(15);
25         tf3=new TextField(15);
26         tf4=new TextField(15);
```

Continued...

## Lab Manual



The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/prininterest.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, and Find. The top status bar shows "Java EE". The main workspace displays Java code for a GUI application. The code defines several components: buttons b1 and b2, text fields tf1 through tf4, and labels l1 through l4. It uses the JPanel class's add() method to add these components to a panel. Each component is assigned specific bounds (x, y, width, height) using the setBounds() method. The code is color-coded, with keywords in blue and variable names in black. The bottom status bar shows "Writable", "Smart Insert", and the current line number "26 : 29". The taskbar at the bottom lists other open windows: "start", "Lab10 - Micro...", "3 Windows ...", "2 Notepad", "Java - Except...", "5 Java(TM) ...", and "untitled - Paint". The system tray shows the date and time as "6:47 PM".

```
27     b1=new Button(" Calculate ");
28     b2=new Button(" Reset   ");
29
30
31     add(l1);
32     add(tf1);
33     add(l2);
34     add(tf2);
35     add(l3);
36     add(tf3);
37     add(l3);
38     add(tf3);
39     add(b1);
40     add(b2);
41     add(l4);
42     add(tf4);
43
44
45     l1.setBounds(50,50,100,30);
46     l2.setBounds(50,100,100,30);
47     l3.setBounds(50,150,100,30);
48     l4.setBounds(50,200,100,30);
49     tf1.setBounds(155,50,100,30);
50     tf2.setBounds(155,100,100,30);
51     tf3.setBounds(155,150,100,30);
52     tf4.setBounds(155,200,100,30);
```

Continued...

## Lab Manual

The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/prininterest.java - Eclipse". The code editor displays Java code for a class named "prininterest". The code includes methods for setting button bounds, adding action listeners, and performing calculations based on user input from text fields. The code is color-coded for syntax highlighting.

```
53     b1.setBounds(50,200,100,30);
54     b2.setBounds(175,200,100,30);
55
56     b1.addActionListener(this);
57     b2.addActionListener(this);
58
59     tf4.setEditable(false);
60     setSize(300,400);
61     setVisible(true);
62 }
63
64 public void actionPerformed(ActionEvent ae)
65 {
66     if(ae.getSource()==b1)
67     {
68         principal=Integer.parseInt(tf1.getText());
69         n=Integer.parseInt(tf3.getText());
70         rate=Float.parseFloat(tf2.getText());
71         interest=(principal*rate*n)/100;
72
73         tf4.setText("Rs."+interest);
74     }
75     else if(ae.getSource()==b2)
76     {
77         tf1.setText("");
78     }
79 }
```

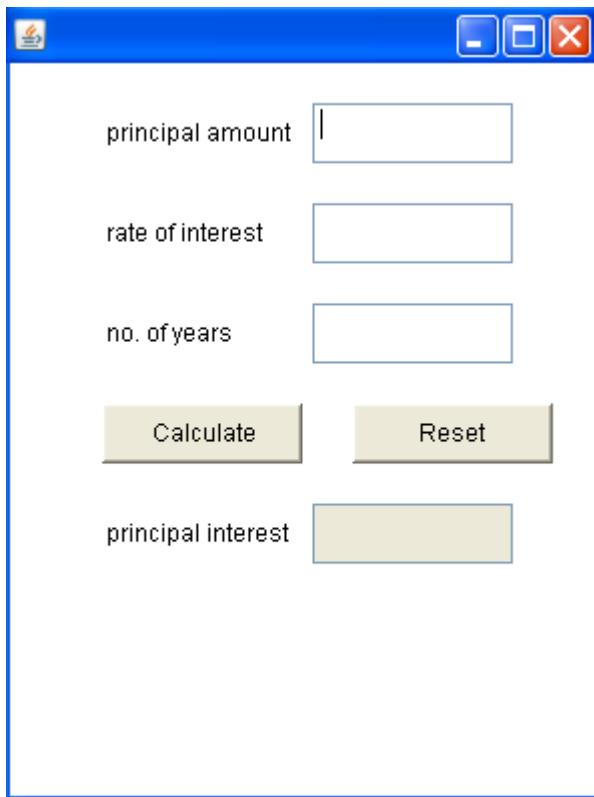
Continued...

The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/prininterest.java - Eclipse". The code editor displays Java code for the "prininterest" class, continuing from the previous screenshot. It includes a main method that creates an instance of the "prininterest" class. The code is color-coded for syntax highlighting.

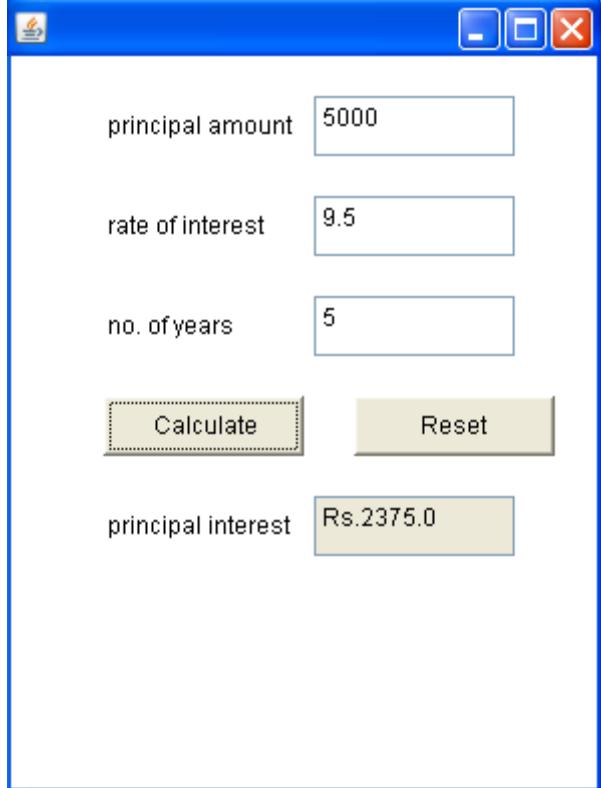
```
68     principal=Integer.parseInt(tf1.getText());
69     n=Integer.parseInt(tf3.getText());
70     rate=Float.parseFloat(tf2.getText());
71     interest=(principal*rate*n)/100;
72
73     tf4.setText("Rs."+interest);
74
75     else if(ae.getSource()==b2)
76     {
77         tf1.setText("");
78         tf2.setText("");
79         tf3.setText("");
80     }
81
82
83 }
84
85
86
87
88 public static void main(String[] args)
89 {
90     prininterest pi=new prininterest();
91 }
92 }
93 }
```

Run as-> Java Application

## Lab Manual



Enter the values and click on Calculate Button(Event Handling)



## **Assignments To Solve**

1. Validate Username and Password for LoginPage created in the previous lab  
Print Access Granted on the Frame for valid data  
Print Access Denied on the Frame for Invalid data  
Handle this event for the click of **OK** Button  
Close the Frame at the click of **Cancel** Button
  
2. Write an event handling code to print your name on the frame at the click of Mouse Button.( hint : Mouse Event)
  
3. Handle an event for the Scrollbars created in the previous lab.  
Set the background color of the frame with the Scroll values of the 3 Scrollbars.
  
4. Handle an event for the TextFields shown below . Show the Focus gained for the TextField when TextField is selected using Tab..(HINT: Use FocusListener)

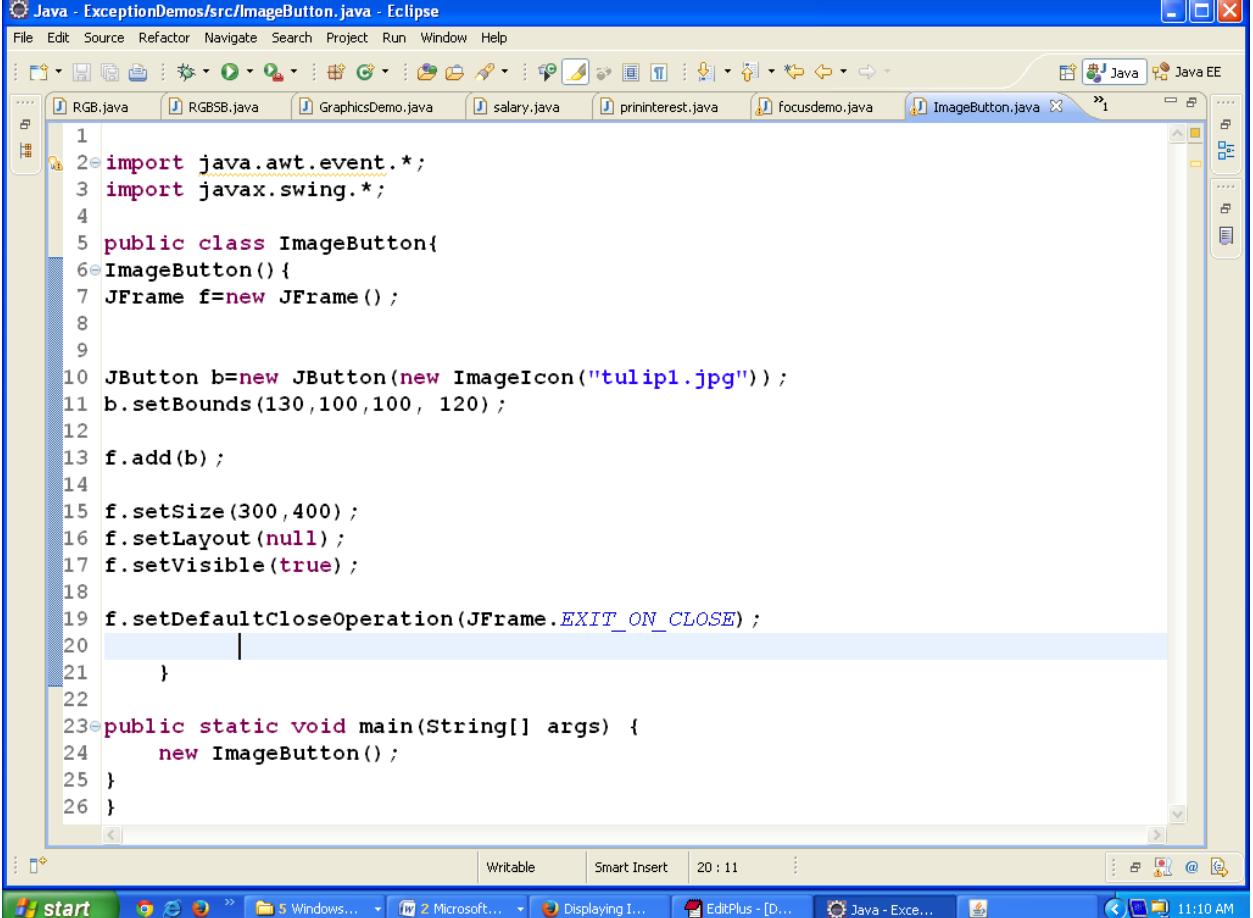


# Lab-11 Assignments

## Swing Components

### MenuBar

#### Demo 1: ImageButton



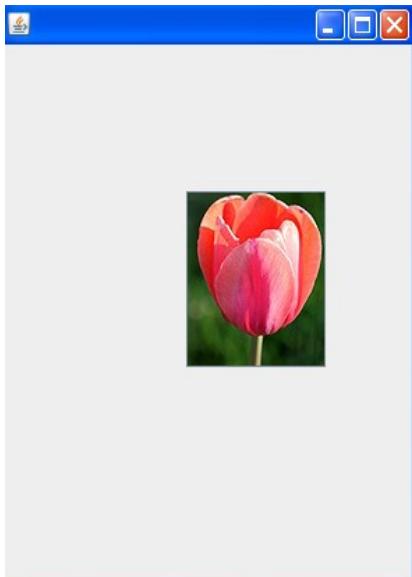
The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/ImageButton.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left margin shows the project structure with files like RGB.java, RGBSB.java, GraphicsDemo.java, salary.java, prininterest.java, Focusdemo.java, and ImageButton.java. The main editor window displays the following Java code:

```
1 import java.awt.event.*;
2 import javax.swing.*;
3 
4 public class ImageButton{
5     ImageButton() {
6         JFrame f=new JFrame();
7 
8         JButton b=new JButton(new ImageIcon("tulip1.jpg"));
9         b.setBounds(130,100,100, 120);
10 
11         f.add(b);
12 
13         f.setSize(300,400);
14         f.setLayout(null);
15         f.setVisible(true);
16 
17         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18     }
19 
20     public static void main(String[] args) {
21         new ImageButton();
22     }
23 }
```

The code creates a JFrame with a JButton containing an image of a tulip. The JButton's bounds are set to (130, 100, 100, 120). The frame is set to null layout and made visible. The default close operation is set to exit on close.

Run as -> Java Application

## Lab Manual



### Demo 2: ScrollBarDemo

The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/ScrollBarExample.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar below the menu bar contains various icons for file operations like Open, Save, and Cut. The central workspace displays the Java code for "ScrollBarExample.java". The code creates a JPanel with a BorderLayout, adds a JLabel, and sets up horizontal and vertical scroll bars with specific unit and block increments. The code is as follows:

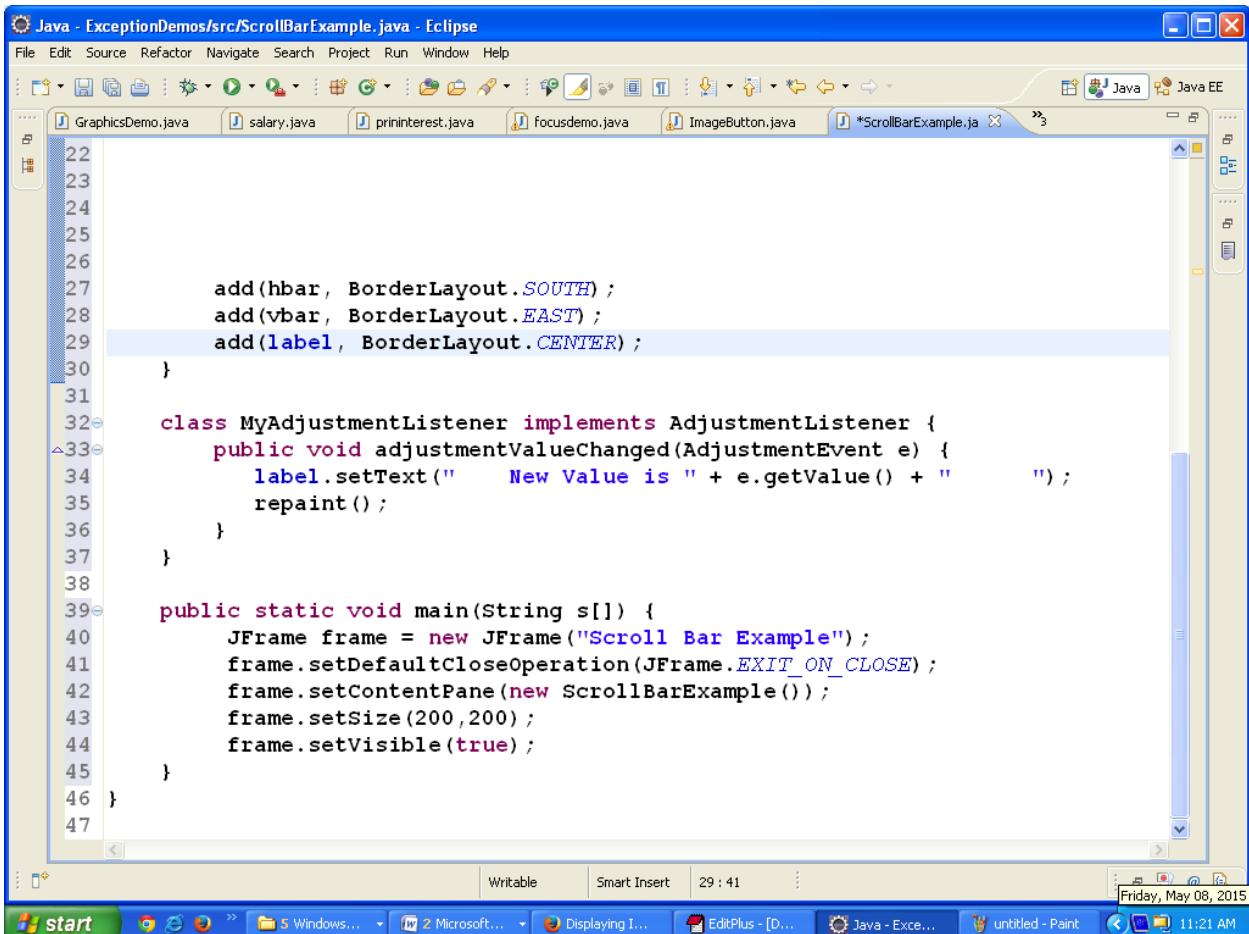
```
1 // ScrollBarExample.java
2 // A quick demonstration of JScrollBar (both vertical and horizontal).
3 //
4 import java.awt.*;
5 import java.awt.event.*;
6 import javax.swing.*;

7 public class ScrollBarExample extends JPanel {
8
9     JLabel label;
10
11     public ScrollBarExample() {
12         super(true);
13         label=new JLabel();
14         setLayout(new BorderLayout());
15
16         JScrollBar hbar=new JScrollBar(JScrollBar.HORIZONTAL, 30, 20, 0, 300);
17         JScrollBar vbar=new JScrollBar(JScrollBar.VERTICAL, 30, 40, 0, 300);
18
19         hbar.setUnitIncrement(2);
20         hbar.setBlockIncrement(1);
21
22         hbar.addAdjustmentListener(new MyAdjustmentListener());
23         vbar.addAdjustmentListener(new MyAdjustmentListener());
24     }
25
26 }
```

The status bar at the bottom shows "Writable", "Smart Insert", "47 : 1", and other system information. The taskbar at the bottom includes icons for Start, Internet Explorer, Microsoft Word, Displaying 1..., EditPlus, Java - Exce..., Paint, and a clock showing 11:14 AM.

Continued...

## Lab Manual



The screenshot shows the Eclipse IDE interface with the title "Java - ExceptionDemos/src/ScrollBarExample.java - Eclipse". The code editor displays Java code for a scroll bar example. The code includes adding components to a frame using BorderLayout (SOUTH, EAST, CENTER), creating a MyAdjustmentListener class that implements AdjustmentListener and handles adjustment value changes, and a main method that creates and sets up a JFrame with a ScrollBarExample content pane. The code is color-coded for syntax.

```
22
23
24
25
26
27     add(hbar, BorderLayout.SOUTH);
28     add(vbar, BorderLayout.EAST);
29     add(label, BorderLayout.CENTER);
30 }
31
32 class MyAdjustmentListener implements AdjustmentListener {
33     public void adjustmentValueChanged(AdjustmentEvent e) {
34         label.setText("    New Value is " + e.getValue() + "    ");
35         repaint();
36     }
37 }
38
39 public static void main(String s[]) {
40     JFrame frame = new JFrame("Scroll Bar Example");
41     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
42     frame.getContentPane(new ScrollBarExample());
43     frame.setSize(200,200);
44     frame.setVisible(true);
45 }
46 }
```

Run as -> Java Application

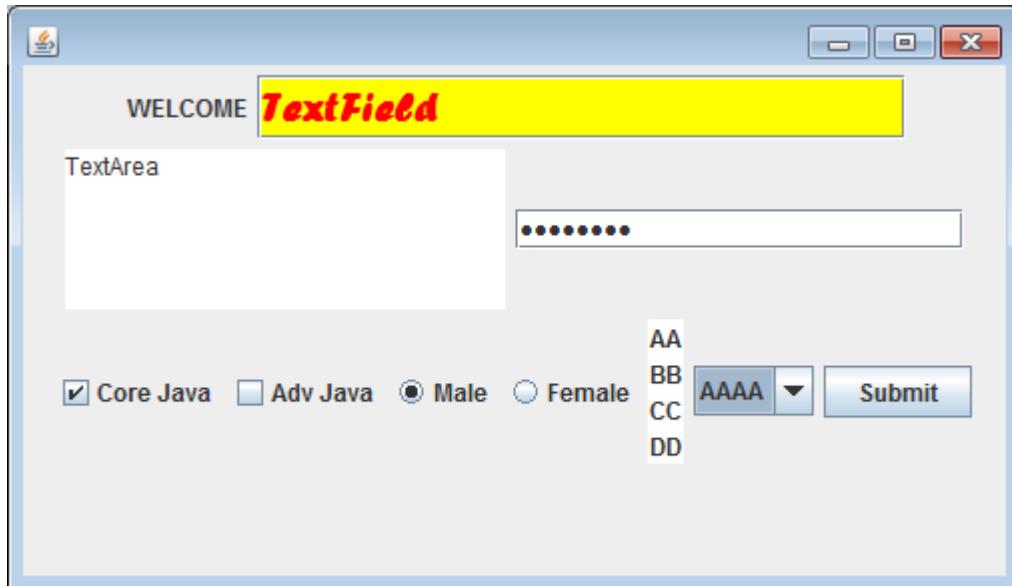


## Assignments To Solve

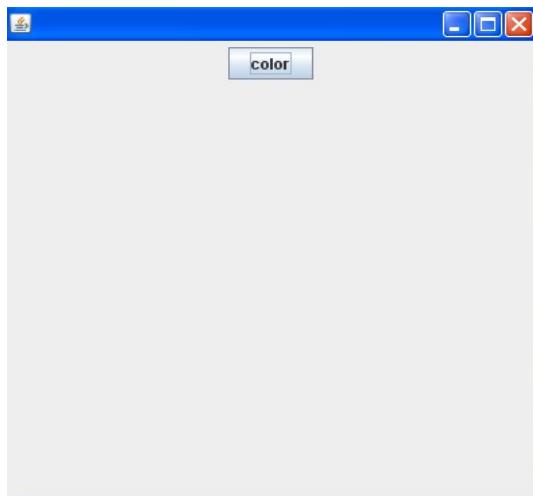
1. Design below form displaying basic swing controls.

## Lab Manual

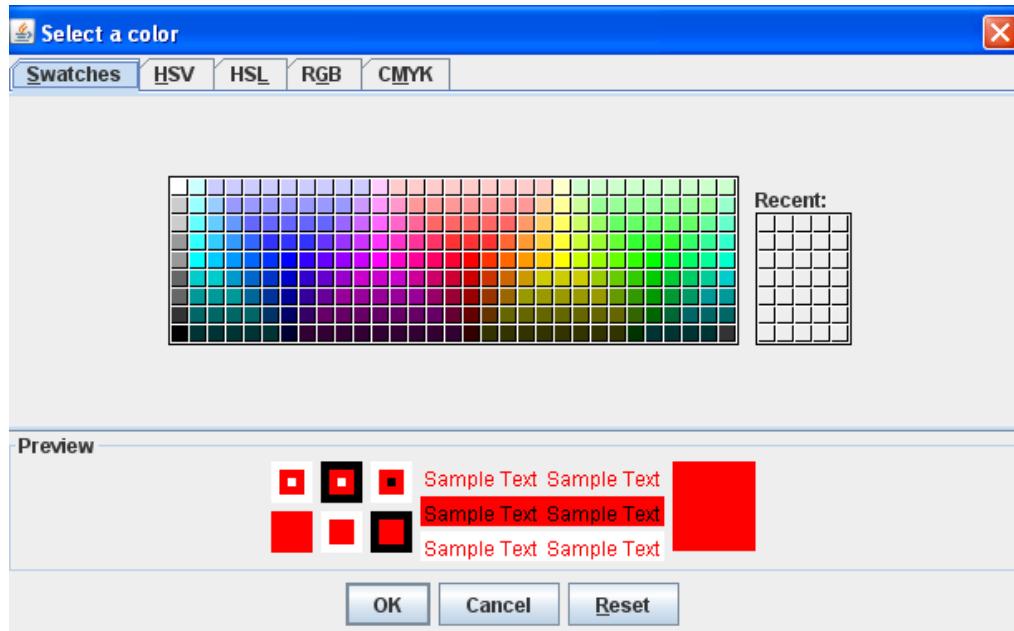
---



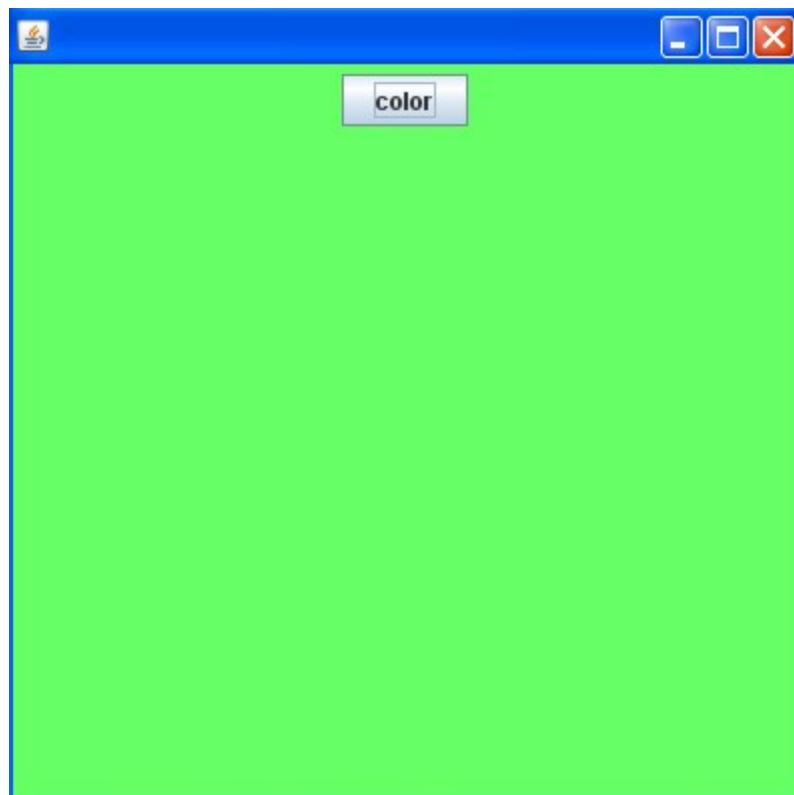
**2. WAP to handle an event for a button to set the background color of the frame by selecting color from JColorChooser class.**



## Lab Manual



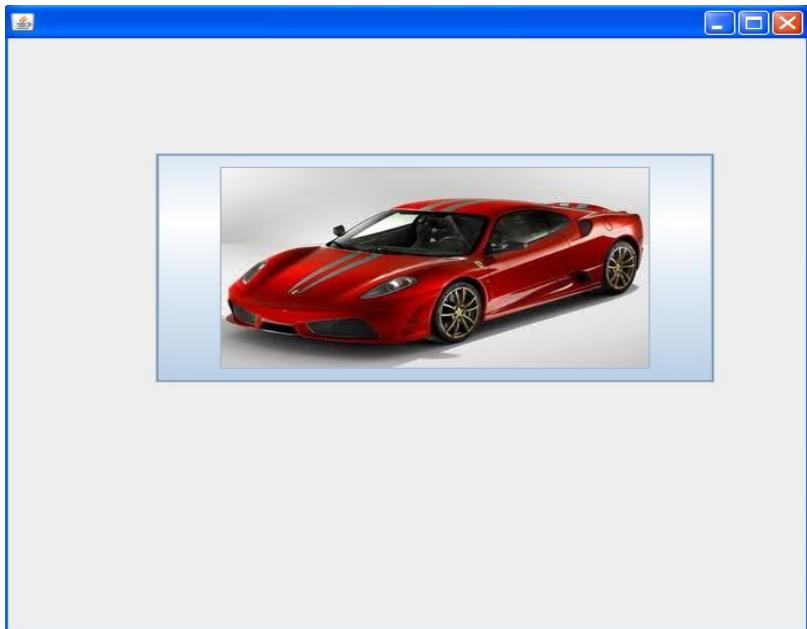
Select and press ok



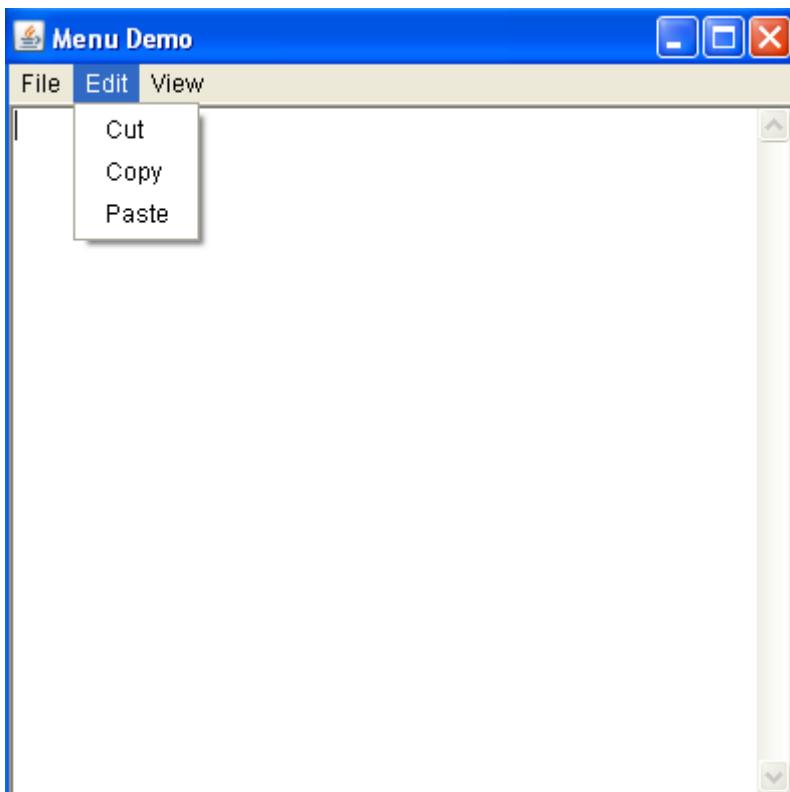
## Lab Manual

---

3. Write an event driven program to move the car up, down, left and right (KeyEvent)



4. Write a java program to design notepad.(Use Jmenubar,Jmenu,JmenuItem,JtextArea)



# **Lab-12 Assignments**

## **IO programming or File handling**

1. Write java program to print file specification such as isFile,isDirectory,last modified date,file size,file patch etc
2. Write a java program to read & write the content to and from “myFile.txt” using FileInputStream and FileOutputStream API
3. Java program to read content from one file and write it into another file.
4. Write a java program to do read/write operation using buffered classes
5. Write a java program that takes a file name and a search string from the user, if the search string occurs in the file, then it counts the no. Of occurrences of the string  
Ex: output :Enter a file name : test.txt  
            Enter a word : you
6. Write a java program to read console data using BufferedReader API
7. Write a Java program to count no. Of words into file
8. Write a java program to read & write the content to and from “myFile.txt” using RandomAccessFile
9. Write a java program to serialize the Student objects and deserialize it(Serializaion)
10. Write a java program to serialize the Employee objects(id,name,salary,address,email,contact) and restrict serialization of email and contact (use ObjectInputStream, ObjectOutputStream,transient)

# Lab-13 Assignments

## Collections & Generics

### Collection APIs

List :ArrayList,LinkedList, Vector

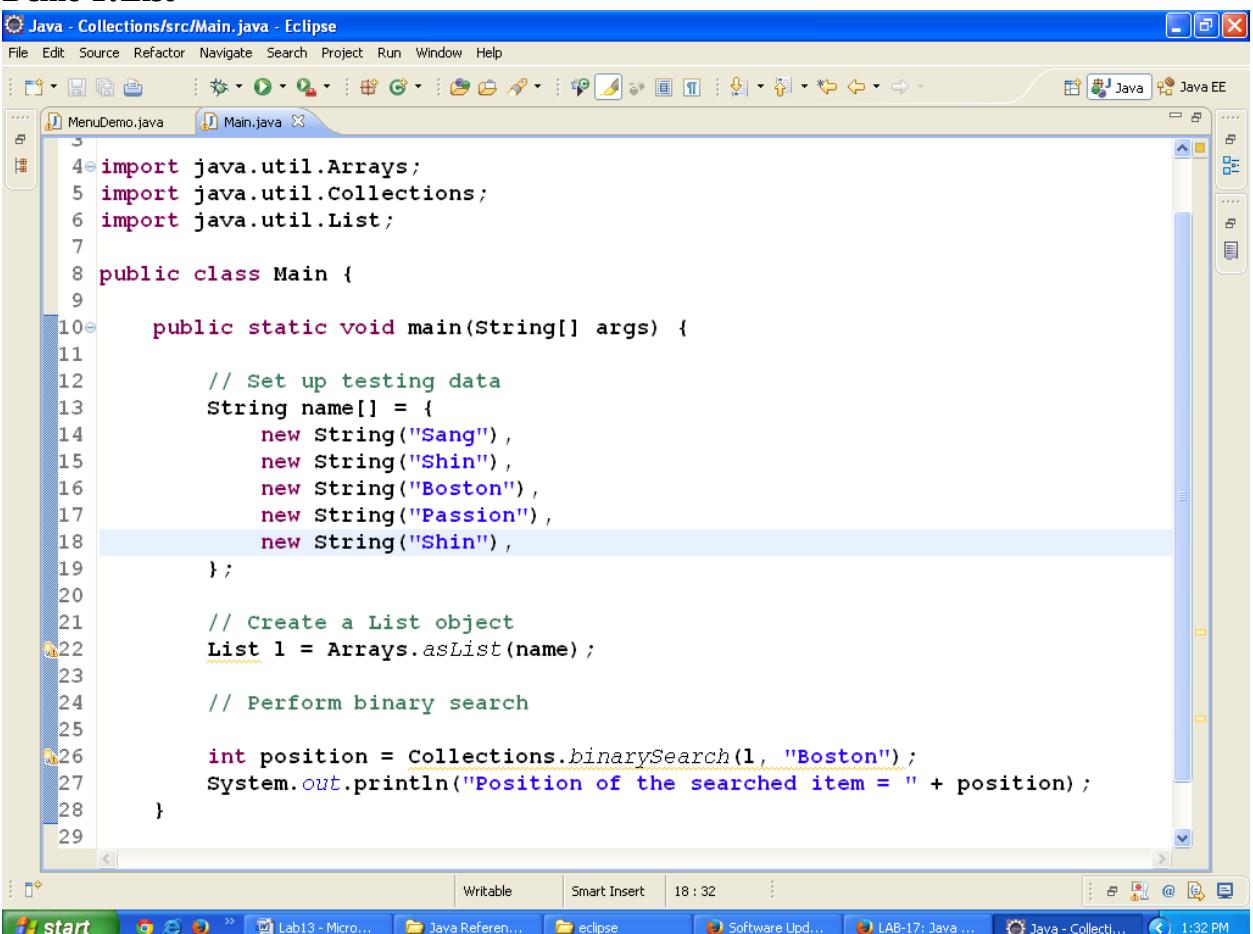
Set :HashSet,LinkedHashSet,TreeSet

Map : HashMap,LinkedHashMap, TreeMap, HashTable

Comparable

Comparator

### Demo 1:List



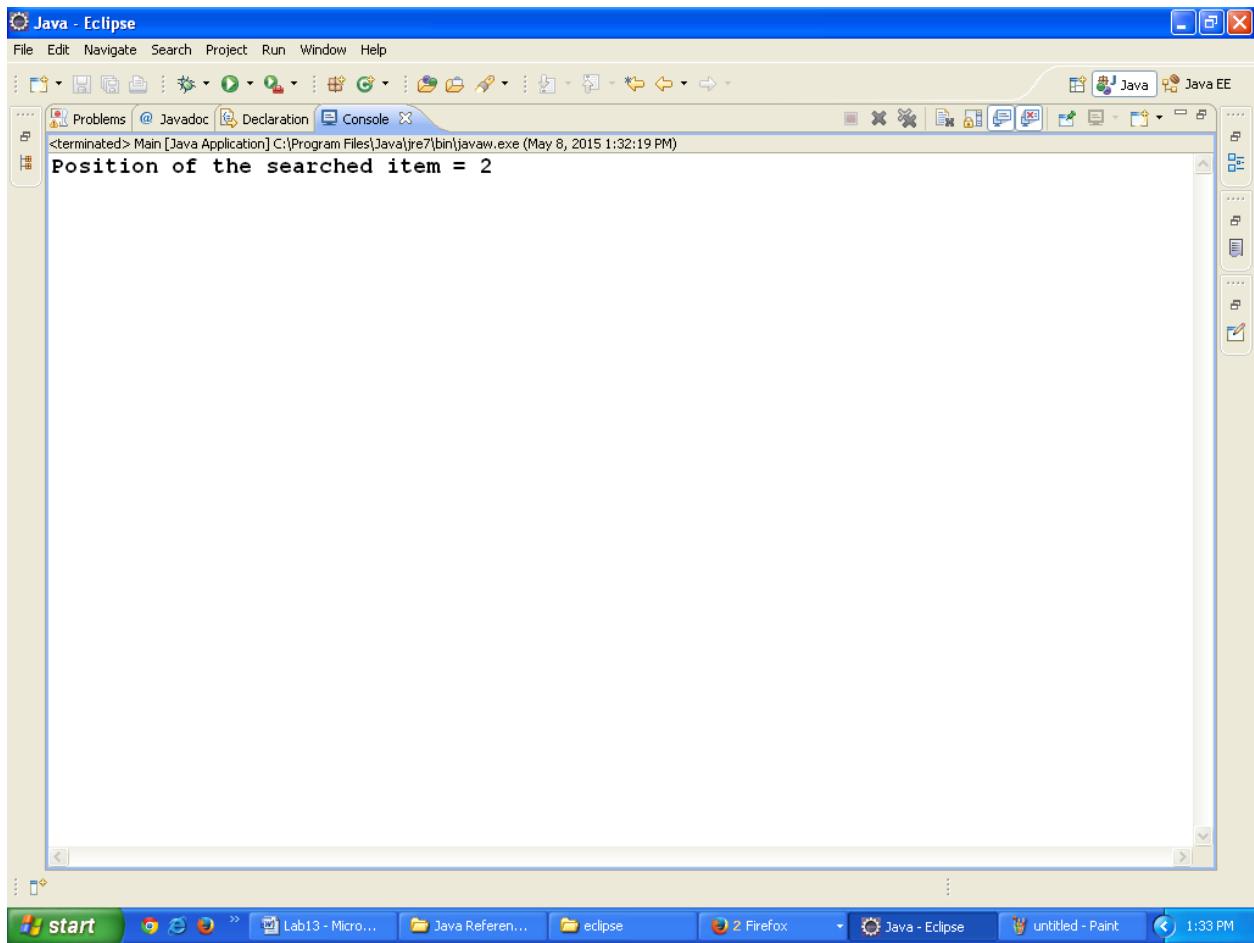
The screenshot shows the Eclipse IDE interface with the title bar "Java - Collections/src/Main.java - Eclipse". The main window displays Java code for a binary search operation on a list of strings. The code imports java.util.Arrays, java.util.Collections, and java.util.List, and defines a Main class with a main method. It creates a String array "name" containing several names and then creates a List "l" using Arrays.asList(name). Finally, it performs a binary search on the list "l" for the string "Boston" and prints the result. The code is highlighted in blue and green, indicating syntax.

```
3  
4 import java.util.Arrays;  
5 import java.util.Collections;  
6 import java.util.List;  
7  
8 public class Main {  
9  
10    public static void main(String[] args) {  
11  
12        // Set up testing data  
13        String name[] = {  
14            new String("Sang") ,  
15            new String("Shin") ,  
16            new String("Boston") ,  
17            new String("Passion") ,  
18            new String("Shin") ,  
19        };  
20  
21        // Create a List object  
22        List l = Arrays.asList(name);  
23  
24        // Perform binary search  
25  
26        int position = Collections.binarySearch(l, "Boston");  
27        System.out.println("Position of the searched item = " + position);  
28    }  
29}
```

Run as -> Java Application

## Lab Manual

---



Demo 2: ArrayListDemo

## Lab Manual

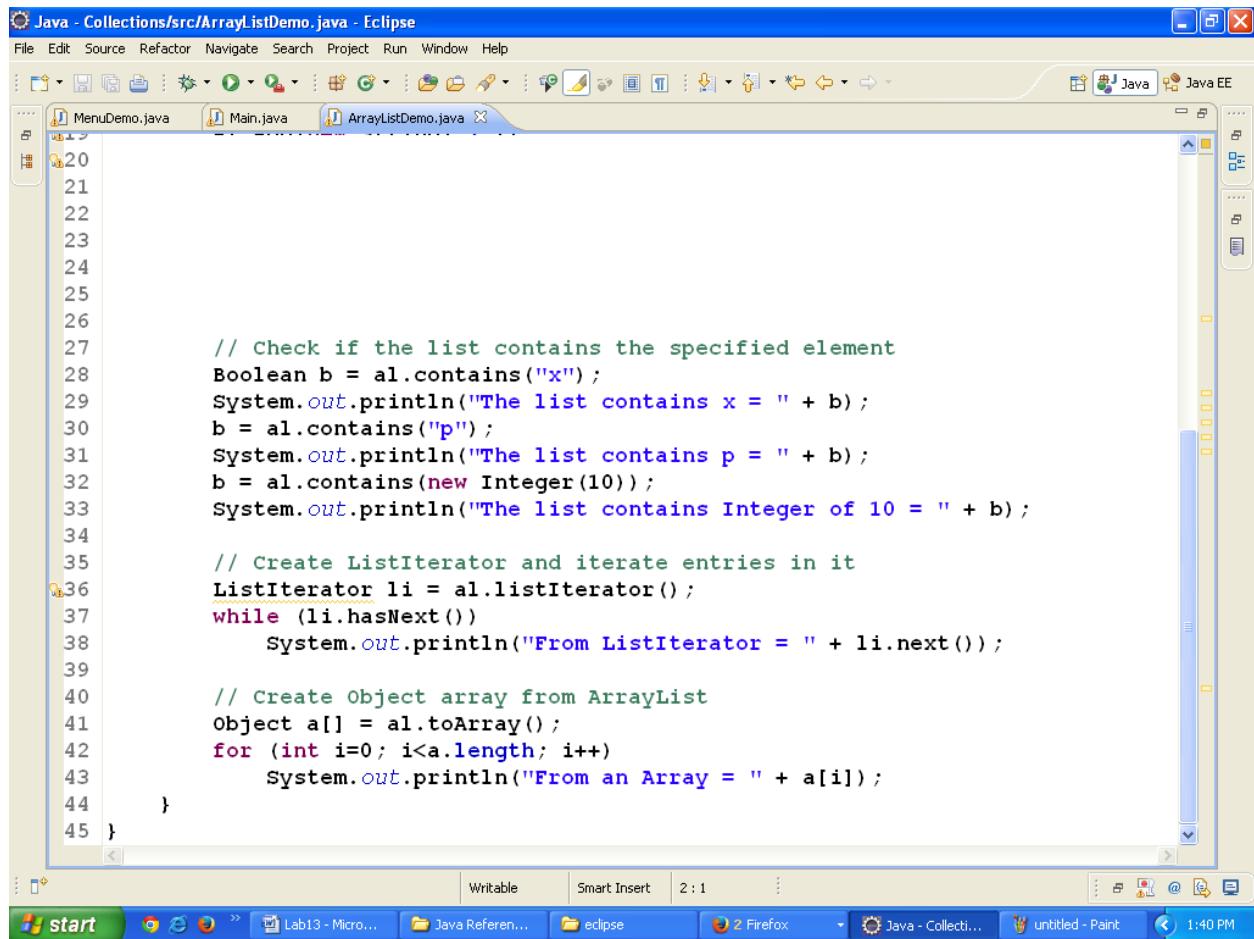
The screenshot shows the Eclipse IDE interface with the title bar "Java - Collections/src/ArrayListDemo.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left margin shows line numbers from 1 to 27. The code editor contains the following Java code:

```
1
2
3
4 import java.util.ArrayList;
5 import java.util.ListIterator;
6
7 public class ArrayListDemo {
8
9     public static void main(String[] args) {
10
11         // Create ArrayList object with capacity of 2 elements
12         ArrayList al = new ArrayList(2);
13         System.out.println(al+", size = "+al.size());
14
15         // Add items to the ArrayList
16         al.add("R");
17         al.add("U");
18         al.add("O");
19         al.add(new String("x"));
20         al.add(2, new Integer(10));
21         System.out.println(al+", size = "+al.size());
22
23         // Remove item
24         al.remove("U");
25         System.out.println(al+", size = "+al.size());
26
27         // Check if the list contains the specified element
}
```

The status bar at the bottom shows "Writable", "Smart Insert", "2:1", and the system tray with icons for start, task switcher, Lab13 - Micro..., Java Referen..., eclipse, Firefox, Java - Collecti..., untitled - Paint, and a clock showing 1:39 PM.

Continued...

## Lab Manual



The screenshot shows the Eclipse IDE interface with the title bar "Java - Collections/src/ArrayListDemo.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left margin shows line numbers 20 to 45. The code editor contains the following Java code:

```
20
21
22
23
24
25
26
27     // Check if the list contains the specified element
28     Boolean b = al.contains("x");
29     System.out.println("The list contains x = " + b);
30     b = al.contains("p");
31     System.out.println("The list contains p = " + b);
32     b = al.contains(new Integer(10));
33     System.out.println("The list contains Integer of 10 = " + b);
34
35     // Create ListIterator and iterate entries in it
36     ListIterator li = al.listIterator();
37     while (li.hasNext())
38         System.out.println("From ListIterator = " + li.next());
39
40     // Create Object array from ArrayList
41     Object a[] = al.toArray();
42     for (int i=0; i<a.length; i++)
43         System.out.println("From an Array = " + a[i]);
44
45 }
```

The status bar at the bottom shows "Writable", "Smart Insert", "2 : 1", and the system tray with icons for Start, Lab13 - Micro..., Java Referenc..., eclipse, Firefox, Java - Collecti..., and untitled - Paint.

Run as -> Java Application

The screenshot shows the Eclipse IDE interface with the title bar "Java - Collections/src/ArrayListDemo.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The main window contains a "Console" tab with the following output:

```
<terminated> ArrayListDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (May 8, 2015 1:42:06 PM)
[], size = 0
[R, U, 10, 0, x], size = 5
[R, 10, 0, x], size = 4
The list contains x = true
The list contains p = false
The list contains Integer of 10 = true
From ListIterator = R
From ListIterator = 10
From ListIterator = 0
From ListIterator = x
From an Array = R
From an Array = 10
From an Array = 0
From an Array = x
```

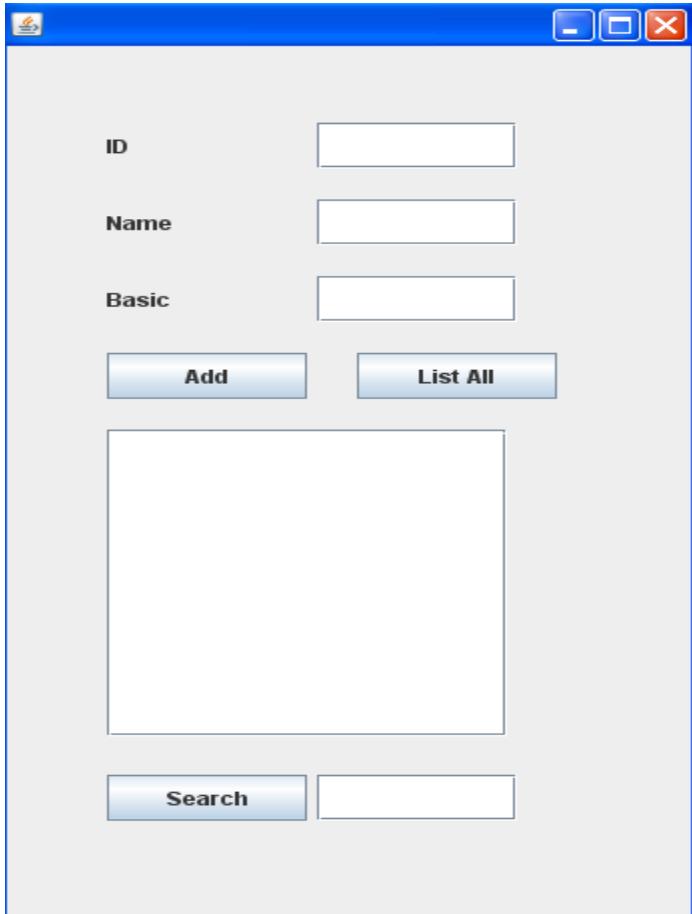
The bottom taskbar shows icons for Start, Micros..., Java Reference, eclipse, Firefox, Java - Collecti..., Paint, and a clock showing 1:42 PM.

## Assignments To Solve :

1. Write a TestBoxing class to check auto boxing and auto unboxing concepts
2. Write a program to test add(),get(),size(),isEmpty(),iterator(),remove() functions with basic collection APIs(List,Set)
2. Write a Java program to create a new array list, add some colors (string) and print out the collection(use iterator() and foreach() for printing elements).
3. Write a Menu Driven program to do following  
To Add elements into ArrayList  
To retrieve an element (at a specified index) from a given array list  
To insert an element into the array list at the first position  
To update specific array element by given element.  
To remove element from given position  
To Search element in an arraylist  
To reverse element of an arraylist  
To sort element of an arraylist
4. WAP to generate the following GUI

## Lab Manual

---



Handle event for Add, List All and Search Buttons

Event for Add:

Retrieve the elements from the textfields, create an Object of Employee and add the Objects to Vector or ArrayList.

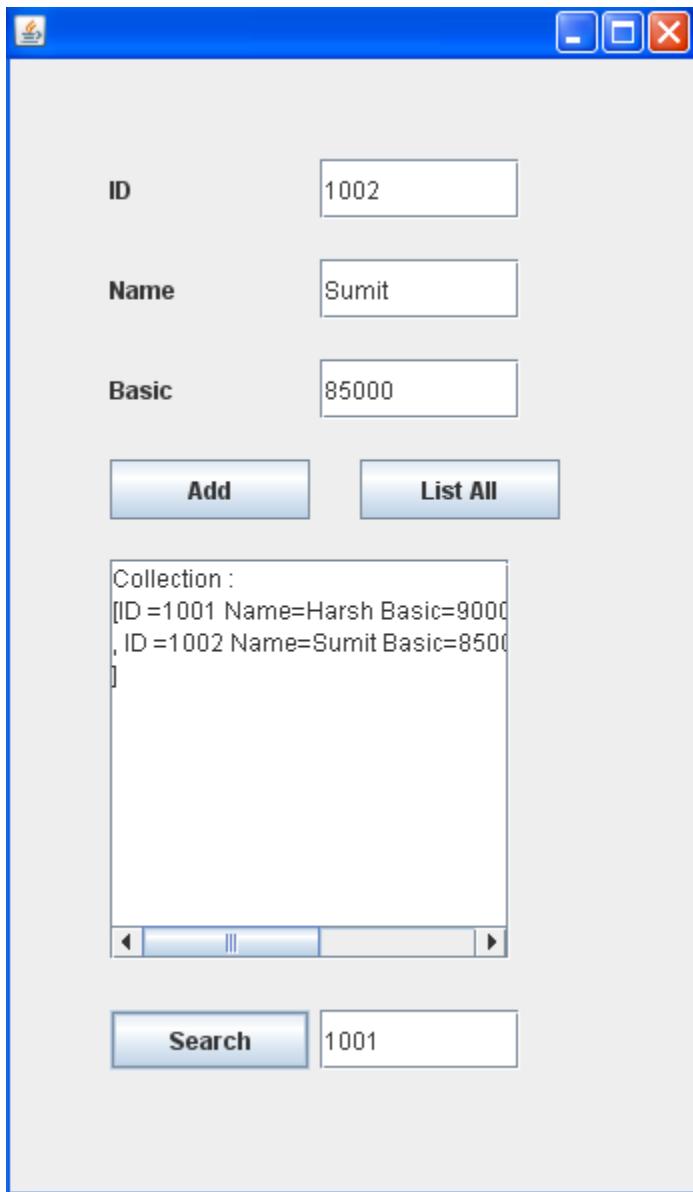
Event for List All

Display All Objects in TextArea

Event for Search

Iterate through the collection to find employee object for a particular id entered and display the details of that Object in TextArea..Or else print message Object not found for given id..

## Lab Manual



5. Create a Product class with Product Id & Product Name. Write a program to accept information of 10 products and store that in HashTable. Search a particular product in the Hash Table. Remove a particular product id and product name from the Hash Table. The product list is as follows:

Product Id	Product Name
P001	Maruti800
P002	MarutiZen
P003	MarutiEsteem

6. Write a Java program to Store the <telephone\_no,Name\_of\_Customer> and sort the data by key of descending order

---

## **Lab Manual**

---

7. Write a Java program which take every entry as name and age of student, and save it as below specified way

Example :

Input :

Enter name and age of 1th :

Pravin 25

Enter name and age of 1th :

Prmod 20

Enter name and age of 1th :

Soham 25

Enter name and age of 1th :

Pravin 5

O/P:

Pravin 25,5

Prmod 20

Soham 25

8. Write a java program to save 5 customer object into TreeSet and print it using descending order. (use comparable or comparator)

# Lab-14 Assignments

## MultiThreading

Thread class and their methods

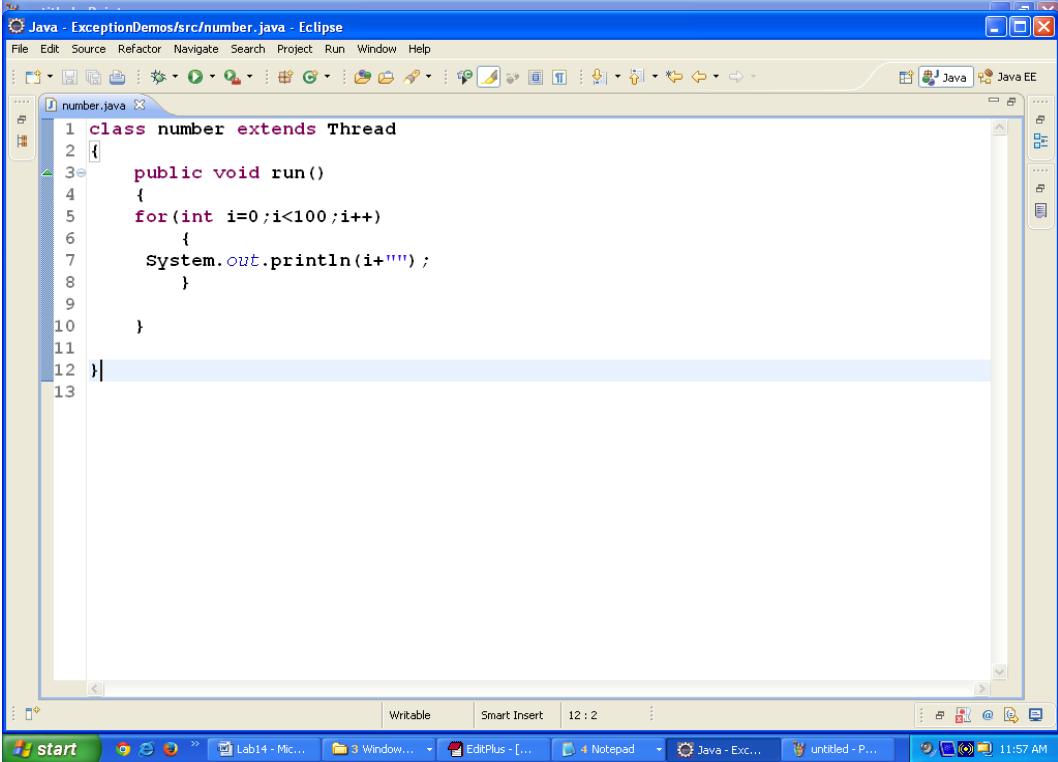
Runnable Interface

Synchronization

Interthread communication

Lock api

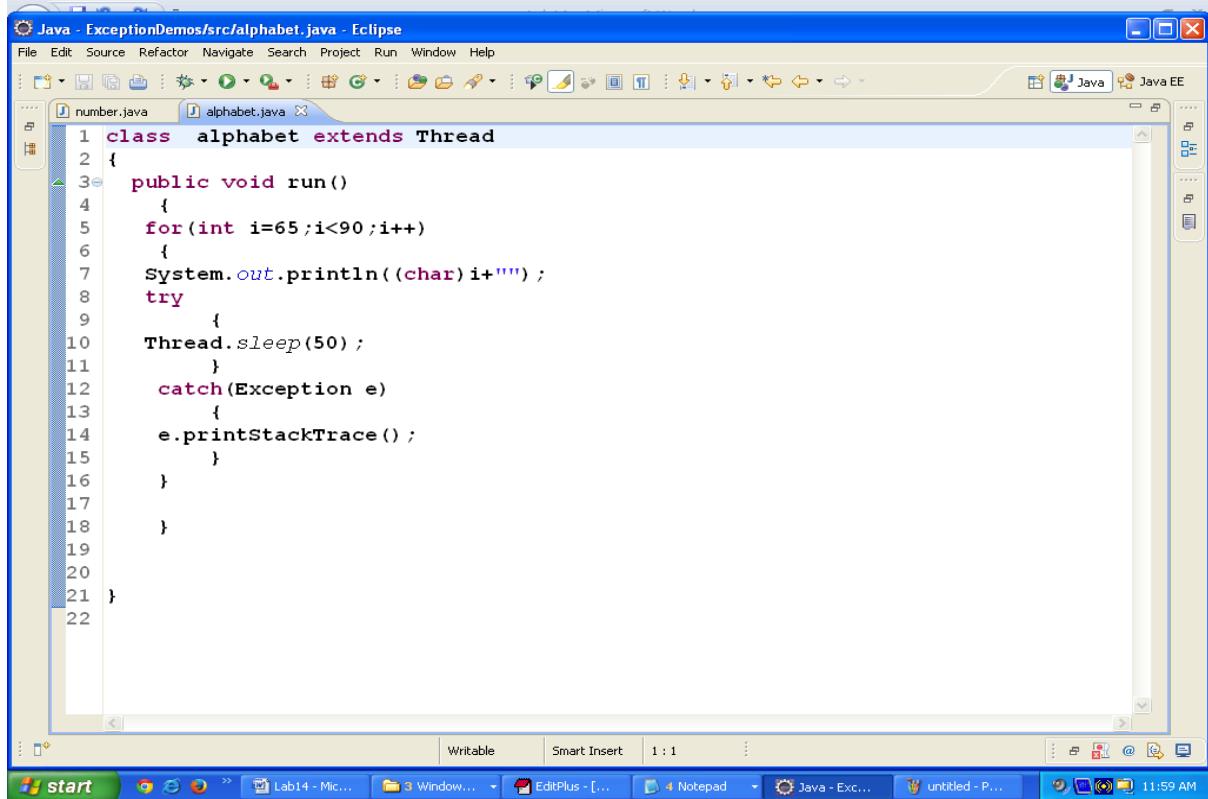
### Demo 1: ThreadDemo



The screenshot shows the Eclipse IDE interface with a Java file named "number.java" open in the editor. The code defines a class "number" that extends the "Thread" class. The "run" method contains a loop that prints integers from 0 to 99 to the console. The Eclipse interface includes toolbars, a menu bar, and various windows for project management and code navigation.

```
Java - ExceptionDemos/src/number.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
number.java
1 class number extends Thread
2 {
3     public void run()
4     {
5         for(int i=0;i<100;i++)
6         {
7             System.out.println(i+"");
8         }
9     }
10 }
11
12 }
13
```

## Lab Manual

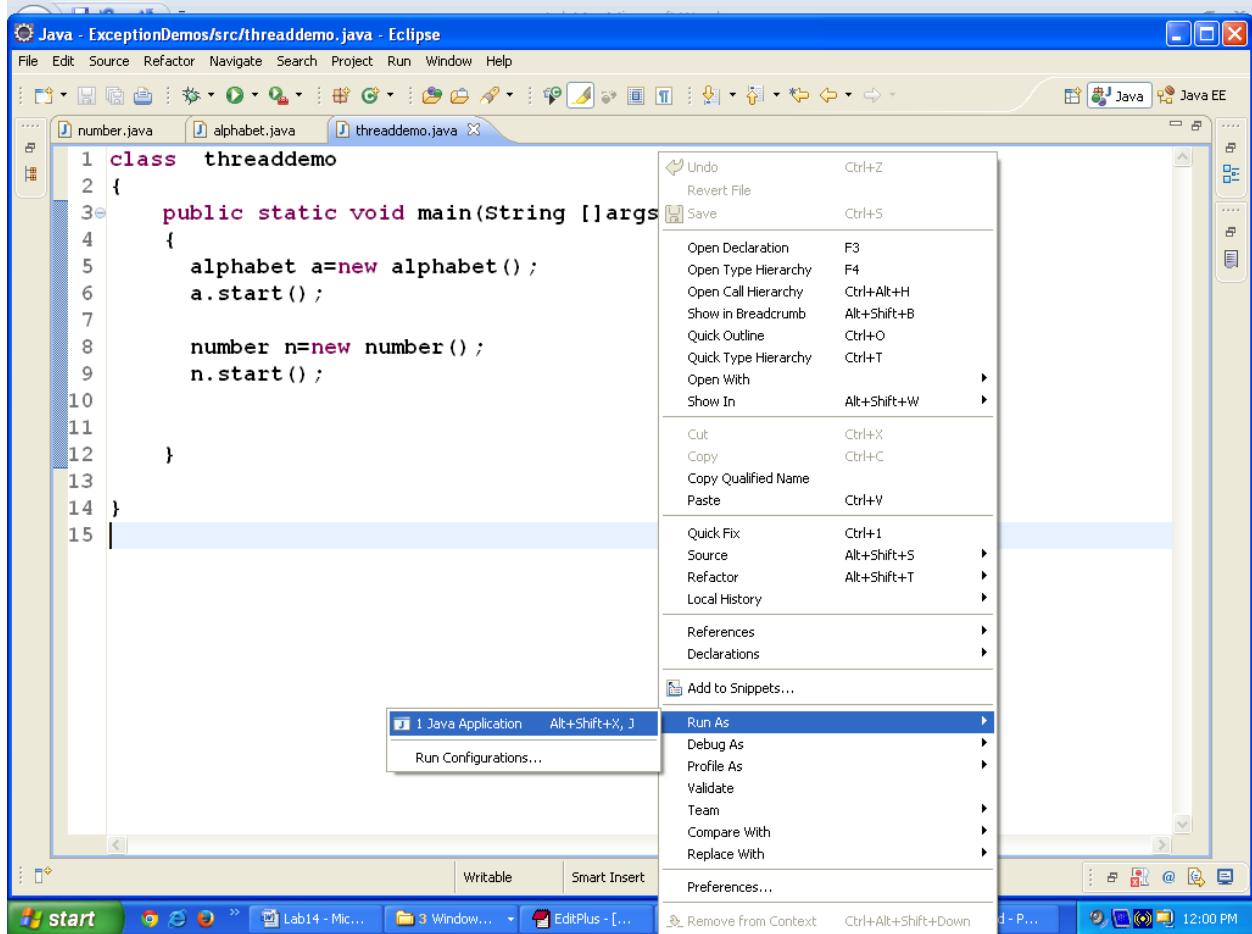


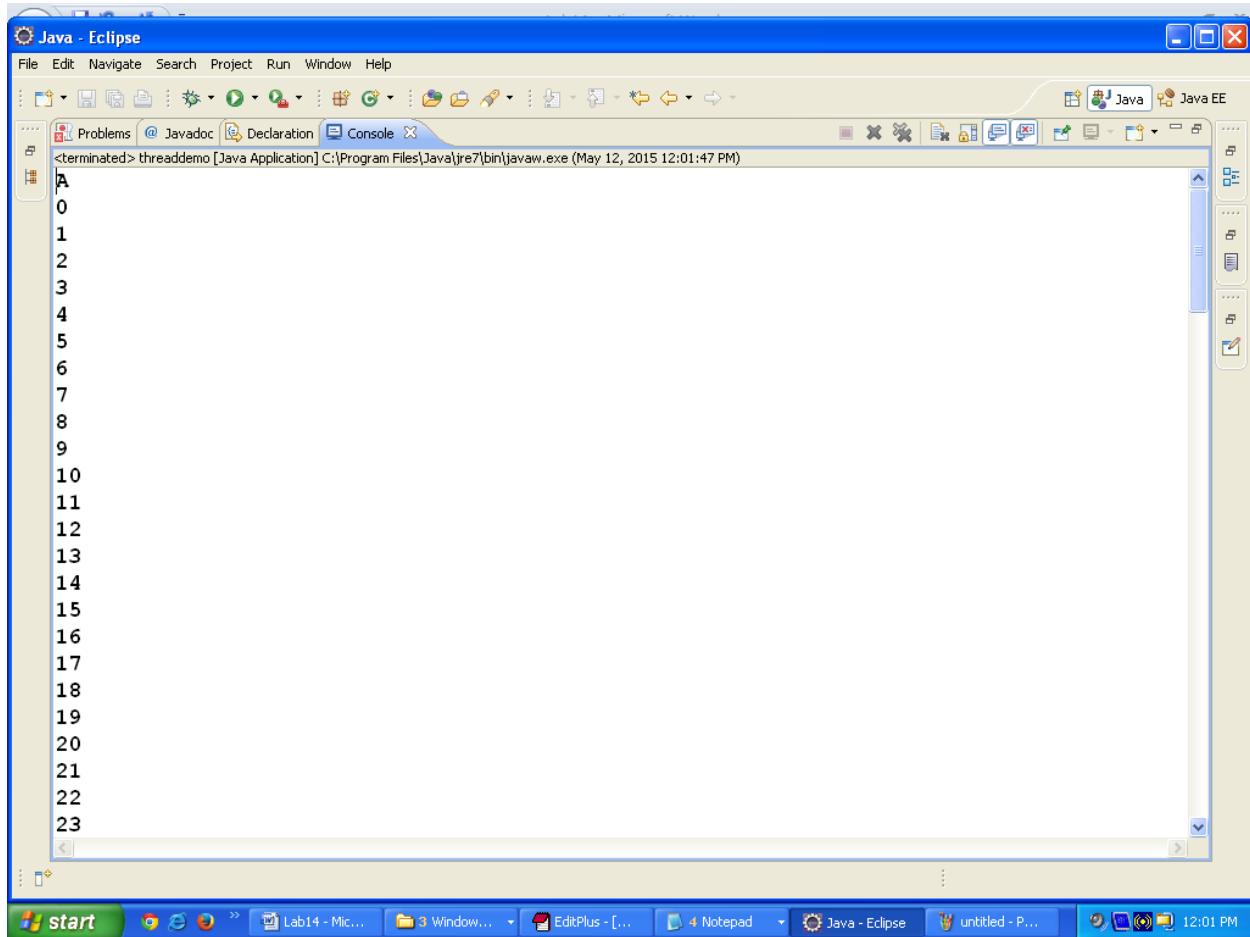
The screenshot shows the Eclipse IDE interface with the title bar "Java - ExceptionDemos/src/alphabet.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, and Find. The left sidebar shows the project structure with "number.java" and "alphabet.java" selected. The main editor window contains the following Java code:

```
1 class alphabet extends Thread
2 {
3     public void run()
4     {
5         for(int i=65;i<90;i++)
6         {
7             System.out.println((char)i+"");
8             try
9             {
10                 Thread.sleep(50);
11             }
12             catch(Exception e)
13             {
14                 e.printStackTrace();
15             }
16         }
17     }
18 }
19
20
21 }
22 }
```

The status bar at the bottom shows "Writable", "Smart Insert", "1 : 1", and a timestamp of "11:59 AM". The taskbar at the bottom has icons for Start, Internet Explorer, Mozilla Firefox, Lab14 - Microsoft Word, 3 Windows, EditPlus, Notepad, Java - Eclipse, and Java - untitled - P... The system tray shows icons for network, battery, and volume.

## Lab Manual





## Assignments To Solve

1. Write a Java program to create Thread and practice below methods.

getName(): It is used for Obtaining a thread's name

getPriority(): Obtain a thread's priority

isAlive(): Determine if a thread is still running

join(): Wait for a thread to terminate

run(): Entry point for the thread

sleep(): suspend a thread for a period of time

start(): start a thread by calling its run() method

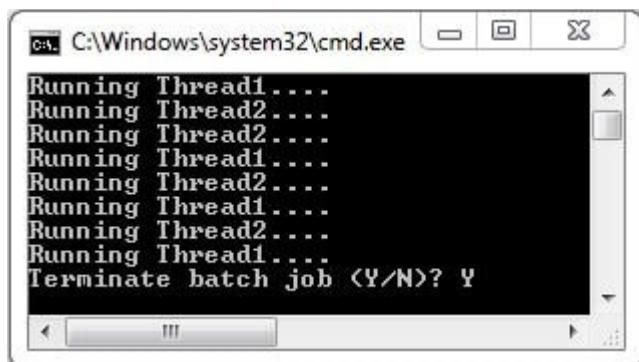
2. Write a Java program to create multiple threads using Runnable interface and print currently running thread.

## Lab Manual

---

3. Write a program that creates 2 threads (Thread1,Thread2)- each displaying a message (Pass the message as a parameter to the constructor). The threads should display the messages continuously till the user presses ctrl+c. (use Thread.sleep(400))

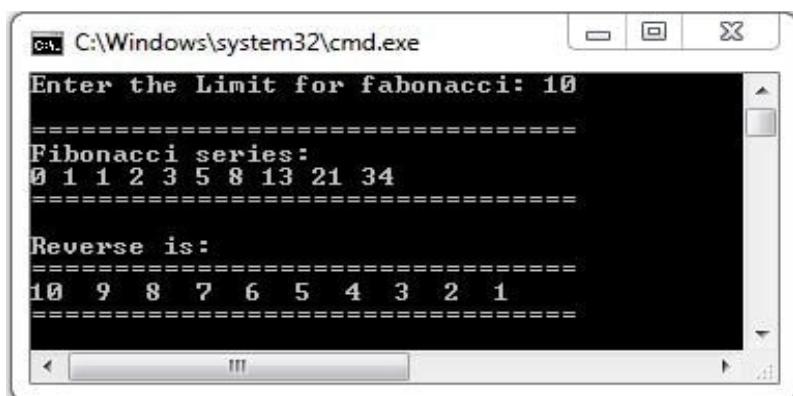
Output :



4. Write a JAVA program which will generate the threads:

- To display 10 terms of Fibonacci series. (class Fibonacci extends Thread)
- To display 1 to 10 in reverse order. (class Reverse extends Thread)

Output :



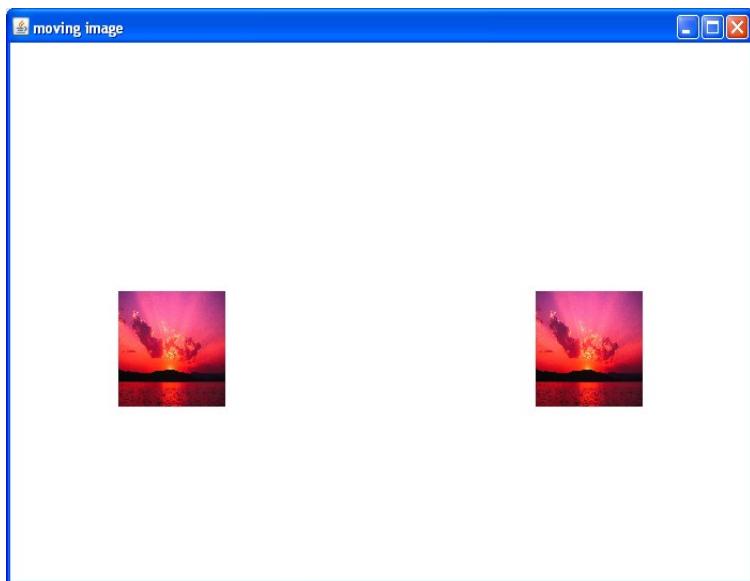
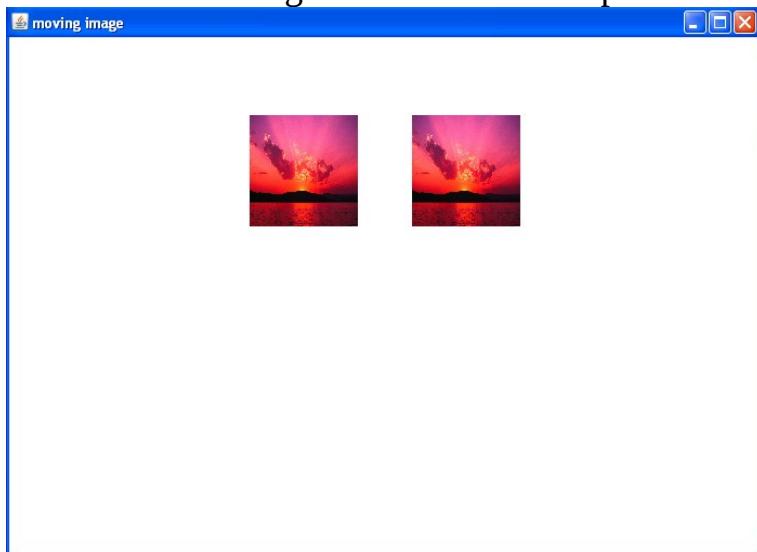
5. Writer a program to move the String ProFound.. As the String Moves the background colour should change.

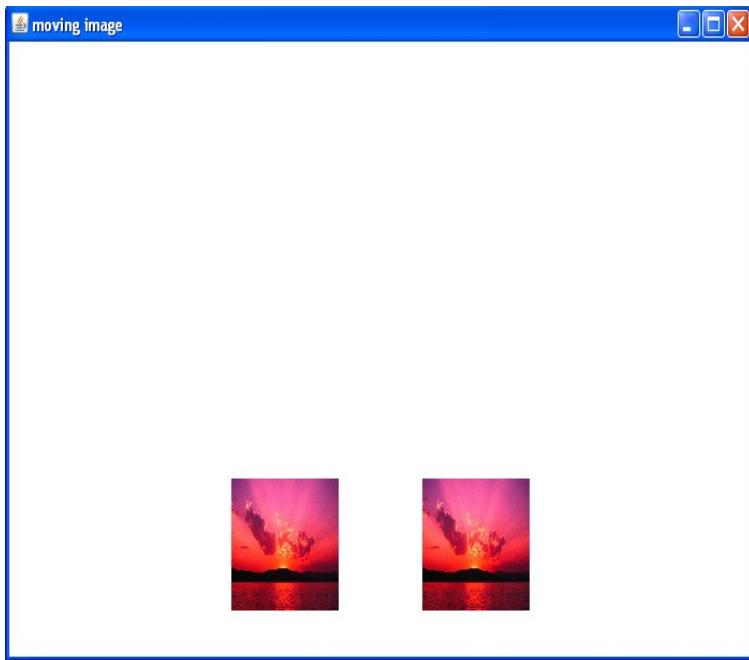
## Lab Manual

---



### 6. WAP to move the image in a Diamond shape





7. Write a Java program using Synchronized Threads, which demonstrates Producer Consumer concept.
8. Create a Banking Application simulating concurrent deposit and withdrawal. Use synchronized method for transaction.

### **Account Class**

Attributes: acno & balance

Methods: getter setter methods and `toString()`

### **Method signature**

```
public synchronized int withdraw(Account acc, int amount)  
{....}
```

```
public synchronized int deposit(Account acc, int amount)  
{....}
```

9. Create a Banking Application simulating concurrent deposit and withdrawal.  
Use synchronized block for transaction

### **Method signature**

```
public int withdraw (Account acc, int amount)  
{
```

```
synchronized (this)
{.....}
}
```

```
public int deposit(Account account, int amount)
{
    synchronized {....}
}
```

10. WAP to move three balls (Use wait(), notify() and notifyAll() of Object class)

# **Lab-15 Assignments**

**1.JDBC Api**  
**Connection**  
**Statement**  
**PreparedStatement**  
**ResultSet**

## **Assignments To Solve:**

1. WAP perform CRUD operation on Student(rollno,name,marks,address) table using JDBC APIs .(use Statement)
2. WAP perform CRUD operation on Book(id,name,author,price,category) table using JDBC APIs .(use PreparedStatement)

## **Advanced JDBC**

### **Assignment to solve:**

1. Create procedures AddnewCustomer() and SearchNameById() in mysql database  
call above procedures by jdbc's Callable statement
2. Write a java application to test Scrollable and nonScrollable resultsets, with Book entity
3. Write a java application to print the metadata above ResultSet as well as Database  
(use DatabaseMetadata,ResultSetMetaData Api)

## Lab Manual

### HTML & CSS

Q. Design the following registration form using HTML and CSS.

#### **Express Registration**

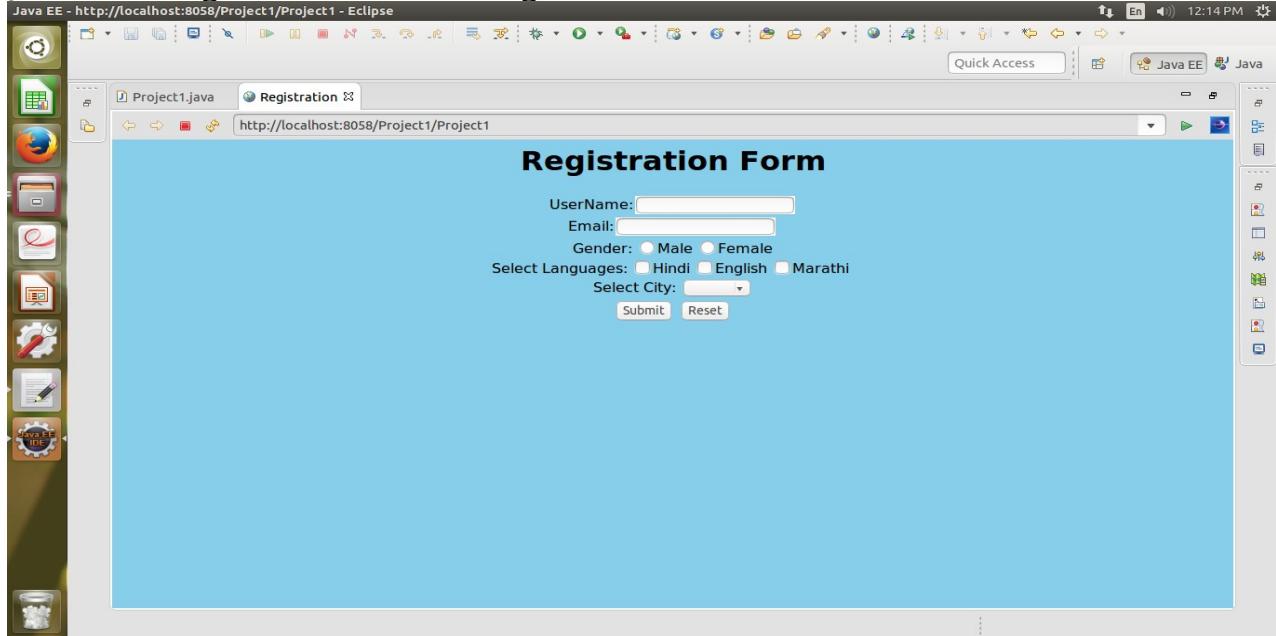
Please note that all fields that have an asterisk (\*) are required to process your registration.

Personal Details		Address	
Title:	(Mr/Mrs/Miss)	Address:	*
First Name:	*		
Last Name:	*	Town:	*
Email Address:	*	County/State:	*
Telephone:	*	Country:	United States <input type="button" value="▼"/>
Cell Phone:		Postcode:	*
<b>Comment</b>  <div style="border: 1px solid #ccc; height: 100px; width: 100%;"></div>			
Feedback	<input type="text"/>	Hear us from?	<input type="checkbox"/> Friend <input type="checkbox"/> Site Owner <input type="checkbox"/> Search Engine
<b>Security Details</b>			
Choose Password:	*	Confirm Password:	<input type="text"/> * <input type="button" value="Register"/>

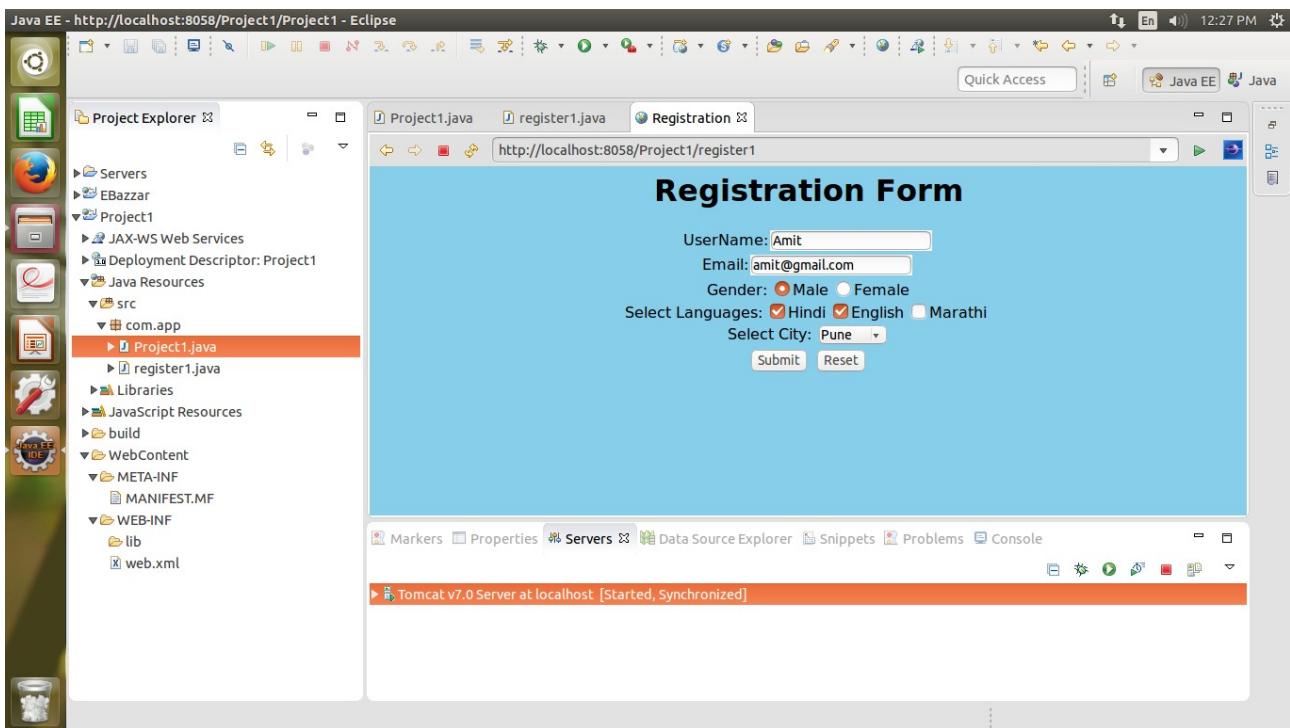
Q. Validate above required field by using form validation before click of register button.

# **Servlet Assignment**

**Q.1 Create Registration Form using Servlet.**



**and Display in Next Servlet.**



## Lab Manual

---

Q.2 Write a servlet that will take a string and show the reverse in the form. (

create three servlets,

FirstServlet take text input and submit it to SecondServlet,

The SecondServlet print that input in reverse,it has original button when click,it navigate to ThirdServlet and then

ThirdServlet print original text

)

Servlets-3 - Reverse2 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

Customize Links Video Login

http://localhost/servlet/com.javaranch.drive.Reverse2Servlet

Servlets-3 - Reverse2

Text:

submit

Done

Servlets-3 - Reverse2 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

Customize Links Video Login

http://localhost/servlet/com.javaranch.drive.Reverse2Servlet

Servlets-3 - Reverse2

Text:  The Tick sez "Spooooon!"

submit

Done

Feed in some text:

Press "submit" and get the form back with the text reversed:

## Lab Manual



Clicking "submit" again should give you the original text:  
Using Servlet basic API.

Q.3 Store email in web.xml as init-param name value pair. Write a Servlet to extract the initialization parameters inside the init method of your Servlet and display it within your service method.

Q.4 Store database specification into web.xml as context-param name value pair. Write a Servlet to extract the context parameters ,and display it within your service method.

Q5. Develop LoginApp as directed below flow  
Login.html[accept username , password and registration link ]  
ValidateServlet (check username and password is valid or not with database, depends

on that it will redirect to either HomeServlet or Login.html )

When you click on register link it will redirect you to Register.html and take all registration details, after submit it will save all detail using RegisterServlet

**Q.6 Create web application to pay online telephone bill using  
(Servlet,ServletConfig,ServletContext & RequestDispatcher with Database.)**

Hint:

Database BSNL

Table PayBill having Cust\_No,Cust\_Name & Current\_Bill

Servlet

Login.html

User enter contact number and login

DisplayServlet.java

Display CustomerName & CurrentBill which fetch from DB using  
ServletConfig & ServletContext.

Click on “Pay” button.

Pay.java

Display Bill payment successful with DB update.

Else

Back to Login if contact number is incorrect.

**Q.7 Display Customer BankDetails like Name,CustomerId**

**Account\_no using ServletFilter chaining concept.**

**CustomFilter—filter customerID**

**AuthenticateFilter – filter Account\_no.**

**Q.8 Develop BookCart Application using Servlet concept in incremental approach.**

**Scenario:**

Cyber Publishing Company is a popular publishing company in India. All the transactions of books happen manually till now. They want to sell books online due to increase No. of customers.

They need to develop an web based application to purchase books online.

Login Id <Login Id >	Category <Category >
----------------------	----------------------

## Lab Manual

---

Book Id	Book Name	Author Name	Price	Purchase
---	---	----	----	<input type="checkbox"/>
---	--	----	---	<input type="checkbox"/>
	Submit [ Hyperlink ]	Logout [Hyperlink ] [ Hyperlink]	Show Cart	

- Login ID must be taken from login page.
  - Book Category must be either Computer or Engg Books.
  - If Computer Books selected then display all the Books with all the details.
  - If Engg. Books selected then display all the Books with all the details.
  - Click on Logout Hyperlink will display the page with Thanks message and a hyperlink which will move the pointer to the login page again.
3. The customer needs to select the checkbox corresponding to **Purchase** if he is interested in that Book.
- Once the customer has selected all the Books for Purchase he needs to click the **Submit** Button.
  - On click of the Hyperlink **Show Cart** the **ShowCartPage** will be called.
- .

The screen of **ShowCartPage** is as follows:

Login Id		<Login Id >	
Book Id	Book Name	Price	Amount
-----	-----	----	100
-----	-----	----	200
		Total Amount	300

## **JSP Assignments**

Q.1 Write JSP Script which accepts user name and nick name from user. At first visit, display message “Hello user name” and for next successive requests, display “Hello nick name”.

Use username if visit count is odd and nick name if visit count is even. (use declaration scripting elements)

Q.2 Write jsp code to use page directives tag's **isErrorPage ,errorPage,import** attribute.

Q.3 Write a jsp code to use all jsp implicit object to read and set content with different scope.

Q.4 Using Scripting Element & Directive tag.

### **Problem Statement 1:**

ZBank is new generation bank they are planning to launch their website. They have approached you to create the banking application using jsp. As initial task to test your expertise in jsp programming they have asked you to create a simple welcome screen with a greeting message.

#### **Requirement Description:**

The welcome page should be named welcome.jsp. It should have the current date displayed “*dd-MM-yyyy*” format. The page should display a greeting based on the time of the day.

*if the time is less than 12.00*

*Print “Good Morning”*

*Else Print “Good Evening”*

The bank wishes to use 24hr time format.

Also as a matter of initial promo the bank wishes to keep track of the number of users

## Lab Manual

visiting the site and this value should be displayed to the user.

Develop a jsp page satisfying the above requirements.

The page should look like the screen shot design provided and the following test cases also needs to be executed.

### Hints:

- Use the following code for date calculations and use scriptlets

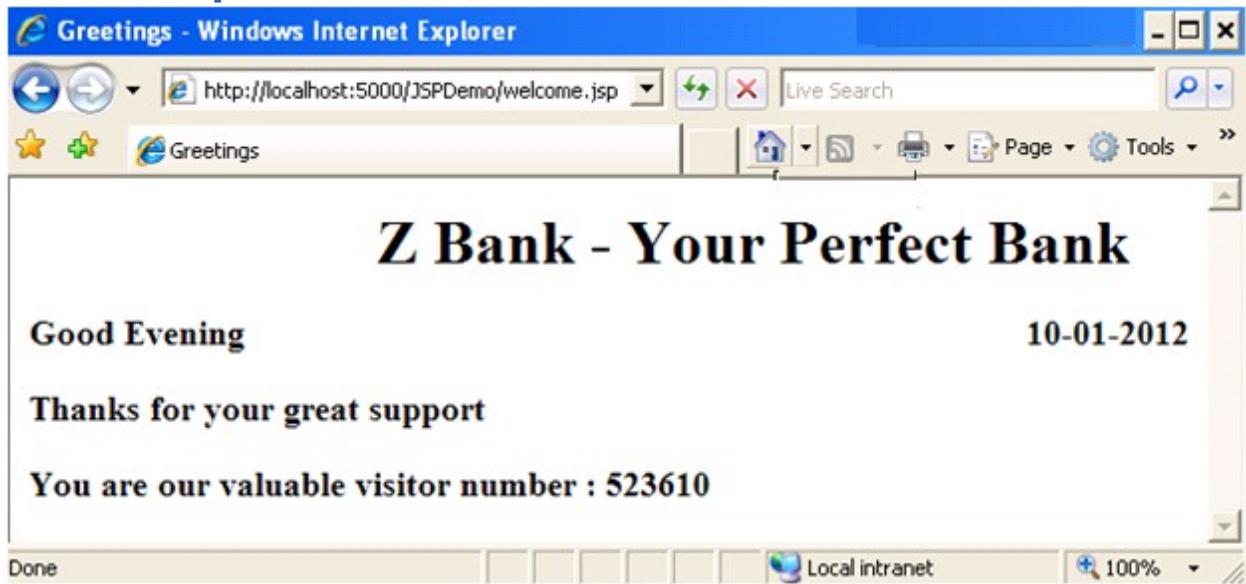
```
java.util.Calendar calendar = new java.util.GregorianCalendar();
java.util.Date date = calendar.getTime();
DateFormat format=new SimpleDateFormat("dd-MM-yyyy");
String dateval=format.format(date);
int hour = calendar.get(Calendar.HOUR_OF_DAY);
```

- Use declaration to declare page count variable.

**Test Cases:** Execute the below test cases after development.

11. **Test case #1:** Access the welcome.jsp it should display the current system date and display the appropriate message.
12. **Test case #2:** As you refresh the JSP page the counter needs to be incremented and should increase for each refresh.

## Screen Expected



---

## **Lab Manual**

---

Q.5 Write JSP code which adds the student education details in a database through Java bean.

Also display the student details who have secured first class in their graduation.

Q6. Write JSP code to perform CRUD operation on Product table.

Define links into index.html (add new Product,display products etc)

Q.7 Create custom tag which display User Name in its initial.

For eg Hrithik Rakesh Roshan ---> H.R.Roshan using either tag with body or attribute.

Q.8 Create custom tag which display UserName in uppercase with specified font color (use tag with body and attribute)

### 4. Ad Java Project

Using : Servlet + JSP+DB with MVC & DAO.

#### Problem Statement:

EBazaar is an online shopping portal where the users can purchase items online. They have approached you to create the application. In first phase they have asked you to create the Search module.

#### Requirement Description:

##### Search Module

The company deals with sales of three categories of products

- 9. Mobiles
- 10. Laptops
- 11. Watches

These are the sequence of steps the user does to purchase a product,

**Step 1:** User logs into the web application using the login.jsp. On successful login the page should be redirected to the home page.

**Step 2:** The home page (index.jsp) should list down the products Mobiles, Laptops and Watches as links.

**Step 3:** The user can select one of the above links and select the product they want to add to the cart.

**Step 4:** After adding the products to the cart the user can purchase the products in the cart.

#### Features:

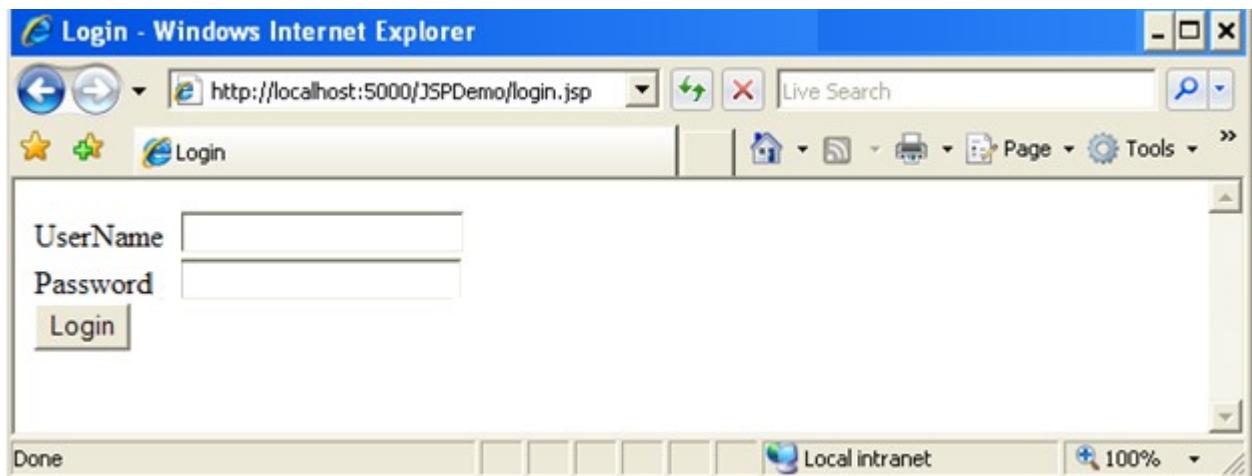
- Each product category page should provide an option for the user to navigate to the home page or to the cart page.
- The cart page should display all the products the user has added to his cart.
- The cart page should also provide the option to remove items from the cart..
- The cart page should also provide a feature for purchasing the products, clicking the purchase button should purchase all the products added by the user to the cart.
- The company also wishes to keep in track of the number of visitors visiting the site and to display it to all the users in all pages. The visitor count is incremented by for every new login. Say Jack logs into the application the count should show one. If he logs out and logs in again it should be incremented and displayed as 2.

#### Design:

Design the components as stated below.

8. **Login.jsp** : Login Page for the user

## Lab Manual



9. **Index.jsp** : Displays various categories of products



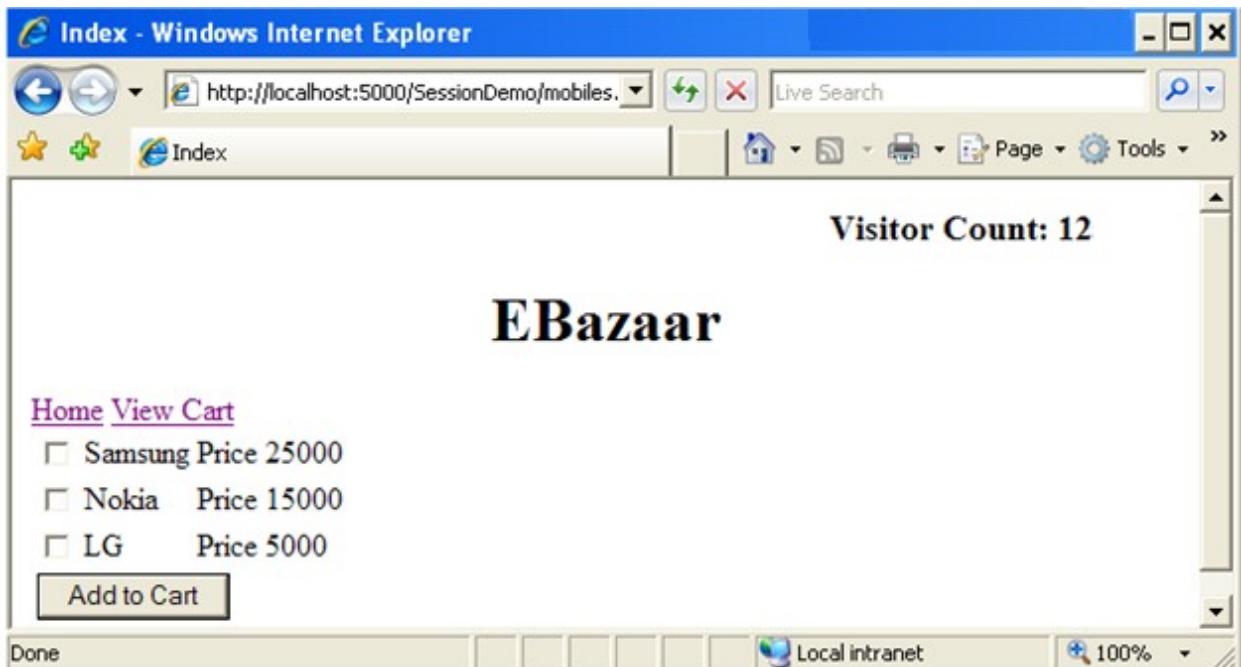
10. **Watch.jsp** : Displays the list of watches available for Sale

11. **Laptop.jsp** : Displays the list of Laptops available for Sale

12. **Mobiles.jsp** : Displays list of mobiles available for Sale

The **Mobile.jsp** should display a response similar to the below screen shot,

## Lab Manual



- User selects the appropriate device he needs to purchase using the check box and clicks Add to cart. This will add the selected products to carts page. Each product category page should provide an option for the user to navigate to the home page or to the cart page. A link “home” for the index page and “View Cart” for the Cart page should be included as per shown in the screen shot above.

**NOTE:** The *Laptop.jsp* and *Watches.jsp* should display a response HTML similar to the above screen shot and display some laptops and mobiles models respectively

13. **Cart.jsp** : Displays the products which was added by the user in the cart.

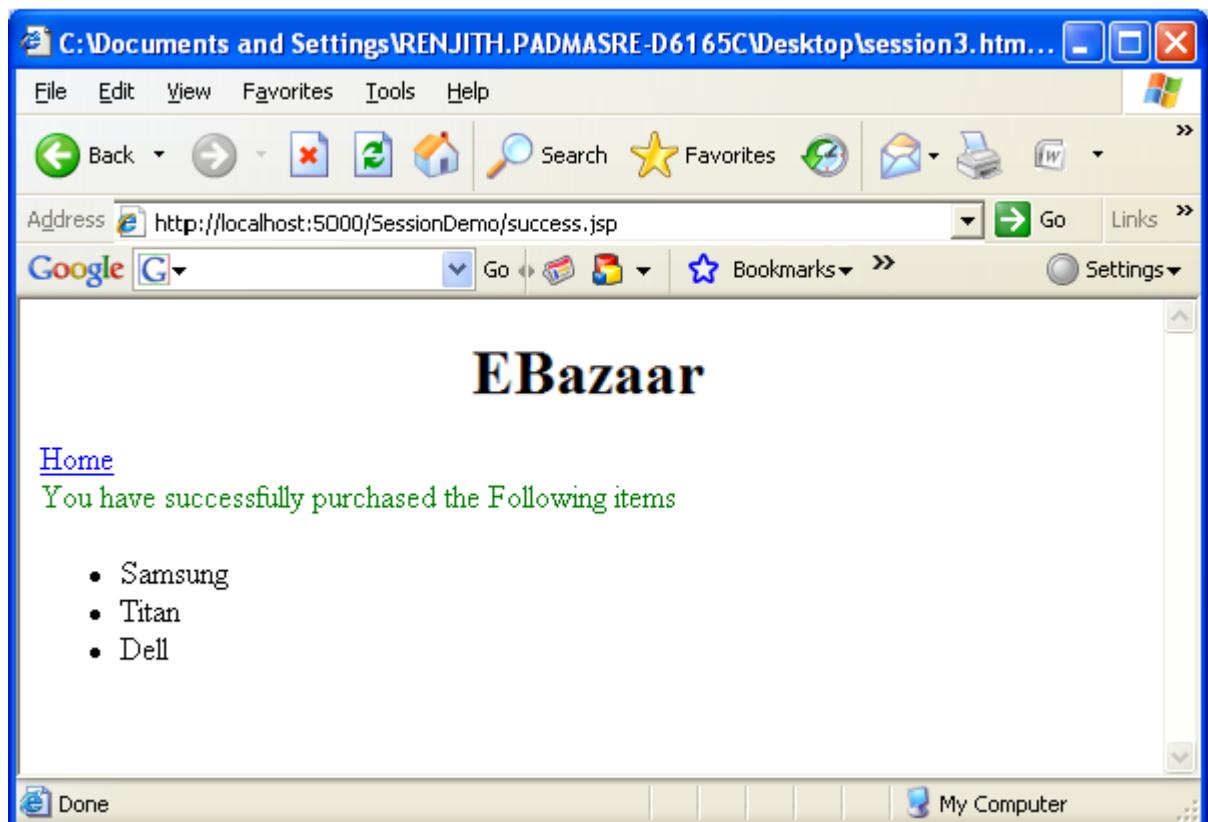


The user can either select products and **remove from the cart** or **purchase** the products.

The user can also drop some products from the page by selecting them and clicking "remove from cart". This should remove the product from the carts page.

On clicking the purchase button the response should be taken to a purchase successful page as illustrated below,

## Lab Manual



After purchase the products from the cart page should be automatically removed.

**Hint:**

4. Use HTTP session management technique to manage the products selected by user.  
Use Sessions add, remove API's to complete this case study.
5. Store the user count in application context to make it available across the application.
6. Use session to check whether the current user visits the index page for the first time in the current session.