Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400050

Division : C3

Batch : C31

Roll Number : 2003145

This is to certify that Mr. / ~~Miss~~ Ratlamwala Idris Ismail of Computers Department, Semester III with Roll No. 2003145 has completed a course of the necessary experiments in the subject **OOP-JAVA** under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the **year 2021-2022 .**

Teacher In-Charge: Prof. Juhi Janjua
Head of the Department: Prof. Tanuja Sarode

Date 07/12/2021 Principal

List of Experiments
S.E. (Comp.) Sem III
Subject: Object Oriented Programming with JAVA (CSL304)

Title of Experiments

Expt No.	Experiment	Date
1	Program on accepting input through keyboard	14/09/21
2	Program on basic programming constructs like looping and branching	27/09/21
3	Program based on Command Line arguments	28/09/21
4	Program on classes and objects	20/10/21
5	Program on 1D Array	25/10/21
6	Program on 2D Array	25/10/21
7	Program on Array of Objects	01/11/21
8	Program on method and construction overloading	08/11/21
9	Program on exceptional handling	18/11/21
10	Program on user defined exception	18/11/21
11	Program on packages	25/11/21
12	Program on Inheritance	08/11/21
13	Program on String	06/12/21
14	Program to create a GUI based application	06/12/21
	Written Assignments/Mini Project	

1	**Assignment 1:** Write a short note on: a. Features of JAVA b. JVM c. Wrapper Classes d. Life of a thread	07/12/21
2	**Assignment 2:** Write a short note on: a. Thread synchronization b. Abstract classes c. JDBC Drivers and Architecture d. Life cycle of an applet	07/12/21
3	**MINI PROJECT REPORT**	07/12/21

Idris Ratlamwala
C31 - 2003145

Experiment 1: Program on accepting input through keyboard

A.

Aim : WAP to check if an integer (Accepted from user via BufferedReader Class) is a two digit number or not .

Theory :

BufferedReader : BufferedReader is a Java class that reads text from the input stream. It buffers the characters so that it can get the efficient reading of characters, arrays, etc. It inherits the reader class and makes the code efficient since we can read the data line-by-line with the readline() method. The constructor of this class accepts an InputStream object as a parameter.

In general, we can configure *BufferedReader* to take any kind of input stream as an underlying source. We can do it using *InputStreamReader* and wrapping it in the constructor:

```
BufferedReader br =
        new BufferedReader(new InputStreamReader(System.in));
```

In the above example, we are reading from *System.in* which typically corresponds to the input from the keyboard. Similarly, we could pass an input stream for reading from a file or any imaginable type of textual input.

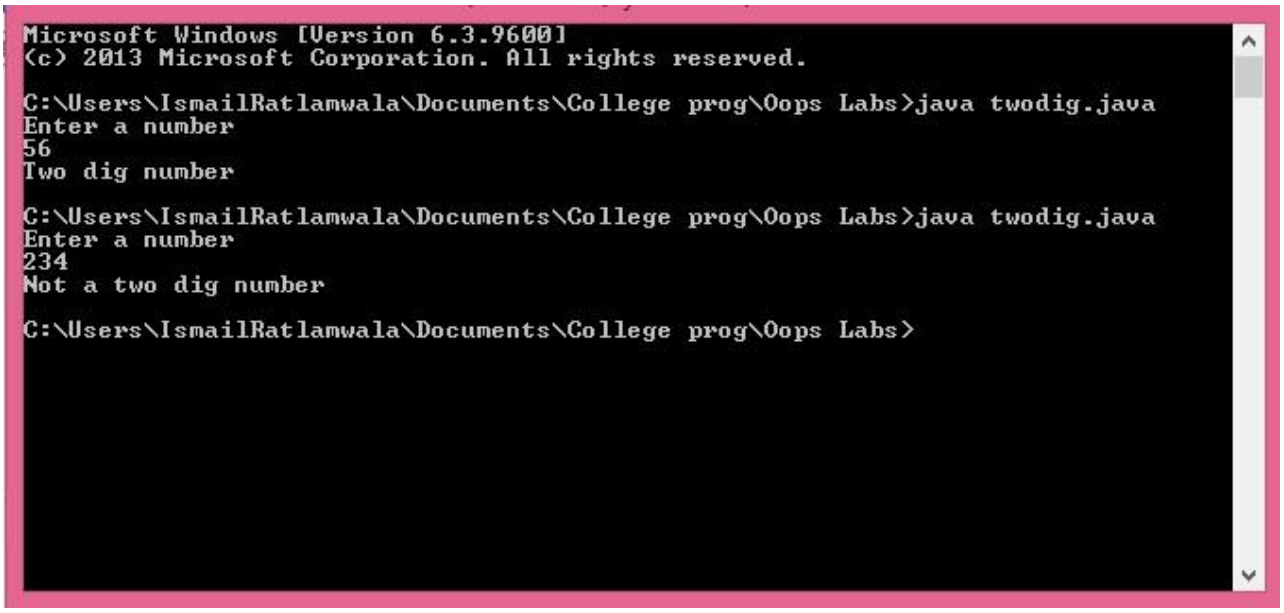
Syntax :

```
BufferedReader br =new BufferedReader(new
InputStreamReader(System.in));
String name = br.readLine();
int roll_no = Integer.parseInt(br.readLine());
```

Program :

```
import java.io.*;

public class twodig {      public static void
main(String[] args) throws IOException
    {
        BufferedReader br = new BufferedReader(new
InputStreamRe ader(System.in));
        System.out.println("Enter a
number");          int number =
Integer.parseInt(br.readLine());
        if(number<100 && number>9)
        System.out.println("Two dig number");
        else System.out.println("Not a two dig
number");
    }
}
```

Output :


Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs>java twodig.java
Enter a number
56
Two dig number
C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs>java twodig.java
Enter a number
234
Not a two dig number
C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs>

B.

Aim : WAP to print the Percentage range of a student as per following criteria for the grade accepted via Scanner Class

Percentage	**Grade**
0-60	F
61-70	D
71-80	C
81-90	B
91-100	A

Theory :

Scanner Class : Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program, though not very efficient.

- To create an object of Scanner class, we pass the object System.in, which represents the standard input stream from keyboard. We may pass an object of class File if we want to read input from a file.
- To read numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort() □ To read strings, we use nextLine().

- To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) function returns the first character in that string.

Syntax:

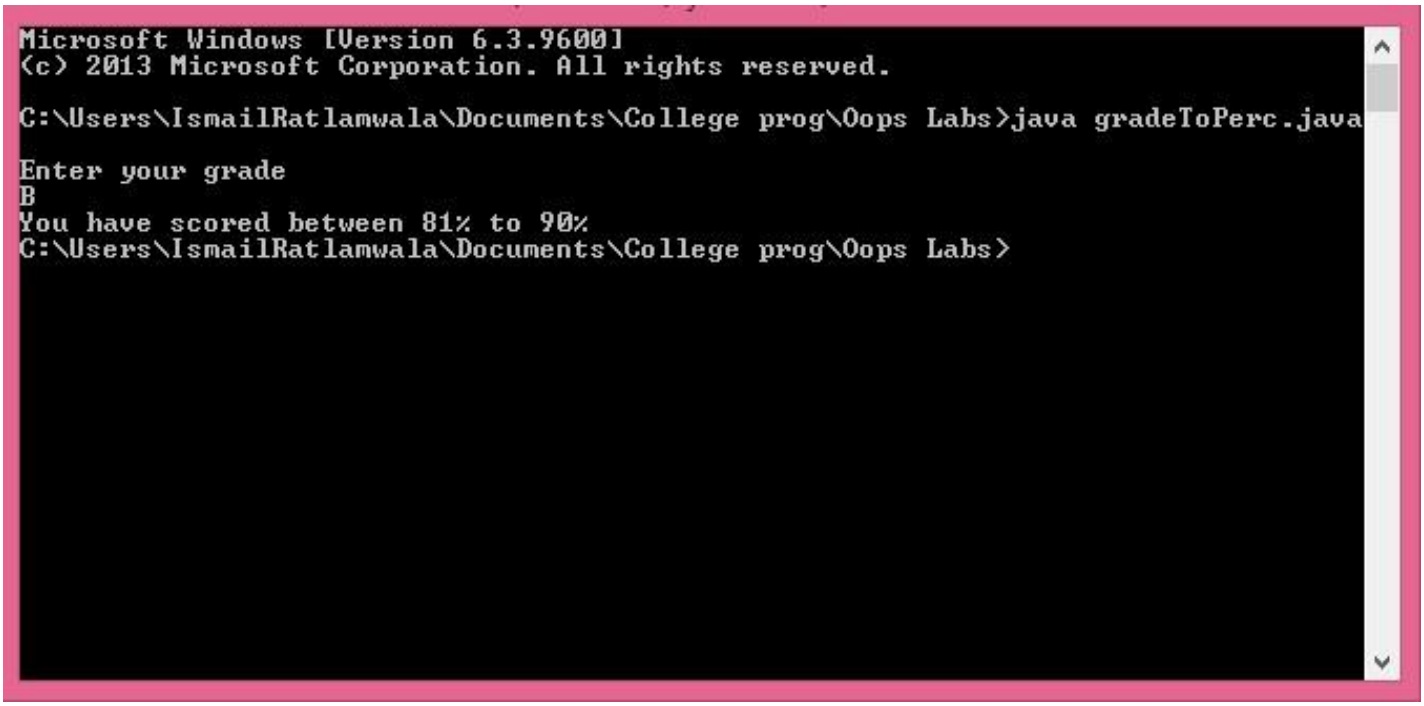
Scanner sc = new Scanner(System.in);
int number = sc.nextInt();

Program :

```
import
   java.util.Scanne
   r;

   public class gradeToPerc {
   public static void
   main(String[] args) {
   char grade;
           Scanner sc=new Scanner(System.in);
   System.out.println("Enter your grade ");
   grade=sc.next().charAt(0);
           System.out.print("You have scored between
   ");         switch(grade)

   {
   case
   'A':
               System.out.print("91%
   to 100%");              break;
```

```
case
'B':
            System.out.print("81%
to 90%");             break;

case
'C':
            System.out.print("71%
to 80%");             break;

case
'D':
            System.out.print("61%
to 70%");             break;

case
'F':
            System.out.print("0%
to 60%");             break;
        }
sc.close();
    }
}
```

Output:



C .

Aim : Admission to a Professional Course is based on following conditions :

i. Marks in mathematics >=60 ii.

Marks in Physics >=50

iii. Marks in Chemistry

>=40

iv. Total marks in three subjects >=200

Accept the marks in three subjects(use BufferedReader class) and decide if the student is eligible to get admission or no.

Theory :

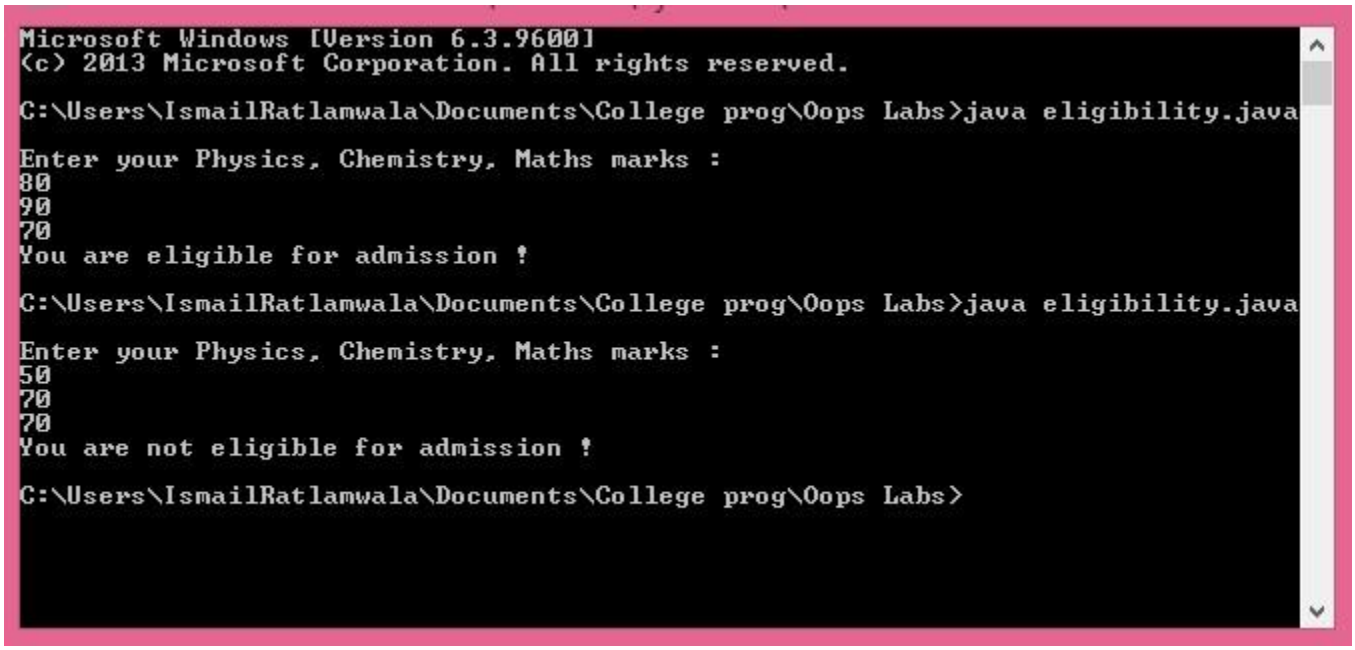
Advantages of BufferedReader class over Scanner class :

- *BufferedReader* is synchronized (thread-safe) while Scanner is not, hence cannot be used in Multi-threading .
- *Scanner* can accept carriage return character as a string/char hence it won't take input properly.
- *BufferedReader* allows for changing the size of the buffer while Scanner has a fixed buffer size .
- *BufferedReader* has a larger default buffer size of about 8Kb .
- *BufferedReader* is usually faster than *Scanner* because it only reads the data without parsing it .

Program :

```
import java.io.*;

public class eligibility {     public static
void main(String[] args) throws IOException{
int p,c,m,total;
        BufferedReader br= new BufferedReader(new
InputStreamRea der(System.in));
        System.out.println("Enter your Physics,
Chemistry, Maths ma rks :");
        p=Integer.parseInt(br.readLine());
c=Integer.parseInt(br.readLine());
m=Integer.parseInt(br.readLine());
total=p+c+m;          if(p>=50 && c>=40 && m>=60 &&
total>=200) System.out.pri ntln("You are eligible
for admission !");          else
System.out.println("You are not eligible for
admission !");
    }
}
```

Output :



Idris Ratlamwala
C31 - 2003145

Experiment 2: Programs on Basic programming constructs like branching and looping

Theory :

Branching : We know that Instruction those are written in java Language are Executed in Sequence wise or Step Wise as they are Written in Program. or these are Executed in Sequence Order. Decision Making statements are used when we wants to execute the statements as according to the user needs. A User can Change the Sequence of the Statements for Execution. Many Times we Wants to Execute the Set of Instructions to be Executed in one Situation and other Statements in other Situations For Executing the Statements in Specific Situation there are Some Decision Making Statements or Decision or Control Statements those are provided by the java Language. In The Decision First a Condition is checked if it is true then it Executes the next Instruction otherwise it Executes another Statements.

1) If else Statements : An if statement tests a particular condition; if the condition evaluates to true, a course-of-action is followed i.e. a statement or set-of-statements is executed. Otherwise (if the condition evaluates to false), the course-of-action is ignored and the statements in else block are performed (if any).

Syntax :

```
if(boolean_expression(ie. condition)) {
 /* statement(s) will execute if the boolean expression is true */
} else {
 /* statement(s) will execute if the boolean expression is false */
}
```

2) Switch Statements : When number of conditions (multiple conditions) occurs in a problem and it is very difficult to solve such type of complex problem with the help of ladder if statement, then there is need of such type of statement which should have different alternatives or different cases to solve the problem in simple and easy way. For this purpose switch statement is used.

Syntax :

```
switch(expression) {
case x:
// code block
 break;
case y:
 // code block
 break;
 default:
   // code block
}
```

Loooping : Loops are used to execute a set of statements repeatedly until a particular condition is satisfied. In Java we have three types of basic loops: for, while and do-while.

1) for loop : Java for loop is used to run a block of code for a certain number of times.

Syntax:

```
for (initialExpression; testExpression; updateExpression) {
        // code block to be executed
}
```

Here,

1. The initialExpression initializes and/or declares variables and executes only once.
2. The condition is evaluated. If the condition is true, the body of the for loop is executed.
3. The updateExpression updates the value of initialExpression.
4. The condition is evaluated again. The process continues until the condition is false.

2) while loop : The Java while loop is used to iterate a part of the program again and again. If the number of iteration is not known, then we can use while loop.

Syntax :

```
while (test expression) {
        // code block to be executed
}
```

3) do while loop : The do...while loop is similar to while loop. However, the body of do...while loop is executed atleast once before the test expression is checked.

Syntax :

```
do {
```

```
        // code block to be executed
} while(textExpression)
```

A.

Aim : WAP to check if a character is a vowel or not .

Program :

```
import java.io.*;

   public class vovels {
       public static void main(String[] args) throws IOException {
           BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
           System.out.println("Enter a character :");
           char c = Character.toLowerCase((char)br.read());
           if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u') System.out.println("Vovel");
           else System.out.println("consonant");
       }
   }
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java vovels.java
Enter a character :
I
Vovel
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java vovels.java
Enter a character :
v
consonant
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java vovels.java
Enter a character :
u
Vovel
```

B.

Aim : WAP to print a two dimensional table of squares of numbers from 1 to 25 using for loop .

Program :

```
public class twodTableSquare {
    public static void main(String[] args) {
        int i;
        for(i=1;i<=25;i++)
        {
            if(i*i<10) System.out.print(i*i+"   ");
            else if(i*i<100) System.out.print(i*i+"  ");
            else System.out.print(i*i+" ");
            if(i%5==0) System.out.println("");
        }
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java twodTableSquare.java
1   4   9   16  25
36  49  64  81  100
121 144 169 196 225
256 289 324 361 400
441 484 529 576 625
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs>
```

C.

Aim : WAP to find number of and sum of all integers greater than 100 and less than 200 that are divisible by 7 .

Program :

```
public class divisibleBy7 {
    public static void main(String[] args) {

        for(int i=100;i<=200;i++)
        {
            if(i%7==0) System.out.println(i);
        }
    }
}
```

Output :

```
\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java divisibleBy7.java
105
112
119
126
133
140
147
154
161
168
175
182
189
196
```

D.

Aim : WAP to print the following pattern,

```
    *
  * * *
* * * * *
  * * *
    *
```

Program :

```
public class pattern1 {
    public static void main(String[] args) {
        for(int i=1;i<=3;i++)
        {
            for(int j=1;j<=3-i;j++)
            {
                System.out.print("  ");
            }
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            for(int j=i;j>=2;j--)
            {
                System.out.print("* ");
            }
            System.out.println("");
        }
        for(int i=2;i>=1;i--)
        {
            for(int j=1;j<=3-i;j++)
            {
                System.out.print("  ");
            }
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            for(int j=i;j>=2;j--)
            {
                System.out.print("* ");
            }
            System.out.println("");
```

```
        }
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs> java pattern1.java
    *
  * * *
* * * * *
  * * *
    *
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs>
```

Experiment 3: Programs based on Command-line arguments

Theory :

Command Line Argument in Java is the information that is passed to the program when it is executed. The information passed is stored in the string array passed to the main() method and it is stored as a string. It is the information that directly follows the program's name on the command line when it is running.

There is no restriction on the number of java command line arguments.You can specify any number of arguments

These are captured into the String args of main method ie.

```
public static void main(String[] args)
```

To access the data we pass the index of that data to args[index], the datatype of arguments is string by default, However to get an int we can use : Integer.parseInt(args[index])

A.

Aim : WAP to accept student's details via command line and display on the screen.

Program :

```java
public class nameRollnoBycmd {
    public static void main(String[] args) {
        System.out.println("Name : "+ args[0]+ ""+ args[1]);
        System.out.println("Roll no : "+ args[2]);
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java nameRollnoBycmd.java Idris Ratlamwala 2003145
Name : Idris Ratlamwala
Roll no : 2003145
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java nameRollnoBycmd.java Priyansh Salian 2003147
Name : Priyansh Salian
Roll no : 2003147
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3>
```

B.

Aim : WAP to add two numbers accepted via command line.

Program :

```java
public class addnumBycmd {
    public static void main(String[] args) {
        int sum =  Integer.parseInt(args[1]) + Integer.parseInt(args[0]);
        System.out.println("Sum = "+sum);
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java addnumBycmd.java 8 20
Sum = 28
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java addnumBycmd.java 69 1
Sum = 70
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3>
```

C.

Aim : WAP to calculate minimum and maximum of three numbers accepted via Command line.

Program :

```java
public class maxOf3Bycmd {
    public static void main(String[] args) {

        int[] n = new int[3];
int max,min;

        for(int i=0;i<3;i++){
            n[i]= Integer.parseInt(args[i]);
        }
        max= min= n[0];

        for (int i : n) {
            max =(i>max) ? i : max ;
min =(i<min) ? i : min ;
        }
        System.out.println(max +" is largest \n"+min +" is smallest");

    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java maxOf3Bycmd.java 210 20 3320
3320 is largest
20 is smallest
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3> java maxOf3Bycmd.java 210 520 30
520 is largest
30 is smallest
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt3>
```

Idris Ratlamwala
C31 - 2003145

Experiment 4: Programs on class and objects

Theory :

Class :
A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers**: A class can be public or has default access.
2. **class keyword:** class keyword is used to create a class.
3. **Class name:** The name should begin with an initial letter (capitalized by convention).
4. **Body:** The class body surrounded by braces, { }.

Constructors are used for initializing new objects. Fields are variables that provides the state of the class and its objects, and methods are used to implement the behavior of the class and its objects.

Object :
It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, which interact by invoking methods. An object consists of :

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Declaring Objects :

When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

Example,
Employee e1;
If we declare reference variable (e1) like this, its value will be undetermined (null) until an object is actually created and assigned to it. Simply declaring a reference variable does not create an object.

Initializing an object :
The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor which is generally used to initialize object.

Hence correct way of creating the object is,
Employee e1 = new Employee();

A.

Aim : WAP to read and display details of an employee using single class and its object .

Program :

```
import java.util.Scanner;

public class employee {

    public static void main(String[] args) {
        Emp e1= new Emp();
        e1.register();
        e1.print();
    }
}

class Emp{
    int id,salary;
    String name;
    Scanner sc= new Scanner(System.in);
    void register() {
        System.out.println("Enter name:");
        name = sc.nextLine();
        System.out.println("\nEnter id number:");
        id = sc.nextInt();
        System.out.println("\nEnter salary:");
        salary = sc.nextInt();
    }
    void print(){
        System.out.println("\nName: "+name+"\nId: "+id+"\nSalary: "+salary);
    }
}
```

Output :

```
Enter name:
Idris Ratlamwala

Enter id number:
2003145

Enter salary:
80000

Name: Idris Ratlamwala
Id: 2003145
Salary: 80000
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt4>
```

B.

Aim : WAP to find maximum of three numbers using conditional operator, using two classes and function returning result .

Program :

```
import java.util.Scanner;

public class maxByclass {

    public static void main(String[] args) {
        Max.get();
        System.out.println("Max : "+ Max.calcMax());
    }
}

class Max{
    static int n1;
    static int n2;
    static int n3;
    static void get() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter 3 numbers : ");
        n1=sc.nextInt();
        n2=sc.nextInt();
        n3=sc.nextInt();
    }
    static int calcMax(){
        return (n1>n2&&n1>n3)?n1:((n2>n3)?n2:n3);
    }
}
```

Output :

```
Enter 3 numbers :
3 9 1
Max : 9
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt4> java maxByclass.java
Enter 3 numbers :
29 11 85
Max : 85
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt4>
```

Idris Ratlamwala
C31 - 2003145

Experiment 5: Programs on One Dimensional Arrays

Theory :

One Dimensional Array in java is always used with only one "[]". A one-dimensional array behaves likes a list of variables. You can access the variables of an array by using an index in square brackets preceded by the name of that array. Index value should be an integer.

Steps:

- Declaration of Array
- Construction of Array
- Initialization of Array

Declaration of One-dimensional array in java

Before using the array, we must declare it. Like normal variables, we must provide data type of array and name of an array. We also need to specify the name of an array so that we can use it later by name.

```
datatype[] arrayName;
Or
datatype arrayName[];
Or
datatype []arrayName;
```

- **datatype** can be a **primitive data type**(int, char, Double, byte etc.)or **Non-primitive data** (Objects).
- **arrayName** is the name of an array
- **[]** is called subscript.

Eg.

```
int[] number
```

Construction of One-dimensional array in java

For the creation of an array new keyword is used with a **data type** of array. we must specify the size of the array. The size should be an integer value or a variable that contains an integer value. How many elements you can store an array directly depends upon the size of an array.

```
arrayName = new DataType[size];
```

Note: When we are creating a new array the elements in the array will automatically be initialized by their default values. For e.g. : **zero** (int types), **false** (boolean), or **null** (for object types).

Initialization of One-dimensional array in java

To initialize the Array we have to put the values at each index of array. Hence we use single for loop to initialize array.

Eg.

```
for(int i=0 ;i<n ;i++) array[i]= sc.nextInt();
```

A.

Aim : WAP to count number of even and odd elements from an array.

Program :

```
    import java.util.Scanner;

    public class OddEven {

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the number of elements in the array :");
            int n= sc.nextInt();
            int[] a = new int[n];
            System.out.println("Enter the elements:");

            for(int i=0;i<n;i++) a[i]= sc.nextInt();

            int even=0,odd=0;
            for(int i=0;i<n;i++){ //this could be done in above loop, but then
there is no meaning of Aim.
                if(a[i]%2==0) even++;
                else odd++ ;
            }
            System.out.println("Number of odd elements:"+odd+"\nNumber of even
elements:"+even);
        }
    }
```

Output :

```
Enter the number of elements in the array :7
Enter the elements:
11 22 33 44 55 66 77
Number of odd elements:4
Number of even elements:3
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt5>  & 'c:\Users\IsmailRatlamwala\.
vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-
16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'C:\Use
rs\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceStorage\c58f9da9765800ca506aa216f07b82a4\redh
at.java\jdt_ws\expt5_89ba4f5a\bin' 'OddEven'
Enter the number of elements in the array :4
Enter the elements:
123 34 578 976
Number of odd elements:1
Number of even elements:3
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt5>
```

B.

Aim : WAP to count total marks and highest marks obtained by a student.

Program :

```
   import java.util.Scanner;

    public class marks {

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int[] a = new int[6];
            System.out.println("Enter the marks of 6 subjects(out of 100):");

            for(int i=0;i<6;i++) a[i]= sc.nextInt();

            int sum=0,highest=0;
            for(int i=0;i<6;i++){
                sum=sum+a[i];
                highest=(highest<a[i]) ? a[i]:highest;
            }
            System.out.println("Total marks scored :"+sum+"/600\nHighest marks
are scored is :"+highest);
        }
    }
```

Output :

```
Enter the marks of 6 subjects(out of 100):
45 56 67 78 89 90
Total marks scored :425/600
Highest marks are scored is :90
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt5>  c:; cd 'c:\Users\IsmailRatlamw
ala\Documents\College prog\Oops Labs\expt5'; & 'c:\Users\IsmailRatlamwala\.vscode\extensions\vscjava
.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX
:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala\AppDa
ta\Roaming\Code\User\workspaceStorage\c58f9da9765800ca506aa216f07b82a4\redhat.java\jdt_ws\expt5_89ba
4f5a\bin' 'marks'
Enter the marks of 6 subjects(out of 100):
87 89 98 94 82 97
Total marks scored :547/600
Highest marks are scored is :98
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt5>
```

Idris Ratlamwala
C31 - 2003145

Experiment 6: Programs on Two Dimensional Arrays

Theory :

The simplest of the multi-dimensional array is a **two-dimensional array**.
A simple definition of 2D arrays is: A 2D array is an array of one-dimensional arrays.

In Java, a two-dimensional array is stored in the form of rows and columns and is represented in the form of a matrix.

The general declaration of a two-dimensional array is,
data_type [] [] array_name;

Here,
data_type = data type of elements that will be stored in an array.
array_name = name of the two-dimensional array.

A 2D array can be created using new as follows:
data_type[][] array_name = new data_type[row_size][column_size];

Here,
row_size = number of rows an array will contain.
column_size = number of columns array will contain.

An integer array named 'myarray' of 3 rows and 2 columns can be declared as below.
int [][] myarray = new int[3][2];

Initialize 2d Array
Similar to 1D array, 2 nested for loops can be used to initialize a 2d array.
for(int i=0;i<3;i++) for(int j=0;j<2;j++) myarray[i][j]=sc.nextInt();

A.

Aim : WAP to find Transpose of a Matrix (One class ,only main).

Program :

```
       import java.util.Scanner;

    public class Transpose {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the number of rows and column in matrix
:");
            int m=sc.nextInt();
            int n=sc.nextInt();
            int[][] a= new int[m][n];
            int[][] b= new int[n][m];

            System.out.println("Enter the elements of matrix :");

            for(int i=0;i<m;i++) for(int j=0;j<n;j++) a[i][j]=sc.nextInt();

            System.out.println("Transposed matrix :");
            for(int i=0;i<m;i++) for(int j=0;j<n;j++) b[j][i]=a[i][j];

            for(int i=0;i<n;i++) {
                for(int j=0;j<m;j++)
                System.out.print(b[i][j]+" ");
                System.out.println();
            }
        }
    }
```

Output :

```
PS C:\Users\IsmailRatlamwala>  & 'c:\Users\IsmailRatlamwala\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Pr
ogram Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala
\AppData\Local\Temp\vscodesws_6d568\jdt_ws\jdt.ls-java-project\bin' 'Transpose'
Enter the number of rows and column in matrix :2 3
Enter the elements of matrix :
1 2 3
4 5 6
Transposed matrix :
1 4
2 5
3 6
```

B.

Aim : WAP to Pass a 2D Matrix to a function which determines if it is a square matrix. If not, program should come to end else the program should find sum of all diagonal elements of a Matrix.

Program :

```
    import java.util.Scanner;

public class sumOfDiagonal {

    public static int sum(int a[][]) {
        if(a.length==a[0].length){
            int sum=0;
            for(int i=0;i<a.length;i++)
            for(int j=0;j<a[0].length;j++)
            sum=(i==j)?(sum+a[i][j]):sum;
            return sum;
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows and column in matrix :");
        int m=sc.nextInt();
        int n=sc.nextInt();
        int[][] a= new int[m][n];

        System.out.println("Enter the elements of matrix :");

        for(int i=0;i<m;i++) for(int j=0;j<n;j++) a[i][j]=sc.nextInt();

        int sum=sum(a);
        if(sum==-1) System.out.println("Given matrix is not a square matrix");
        else System.out.println("Sum of diagonal elements : "+sum);
    }
}
```

Output :

```
Enter the number of rows and column in matrix :2 3
Enter the elements of matrix :
1 2 3
4 5 6
Given matrix is not a square matrix
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt6> java sumOfDiagonal.java
Enter the number of rows and column in matrix :3 3
Enter the elements of matrix :
1 2 3
4 5 6
7 8 9
Sum of diagonal elements : 15
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt6>
```

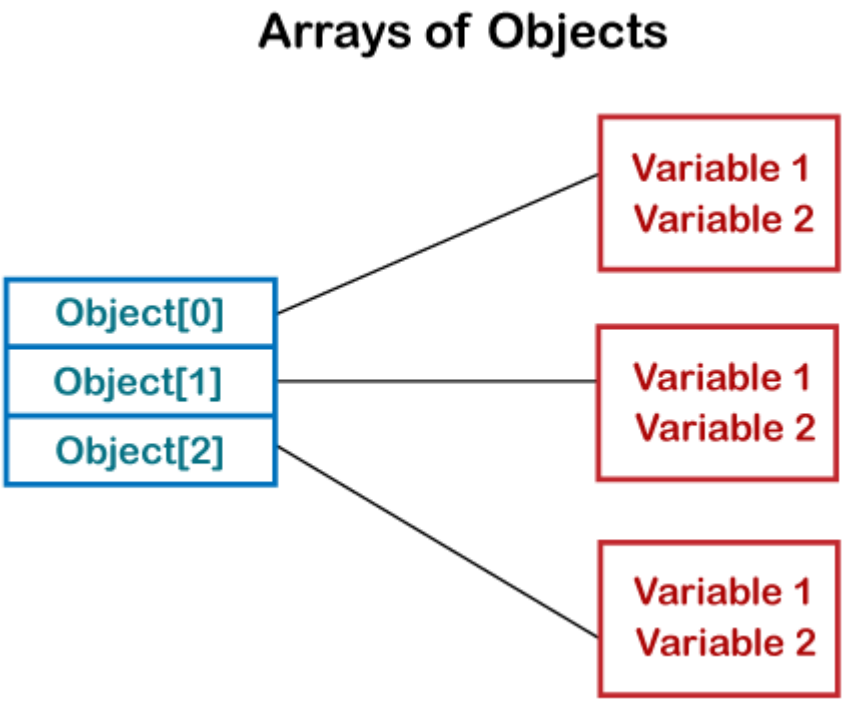
Idris Ratlamwala
C31 - 2003145

Experiment 7: Programs on Array of Objects

Theory :

Array of objects :

Java is an object-oriented programming language. Most of the work done with the help of **objects**. We know that an array is a collection of the same data type that dynamically creates objects and can have elements of primitive types. Java allows us to store objects in an array. In Java, the class is also a user-defined data type. An array that conations **class type elements** are known as an **array of objects**. It stores the reference variable of the object.



Creating an Array of Objects

Before creating an array of objects, we must create an instance of the class by using the new keyword. We can use any of the following statements to create an array of objects.

Syntax:

1. ClassName obj[] = new ClassName[array_length];

Or

2. ClassName[] objArray;

Or

3. ClassName objeArray[];

Suppose, we have created a class named Employee. We want to keep records of 20 employees of a company having three departments. In this case, we will not create 20 separate variables. Instead of this, we will create an array of objects, as follows.

1. Employee department1[20];
2. Employee department2[20];
3. Employee department3[20];

The above statements create an array of objects with 20 elements.

Initializing the array Of Objects

Once the array of objects is instantiated, we have to initialize it with values. As the array of objects is different from an array of primitive types, we cannot initialize the array in the way we do with primitive types.
In the case of an array of objects, each element of array i.e. an object needs to be initialized. One way to initialize the array of objects is by using the constructors. When we create actual objects, we can assign initial values to each of the objects by using for loop. We can also have a separate member method in a class that will assign data to the objects.

Eg.

```
for(int i=0 ; i<20 ; i++)  department1 [i]=new Employee ();
```

A.

Aim :

WAP to accept details of 5 employees like name, id, nohr. Depending upon the number of hours a prson has worked, calculate his wages for a particular day @100 Rs. Per hr. Display the information in tabular format as:

Id	Name	No. of Hours	Wages

Also display the details of the employee who got highest payment amongst all

Program :

```
import java.util.Scanner;

class Emp{
    int id,hours,wages;
    String name;
}

public class Employee {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n=5,maxWage=0;
        Emp[] e = new Emp[n];

        for(int i=0;i<n;i++){
            e[i]=new Emp();
            System.out.print("\nEnter name : ");
            e[i].name=sc.next();
            System.out.print("Enter ID : ");
            e[i].id=sc.nextInt();
            System.out.print("Enter no. of hours : ");
            e[i].hours=sc.nextInt();
        }
        System.out.println("\nId\t| Name\t| Hours\t| Wages\n-------|------|------|------
");
        for(int i=0;i<n;i++){
            e[i].wages=e[i].hours*100;
```

```
            System.out.println(e[i].id+"\t| "+e[i].name+"\t| "+e[i].hours+"\t|
"+e[i].wages);
            maxWage=(e[i].wages>e[maxWage].wages) ? i : maxWage;
        }
        System.out.println("\nDetails of employee with highest wage");
        System.out.println("Name : "+e[maxWage].name+"\nId : "+e[maxWage].id+"\nNo of
hours : "+e[maxWage].hours+"\nWage : "+e[maxWage].wages);

    }
}
```

Output :

```
Enter name : Idris
Enter ID : 234
Enter no. of hours : 8
```

```
Enter name : Yash
Enter ID : 456
Enter no. of hours : 7

Enter name : Anesh
Enter ID : 678
Enter no. of hours : 6

Enter name : Arya
Enter ID : 259
Enter no. of hours : 5

Enter name : Adil
Enter ID : 146
Enter no. of hours : 4

Id       | Name   | Hours | Wages
---------|--------|-------|-------
234      | Idris  | 8     | 800
456      | Yash   | 7     | 700
678      | Anesh  | 6     | 600
259      | Arya   | 5     | 500
146      | Adil   | 4     | 400

Details of employee with highest wage
Name : Idris
Id : 234
No of hours : 8
Wage : 800
```

B.

Aim : For Annual Examination results of 5 students, taking into consideration marks obtained in three subjects, WAP to determine

i . Determine Total marks obtained by each student

ii. The student who obtained highest total marks.

Program :

```
import java.util.Scanner;

class Student{
    int phy,chem,math,total;
    Scanner sc = new Scanner(System.in);

    Student(int i){
        System.out.println("Student "+(i+1)+" enter your marks in ");
        System.out.print("physics : ");
        phy=sc.nextInt();
        System.out.print("Chemistry : ");
        chem=sc.nextInt();
        System.out.print("Maths : ");
        math=sc.nextInt();
        total= phy+chem+math;
    }
}

public class AnnualExamination {
    public static void main(String[] args) {
        int n=5,max=0;
        Student[] s = new Student[n];

        for(int i=0;i<n;i++) s[i] =new Student(i);

        System.out.println("\nStudent\tPhy\tChem\tMath\tTotal\n-----------------------
----------------");
        for(int i=0;i<n;i++){
            System.out.println((i+1)+"\t"+s[i].phy+"\t"+s[i].chem+"\t"+s[i].math+"\t"+
s[i].total);
            max = (s[i].total>s[max].total) ? i : max;
        }
        System.out.println("\nHighest marks("+s[max].total+") are scored by student
"+(max+1)+".");
    }
}
```

Output :

```
Student 1 enter your marks in
physics : 79
Chemistry : 45
Maths : 78
Student 2 enter your marks in
physics : 23
Chemistry : 78
Maths : 89
Student 3 enter your marks in
physics : 56
Chemistry : 89
Maths : 90
Student 4 enter your marks in
physics : 67
Chemistry : 34
Maths : 78
Student 5 enter your marks in
physics : 47
Chemistry : 68
Maths : 36

Student Phy     Chem    Math    Total
---------------------------------------
1       79      45      78      202
2       23      78      89      190
3       56      89      90      235
4       67      34      78      179
5       47      68      36      151

Highest marks(235) are scored by student 3.
```

Idris Ratlamwala
C31 - 2003145

Experiment 8: Program on method and constructor overloading

Theory :

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

For example the argument list of a method add(int a, int b) having two parameters is different from the argument list of the method add(int a, int b, int c) having three parameters.

Three ways to overload a method

In order to overload a method, the argument lists of the methods must differ in either of these:

1. Number of parameters.
For example: This is a valid case of overloading

```
add(int, int)
add(int, int, int)
```

2. Data type of parameters.
For example:

```
add(int, int)
add(int, float)
```

3. Sequence of Data type of parameters.
For example:

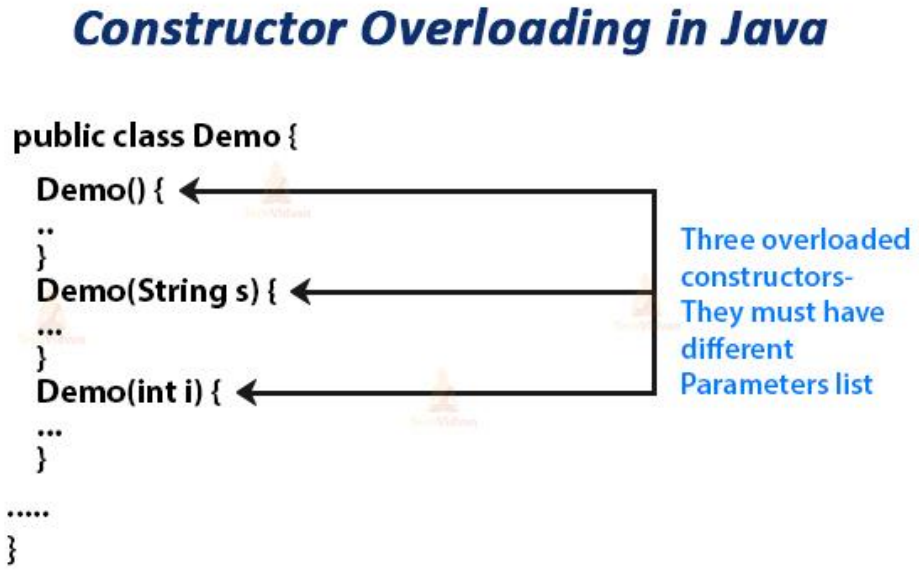
```
add(int, float)
add(float, int)
```

Invalid case of method overloading:
When I say argument list, I am not talking about return type of the method, for example if two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw compilation error.

```
int add(int, int)
float add(int, int)
```

Like methods, constructors can also be overloaded

Constructor overloading is a concept of having more than one constructor with different parameters list, in such a way so that each constructor performs a different task. For e.g. Vector class has 4 types of constructors. If you do not want to specify the initial capacity and capacity increment then you can simply use default constructor of Vector class like this Vector v = new Vector(); however if we need to specify the capacity and increment then we call the parameterized constructor of Vector class with two int arguments like this: Vector v= new Vector(10, 5);



A.

Aim : Calculate area of different shapes (Square, Rectangle, Circle) using method overloading and multiple class concept.

Program :

```
    class Area{

        int calculateArea(int l){
            return l*l;
        }
        int calculateArea(int l,int b){
            return l*b;
        }
        double calculateArea(float r){
            return 3.142*r*r;
        }
    }

    public class AreaMethodOL {

        public static void main(String[] args) {

            Area a1= new Area();

            System.out.println("Area of square with sides 12 is
"+a1.calculateArea(12));
            System.out.println("Area of rectangle with sides 12 and 4 is
"+a1.calculateArea(12,4));
            System.out.println("Area of circle with radius 2 is
"+a1.calculateArea((float)2));

        }
    }
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt8>  c:; cd '
lege prog\Oops Labs\expt8'; & 'c:\Users\IsmailRatlamwala\.vscode\extensions\vs
launcher.bat' 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDe
ing=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspac
bb\redhat.java\jdt_ws\expt8_89ba4f5d\bin' 'AreaMethodOL'
Area of square with sides 12 is 144
Area of rectangle with sides 12 and 4 is 48
Area of circle with radius 2 is 12.568
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt8>
```

B.

Aim : Calculate area of different shapes (Square, Rectangle , Circle) using constructor overloading and multiple class concept.

Program :

```
    class Areas{
    Areas(int l){
        System.out.println("Area of square is "+l*l);
    }
    Areas(int l,int b){
        System.out.println("Area of square is "+l*b);
    }
    Areas(float r){
        System.out.println("Area of square is "+3.142*r*r);
    }
}

public class AreaConstOL {
    public static void main(String[] args) {
        new Areas(12);  //Square
        new Areas(12*4);  //Rectangle
        new Areas((float)5); //Circle
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt8>  c:;
lege prog\Oops Labs\expt8'; & 'c:\Users\IsmailRatlamwala\.vscode\extension
launcher.bat' 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCo
ing=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\work
bb\redhat.java\jdt_ws\expt8_89ba4f5d\bin' 'AreaConstOL'
Area of square is 144
Area of square is 2304
Area of square is 78.55
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt8>
```

Idris Ratlamwala
C31 - 2003145

Experiment 9: Program on Exception handling

Theory :

An **Exception** is an unwanted event that interrupts the normal flow of the program. When an exception occurs program execution gets terminated. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled in Java. By handling the exceptions we can provide a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

If an exception occurs, which has not been handled by programmer then program execution gets terminated and a system generated error message is shown to the user.

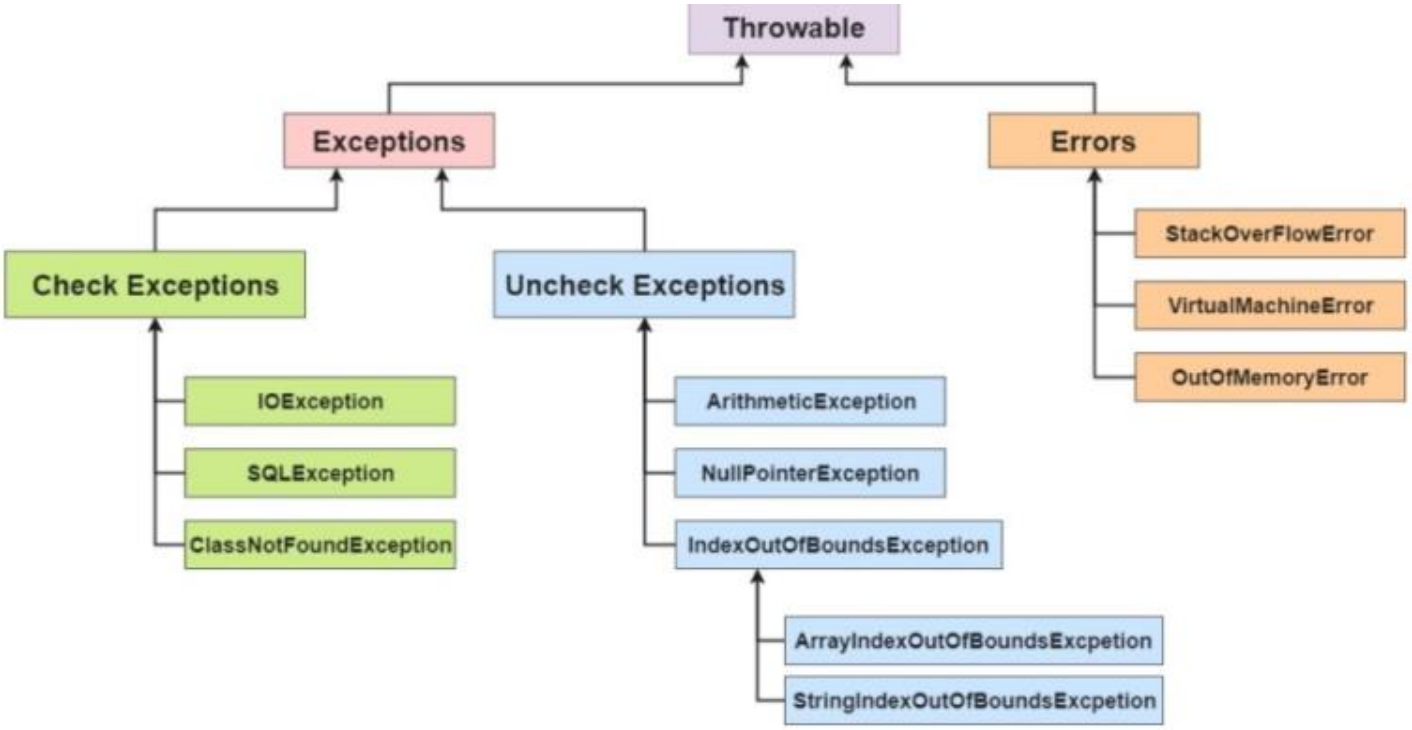
Advantage of exception handling
Exception handling ensures that the flow of the program doesn't break when an exception occurs. For example, if a program has bunch of statements and an exception occurs mid way after executing certain statements then the statements after the exception will not execute and the program will terminate abruptly.
By handling we make sure that all the statements execute and the flow of program doesn't break.

There are two types of exceptions in Java:
1)Checked exceptions
2)Unchecked exceptions

Checked exceptions
All exceptions other than Runtime Exceptions are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not. If these exceptions are not handled/declared in the program, you will get compilation error. For example, SQLException, IOException, ClassNotFoundException etc.

Unchecked Exceptions

Runtime Exceptions are also known as Unchecked Exceptions. These exceptions are not checked at compile-time so compiler does not check whether the programmer has handled them or not but it's the responsibility of the programmer to handle these exceptions and provide a safe exit. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.



Try block :

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.

Syntax of try block

```
try{
   //statements that may cause an exception
}
```

While writing a program, if we suspect that certain statements in a program can throw a exception, enclosed them in try block and handle that exception

Catch block :

A catch block is where you handle the exceptions, this block must follow the try block. A single try block can have several catch blocks associated with it. You can catch different exceptions in different catch blocks. When an exception occurs in try block, the corresponding catch block that handles that particular exception executes. For example if an arithmetic exception occurs in try block then the statements enclosed in catch block for arithmetic exception executes.

Syntax of try catch in java

```
try
{
     //statements that may cause an exception
}
catch (exception(type) e(object))
{
     //error handling code
}
```

A.

Aim : WAP to catch any three built-in exceptions.

Program :

```
public class ExceptionHandling{
    public static void main(String[] args) {
        System.out.println("\nArithmetic Exception :");
        try {
            int a=10,b=0;
            int c=a/b;
            System.out.println(c);
        } catch (ArithmeticException e) {
            System.out.println(e);
        }

        System.out.println("\nArray index out of bounds Exception :");
        try {
            int[] a = {10,20,30,40,50};
            System.out.println(a[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e);
        }

        System.out.println("\nNull pointer Exception :");
        try{
            StringBuffer[] str = new StringBuffer[4];
            str[0].append("Hello");
            System.out.println(str[0]);
        } catch(NullPointerException e){
            System.out.println(e);
        }
    }
}
```

Output :

```
Arithmetic Exception :
java.lang.ArithmeticException: / by zero

Array index out of bounds Exception :
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5

Null pointer Exception :
java.lang.NullPointerException: Cannot invoke "java.lang.StringBuffer.append(String)" because "str[0]" is null
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt9>
```

Experiment 10: Program on user defined exception

Theory :

The predefined exceptions are used to handle errors occurring in the program.
But sometimes the programmer wants to create his own customized exception as per the requirements of the application which is called **user-defined exception** or custom exception.

Custom exceptions in Java are those exceptions that are created by a programmer to meet the specific requirements of the application. That's why it is also known as user-defined exception in Java.

For example:

1. A banking application, a customer whose age is lower than 18 years, the program throws a custom exception indicating "needs to open joint account".
2. Voting age in India: If a person's age entered is less than 18 years, the program throws "invalid age" as a custom exception.

Syntax:

We do not have any particular syntax for Java user-defined exception,

Below is the code which will help to Create a User-defined exception class,

```
class SampleException{
public static void main(String args[]){
try{
throw new UserException(<value>); // used to create new
exception and throw
}
catch(Exception e){
System.out.println(e);
}
}
}
class UserException extends Exception{
// code for exception class
}
```

Here, while creating an exception class, it needs to be extended from java. lang.Exception.

An exception is an event that leads to sudden termination of the program during execution at runtime.

A.

Aim : WAP to accept any integer from the user & if the entered number is not any of 5 or 6 or 7 then create an exception & catch it.

Program :

```
import java.util.*;

class MyException extends Exception {
    MyException(String s) {
        super(s);
    }
}

public class InvalidNumExcep {

    static void check(int num) throws MyException {
        if (num != 5 && num != 6 && num != 7) {
            throw new MyException("Invalid input");
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter a number : ");
            int num = sc.nextInt();
            check(num);
            System.out.println("Valid input");
        } catch (Exception e) {
            System.out.println(e);
        }
        sc.close();
    }
}
```

Output :

```
Enter a number : 4
MyException: Invalid input
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt10>  c:; cd 'c:\Users\IsmailRatlamwala\Documents\Co
llege prog\Oops Labs\expt10'; & 'c:\Users\IsmailRatlamwala\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\script
s\launcher.bat' 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.enc
oding=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceStorage\7a2e4b98f843524d3509bbcc9a63
03b4\redhat.java\jdt_ws\expt10_ad8f9b9a\bin' 'InvalidNumExcep'
Enter a number : 6
Valid input
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt10>  c:; cd 'c:\Users\IsmailRatlamwala\Documents\Co
llege prog\Oops Labs\expt10'; & 'c:\Users\IsmailRatlamwala\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\script
s\launcher.bat' 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.enc
oding=UTF-8' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceStorage\7a2e4b98f843524d3509bbcc9a63
03b4\redhat.java\jdt_ws\expt10_ad8f9b9a\bin' 'InvalidNumExcep'
Enter a number : 8
MyException: Invalid input
```

B.

Aim : WAP to create an exception 'PayOutOfBounds' when the basic pay paid to the Superintendent it is less than 25,000 and greater than 50,000.

Program :

```
    import java.util.Scanner;

    class PayOutOfBoundsException extends Exception{
        PayOutOfBoundsException(String s){
            super(s);
        }
    }

    public class PayOutOfBounds {
        static void basicPayCheck(int salary)throws PayOutOfBoundsException{
            if(salary<25000 || salary>50000){
                throw new PayOutOfBoundsException("Pay "+salary+"out
ofbounds");
            }
        }
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter basic pay :");
            int salary= sc.nextInt();

            try{
                basicPayCheck(salary);
                System.out.println("Basic pay is valid");
            }
            catch(Exception e){
                System.out.println(e);
            }
            sc.close();
        }
    }
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt10> java PayOutOfBounds
Enter basic pay :10000
PayOutOfBoundsException: Pay 10000 out of bounds
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt10> java PayOutOfBounds
Enter basic pay :40000
Basic pay is valid
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt10> java PayOutOfBounds
Enter basic pay :60000
PayOutOfBoundsException: Pay 60000 out of bounds
```

Experiment 11: Program on packages

Theory :

There are folders or directories in our computers for the classification and accessibility of various files, and in Java, we have packages for the same. In Java, Packages are similar to folders, which are mainly used to organize classes and interfaces.
Packages help us to write better and manageable code by preventing naming conflicts. Java provides some built-in packages which we can use but we can also create our own (user-defined) packages.

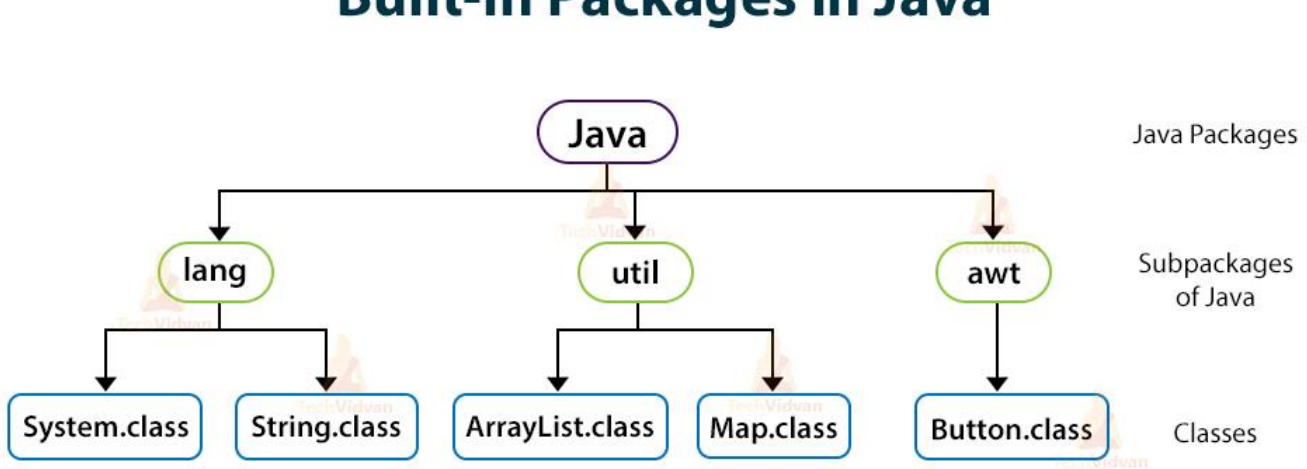
A **package** is a collection of similar types of Java entities such as classes, interfaces, subclasses, exceptions, errors, and enums. A package can also contain sub-packages.

Types of Packages in Java
They can be divided into two categories:

1. Java API packages or built-in packages and
2. User-defined packages.

1. Java API packages or built-in packages
Java provides a large number of classes grouped into different packages based on a particular functionality.

Examples:
java.lang: It contains classes for primitive types, strings, math functions, threads, and exceptions.
java.util: It contains classes such as vectors, hash tables, dates, Calendars, etc.
java.io: It has stream classes for Input/Output.
java.awt: Classes for implementing Graphical User Interface – windows, buttons, menus, etc.
java. Applet: Classes for creating and implementing applets



2. User-defined packages

As the name suggests, these packages are defined by the user. We create a directory whose name should be the same as the name of the package. Then we create a class inside the directory.

Creating a Package in Java

To create a package, we choose a package name and to include the classes, interfaces, enumerations, etc, inside the package, we write the package with its name at the top of every source file.

There can be only one package statement in each type of file. If we do not write class, interfaces, inside any package, then they will be placed in the current default package.

Accessing Packages or Classes from Another Package

If we want to access all the classes and interfaces of an existing package then we use the **import** statement. We can do it in three different ways:

- import package.*;
- import package.classname;
- fully qualified name.

1. By using ***** after the import statement, we can access all the classes of the package but not the sub-packages.

Syntax:

For importing all the classes:

import packageName.*;

2. By using a **particular class name** after the import statement, we can access that particular class package but not the sub-packages.

Syntax:
For importing a particular class:
import packageName.className;

3. Using a **Fully qualified name** means we can access the declared class of different packages without using the import statement. But you need to use a fully qualified name every time when you are accessing the class or interface which is present in a different package.
This type of technique is generally used when two packages have the same class name example class Date is present in both the packages **java.util** and **java.sql**.

A.

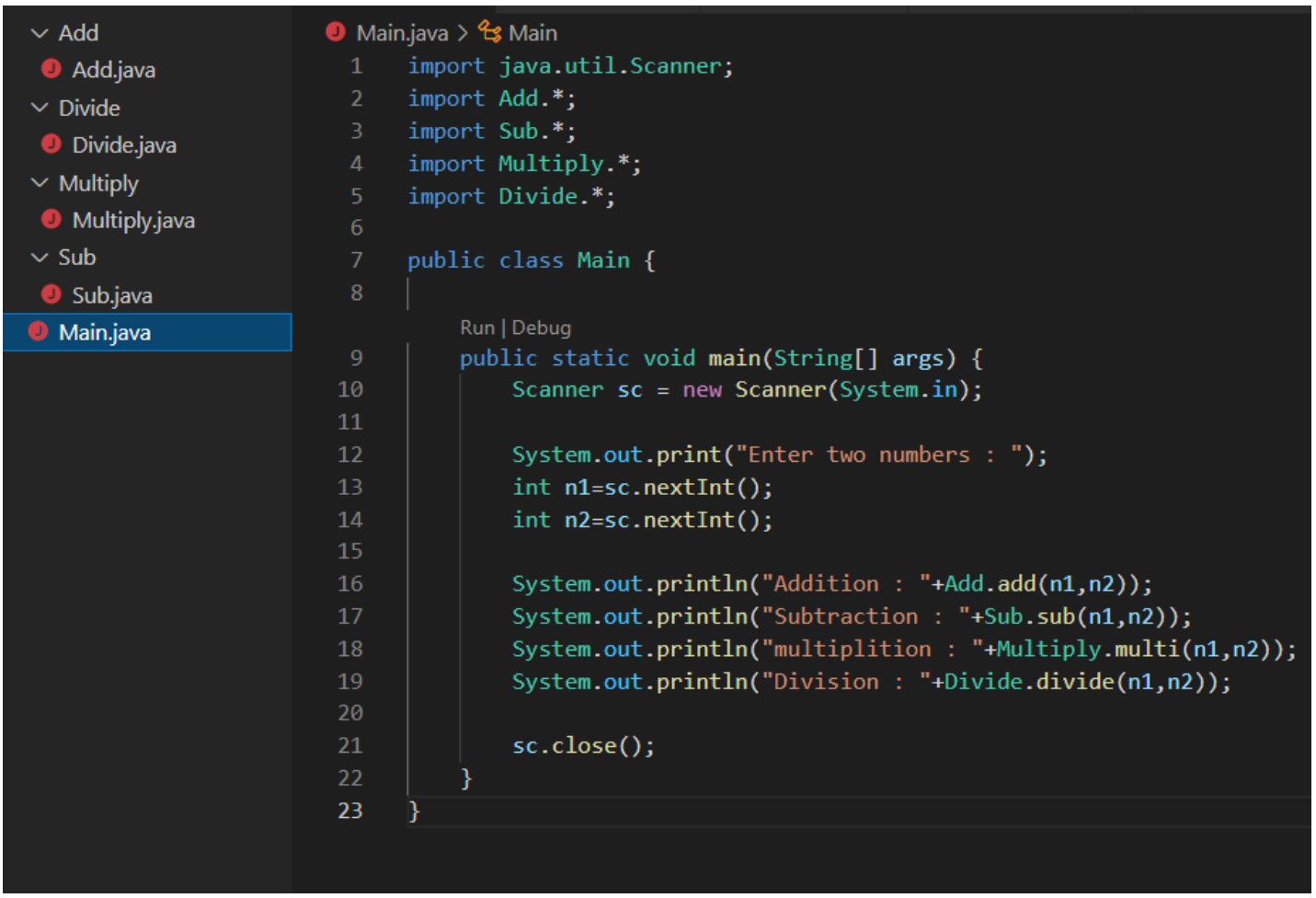
Aim : Write a program to perform four basic Arithmetic Operations using packages

Each operation should be a part of each class inside the package

Main class should import all the packages and perform operations .

Program :

Packages paths:



------ Main class ------

```java
import java.util.Scanner;
import Add.*;
import Sub.*;
import Multiply.*;
import Divide.*;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter two numbers : ");
        int n1=sc.nextInt();
        int n2=sc.nextInt();

        System.out.println("Addition : "+Add.add(n1,n2));
        System.out.println("Subtraction : "+Sub.sub(n1,n2));
        System.out.println("multiplition : "+Multiply.multi(n1,n2));
        System.out.println("Division : "+Divide.divide(n1,n2));

        sc.close();
    }
}
```

----- Code for add class in package -----

```java
package Add;

public class Add {
    public static int add(int a,int b){
        return a+b;
    }
}
```

----- Code for subtract class in package -----

```java
package Sub;

public class Sub {
    public static int sub(int a,int b){
        return b-a;
    }
}
```

----- Code for multiply class in package -----

```java
package Multiply;

public class Multiply {
    public static int multi(int a,int b){
```

```
            return a*b;
        }
    }
```

----- Code for divide class in package -----

```
    package Divide;

    public class Divide {
        public static float divide(int a,int b){
            return (float)b/a;
        }
    }
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt11>  c:; cd 'c:\Users\IsmailRatlamwala\Documents\Co
llege prog\Oops Labs\expt11'; & 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessa
ges' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceStorage\6bc1a0e23374ad28359f97af4134a140\red
hat.java\jdt_ws\expt11_ad8f9b9b\bin' 'Main'
Enter two numbers : 12 48
Addition : 60
Subtraction : 36
multiplition : 576
Division : 4.0
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt11>  c:; cd 'c:\Users\IsmailRatlamwala\Documents\Co
llege prog\Oops Labs\expt11'; & 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessa
ges' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceStorage\6bc1a0e23374ad28359f97af4134a140\red
hat.java\jdt_ws\expt11_ad8f9b9b\bin' 'Main'
Enter two numbers : 34 467
Addition : 501
Subtraction : 433
multiplition : 15878
Division : 13.735294
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt11>
```

Idris Ratlamwala
C31 - 2003145

Experiment 12: Programs on Inheritance

Theory :

Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

Important terminology:

- **Super Class:** The class whose features are inherited is known as superclass(or a base class or a parent class).
- **Sub Class:** The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

The **extends** keyword is used for inheritance.

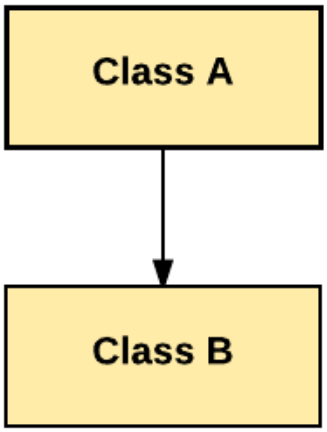
The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

Types of Inheritance :

Single Inheritance:

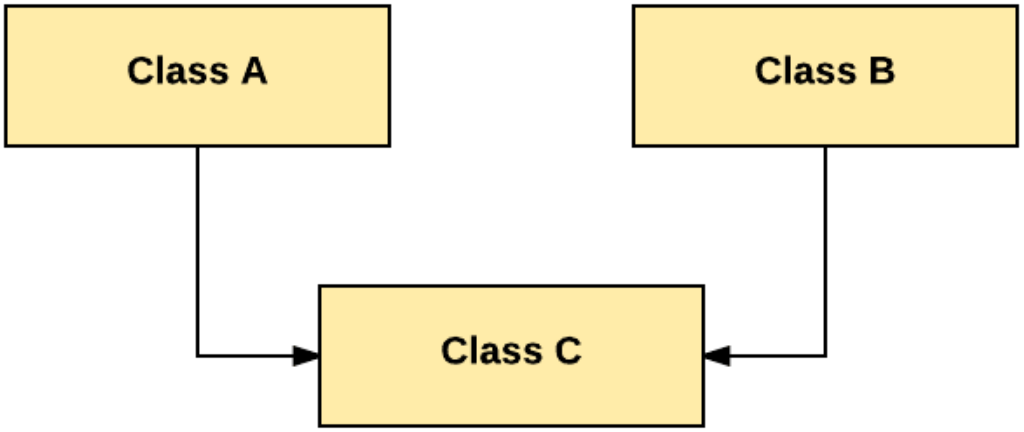
In Single Inheritance one class extends another class (one class only).



In above diagram, Class B extends only Class A. Class A is a super class and Class B is a Sub-class.

Multiple Inheritance:

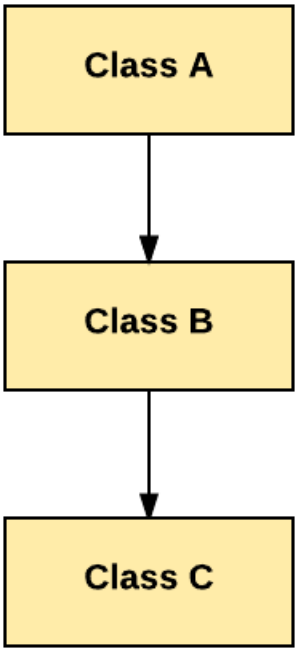
Multiple Inheritance is one of the inheritance in Java types where one class extending more than one class. Java does not support multiple inheritance.



As per above diagram, Class C extends Class A and Class B both.

Multilevel Inheritance:

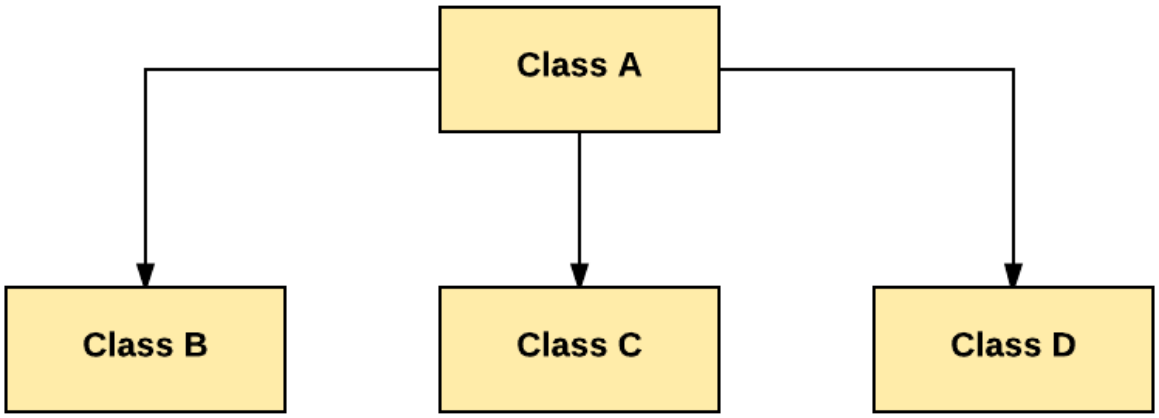
In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.



As per shown in diagram Class C is subclass of B and B is a of subclass Class A.

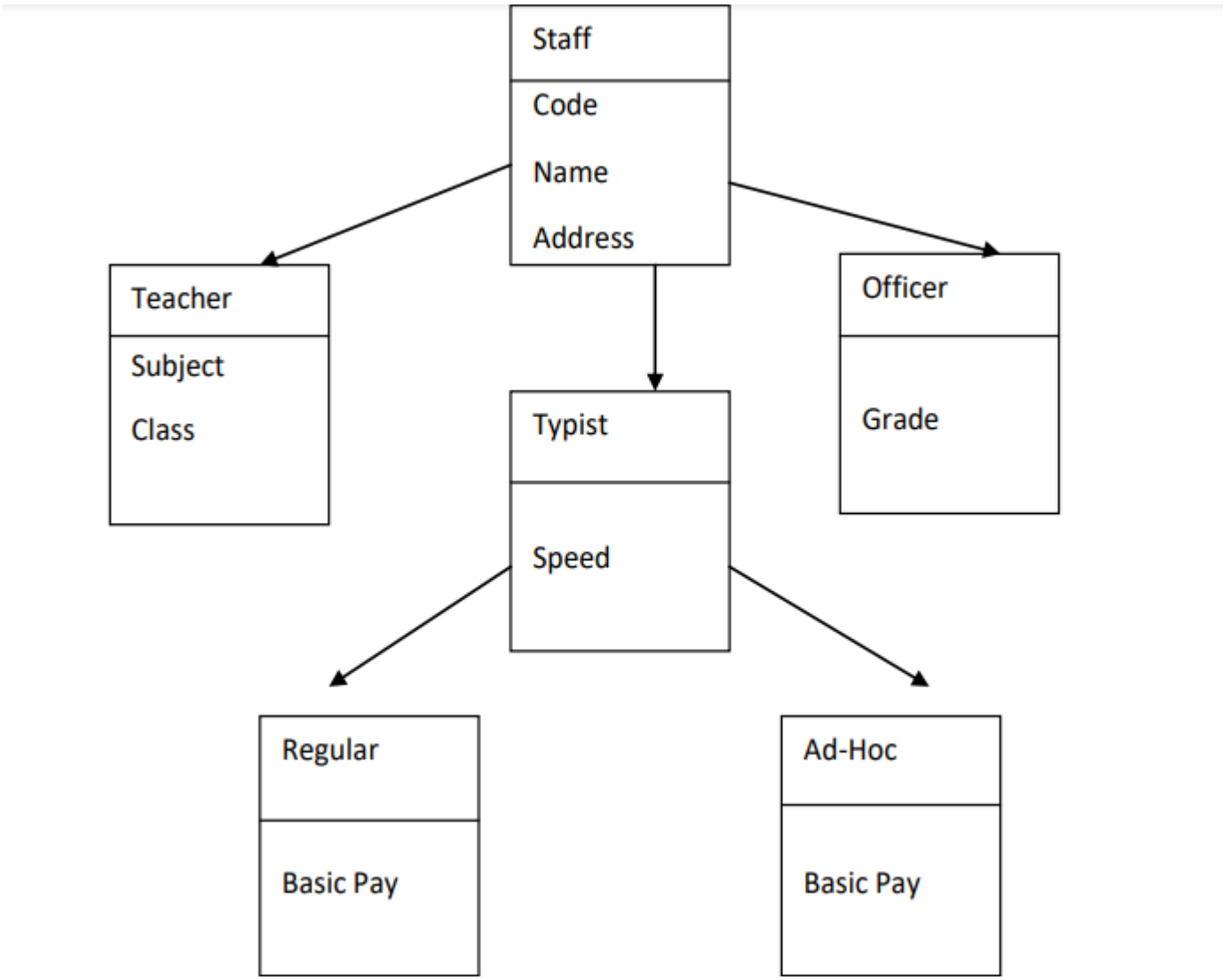
Hierarchical Inheritance:

In Hierarchical Inheritance, one class is inherited by many sub classes.



A.

Aim : Write a Java program to implement following Inheritance,



Program :

```
class Staff{
    int code;
    String name,address;

    Staff(int code,String name,String address){
        this.code=code;
        this.name=name;
        this.address=address;
    }
    void staffPrint(){
        System.out.println("Code : "+code+"\nName : "+name+"\nAddress : "+address);
    }
}
```

```
class Teacher extends Staff{
    String subject,Class;

    Teacher(int code,String name,String address,String subject,String Class){
        super(code, name, address);
        this.subject=subject;
        this.Class=Class;
    }
    void print(){
        staffPrint();
        System.out.println("Subject : "+subject+"\nClass : "+Class+"\n");
    }
}

class Officer extends Staff{
    String grade;

    Officer(int code,String name,String address,String grade){
        super(code, name, address);
        this.grade=grade;
    }
    void print(){
        staffPrint();
        System.out.println("Grade : "+grade+"\n");
    }
}

class Typist extends Staff{
    int speed;
    Typist(int code,String name,String address,int speed){
        super(code, name, address);
        this.speed=speed;
    }
    void typistPrint(){
        staffPrint();
        System.out.println("Speed : "+speed);
    }
}

class Regular extends Typist{
    int basic_pay;
    Regular(int code,String name,String address,int speed,int basic_pay){
        super(code, name, address, speed);
        this.basic_pay=basic_pay;
    }
    void print(){
        typistPrint();
        System.out.println("Basic Pay : "+basic_pay+"\n");
    }
}

class Ad_Hoc extends Typist{
    int basic_pay;
    Ad_Hoc(int code,String name,String address,int speed,int basic_pay){
```

```
        super(code, name, address, speed);
        this.basic_pay=basic_pay;
    }
    void print(){
        typistPrint();
        System.out.println("Basic Pay : "+basic_pay+"\n");
    }
}


public class CollegeInheritance {
    public static void main(String[] args) {
        int code,speed,basicPay;
        String name,address,subject,Class,grade;

        name="Juhi J";code=647452;address="Mumbai.";subject="OOPs";Class="B.E";
        Teacher t1 = new Teacher(code, name, address, subject, Class);

        name = "Rakesh M"; code = 567456; grade = "B+";
        Officer o1 = new Officer(code, name, address, grade);

        name = "Sahil K"; code = 238456; speed = 70; basicPay = 20000;
        Regular r1 = new Regular(code, name, address, speed, basicPay);

        name = "Mehir M"; code = 678457; speed = 130; basicPay = 30000;
        Ad_Hoc a1 = new Ad_Hoc(code, name, address, speed, basicPay);

        System.out.println("Teacher :");
        t1.print();
        System.out.println("Officer :");
        o1.print();
        System.out.println("Regular :");
        r1.print();
        System.out.println("Ad Hoc :");
        a1.print();
    }
}
```

Output :

```
Teacher :
Code : 647452
Name : Juhi J
Address : Mumbai.
Subject : OOPs
Class : B.E

Officer :
Code : 567456
Name : Rakesh M
Address : Mumbai.
Grade : B+

Regular :
Code : 238456
Name : Sahil K
Address : Mumbai.
Speed : 70
Basic Pay : 20000

Ad Hoc :
Code : 678457
Name : Mehir M
Address : Mumbai.
Speed : 130
Basic Pay : 30000
```

Idris Ratlamwala
C31 - 2003145

Experiment 13: Program on String

Theory :

String is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created.

Creating a String
There are two ways to create a String in Java

1. String literal
2. Using new keyword

String literal
In java, Strings can be created like this: Assigning a String literal to a String instance:
String str1 = "Welcome";
String str2 = "Welcome";

The problem with this approach: String is an object in Java. However we have not created any string object using new keyword above. The compiler does that task for us it creates a string object having the string literal (that we have provided , in this case it is “Welcome”) and assigns it to the provided string instances.

But if the object already exist in the memory it does not create a new Object rather it assigns the same old object to the new instance, that means even though we have two string instances above(str1 and str2) compiler only created on string object (having the value “Welcome”) and assigned the same to both the instances. For example there are 10 string instances that have same value, it means that in memory there is only one object having the value and all the 10 string instances would be pointing to the same object.
Using New Keyword

As we saw above that when we tried to assign the same string object to two different literals, compiler only created one object and made both of the literals to point the same object. To overcome that approach we can create strings like this:
String str1 = new String("Welcome");
String str2 = new String("Welcome");
In this case compiler would create two different object in memory having the same string.

StringBuffer class is used to create a **mutable** string object. It means, it can be changed after it is created. It represents growable and writable character sequence.

It is similar to String class in Java both are used to create string, but stringbuffer object can be changed.

So **StringBuffer** class is used when we have to make lot of modifications to our string. It is also thread safe i.e multiple threads cannot access it simultaneously. StringBuffer defines 4 constructors.

1. **StringBuffer**(): It creates an empty string buffer and reserves space for 16 characters.
2. **StringBuffer**(int size): It creates an empty string and takes an integer argument to set capacity of the buffer.
3. **StringBuffer**(String str): It creates a stringbuffer object from the specified string.
4. **StringBuffer**(charSequence []ch): It creates a stringbuffer object from the charsequence array.

A.

Aim : WAP to check if a string is a palindrome .

Program :

```
import java.util.Scanner;

public class palindrome{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a string : ");
        String inp = sc.next();
        String reversedString = reverseString(inp);

        if (inp.equals(reversedString))  System.out.println(inp + " is a palindrome");
        else  System.out.println(inp + " is not a palindrome");
    }

    private static String reverseString(String inp) {
        StringBuffer buffer = new StringBuffer();
        buffer.append(inp);
        return buffer.reverse().toString();
    }
}
```

Output :

```
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>  & 'C:\Program Files\Java\jdk-16.0.2\bin\java.e
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceSto
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin' 'palindrome'
Enter a string : madam
madam is a palindrome
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>  & 'C:\Program Files\Java\jdk-16.0.2\bin\java.e
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala\AppData\Roaming\Code\User\workspaceSto
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin' 'palindrome'
Enter a string : hello
hello is not a palindrome
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>
```

B.

Aim : WAP to accept a string from user and display the number of uppercase, lowercase, special characters, blank spaces & digits present in the accepted string.

Program :

```
import java.io.*;
import java.util.Scanner;

class Count
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string :");
        String str = sc.nextLine();
        int upper = 0, lower = 0, number = 0, special = 0, blank = 0;

        for(int i = 0; i < str.length(); i++)
        {
            char ch = str.charAt(i);
            if (ch >= 'A' && ch <= 'Z')
                upper++;
            else if (ch >= 'a' && ch <= 'z')
                lower++;
            else if (ch==' ')
                blank++;
            else if (ch >= '0' && ch <= '9')
                number++;
            else
                special++;
        }

        System.out.println("Lower case letters : " + lower);
        System.out.println("Upper case letters : " + upper);
        System.out.println("Special characters : " + special);
        System.out.println("Blank characters : " + blank);
        System.out.println("Number : " + number);
    }
}
```

Output :

```
xe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\IsmailRatlamwala
rage\7ab5dfaeabdccc59425a28238aed5cf8\redhat.java\jdt_ws\expt13_ad8f9b9d\bin'
Enter a string :
Hello World !
Lower case letters : 8
Upper case letters : 2
Special characters : 1
Blank characters : 2
Number : 0
PS C:\Users\IsmailRatlamwala\Documents\College prog\Oops Labs\expt13>
```

Idris Ratlamwala
C31 - 2003145

Experiment 14: Program to create a GUI based Application

Theory :

Swing in java is part of Java foundation class which is lightweight and platform independent. It is used for creating window based applications. It includes components like button, scroll bar, text field etc. Putting together all these components makes a graphical user interface. In this article, we will go through the concepts involved in the process of building applications using swing in Java.

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes). It is build on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components. It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch.

Container Class
Any **class** which has other components in it is called as a container class. For building GUI applications at least one container class is necessary.

Following are the three types of container classes:

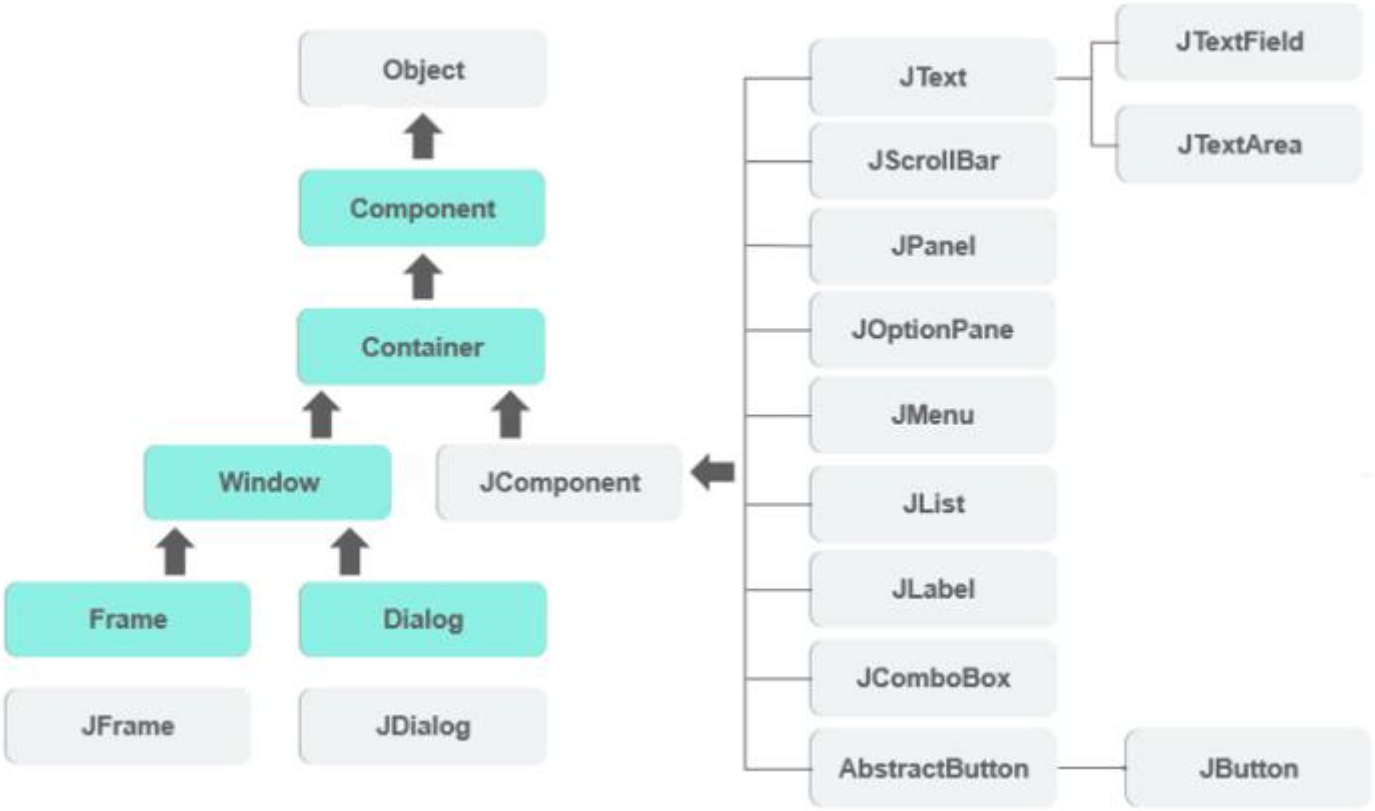
1. Panel – It is used to organize components on to a window
2. Frame – A fully functioning window with icons and titles
3. Dialog – It is like a pop up window but not fully functional like the frame

Difference Between AWT and Swing

AWT	SWING

• Platform Dependent	• Platform Independent
• Does not follow MVC	• Follows MVC
• Lesser Components	• More powerful components
• Does not support pluggable look and feel	• Supports pluggable look and feel
• Heavyweight	• Lightweight

Java Swing Class Hierarchy



All the components in swing like JButton, JComboBox, JList, JLabel are inherited from the JComponent class which can be added to the container classes. Containers are the windows like frame and dialog boxes. Basic swing components are the building blocks of any gui application. Methods like setLayout override the default layout in each container. Containers like JFrame and JDialog can only add a component to itself. Following are a few components with examples to understand how we can use them.

JButton Class

It is used to create a labelled button. Using the ActionListener it will result in some action when the button is pushed. It inherits the AbstractButton class and is platform independent.

JTextField Class
It inherits the JTextComponent class and it is used to allow editing of single line text.

JPanel Class
It inherits the JComponent class and provides space for an application which can attach any other component.

Java ActionListener Interface

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed(). actionPerformed() method

The actionPerformed() method
is invoked automatically whenever you click on the registered component.

```
public abstract void actionPerformed(ActionEvent e);
```

The common approach is to implement the ActionListener. We need to follow 3 steps:

1) Implement the ActionListener interface in the class:

```
public class ActionListenerExample Implements ActionListener
```

2) Register the component with the Listener:

```
component.addActionListener(instanceOfListenerclass);
```

3) Override the actionPerformed() method:

```
1. public void actionPerformed(ActionEvent e){
2.         //Write the code here
3. }
```

A.

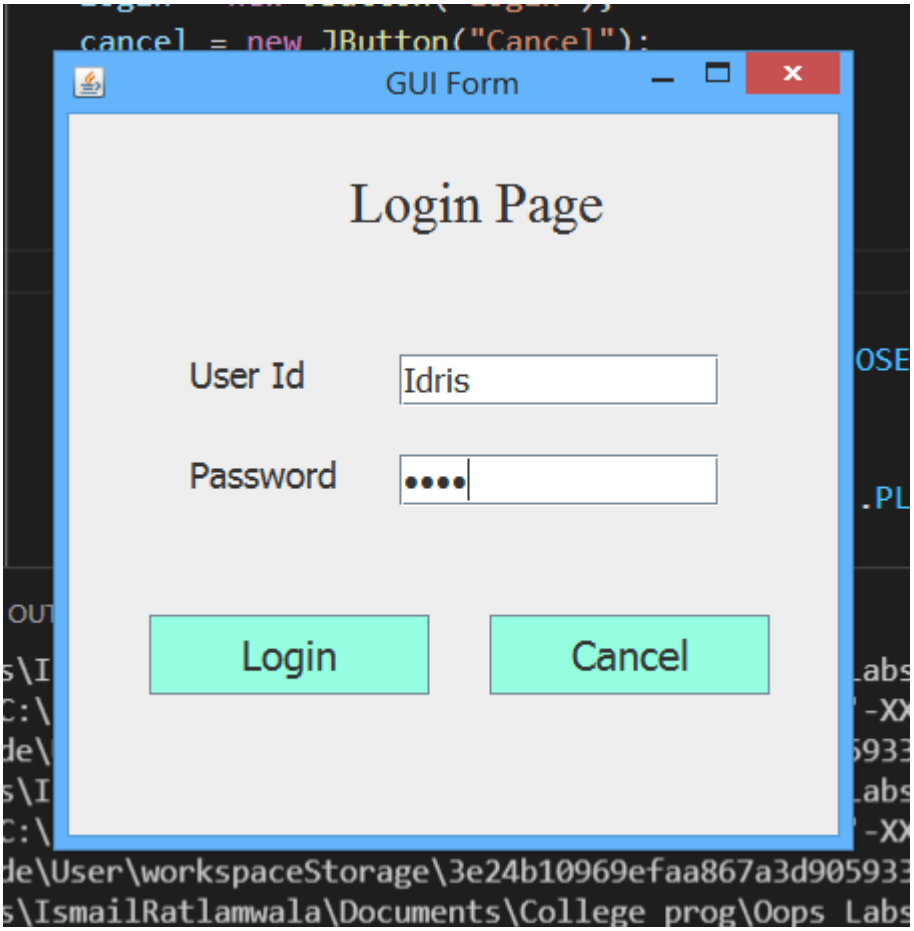
Aim : Create a registration form containing required fields .
The form should have all the studied components
The form should have minimum two buttons “Submit” and “Cancel”….giving
appropriate messages at corresponding click.

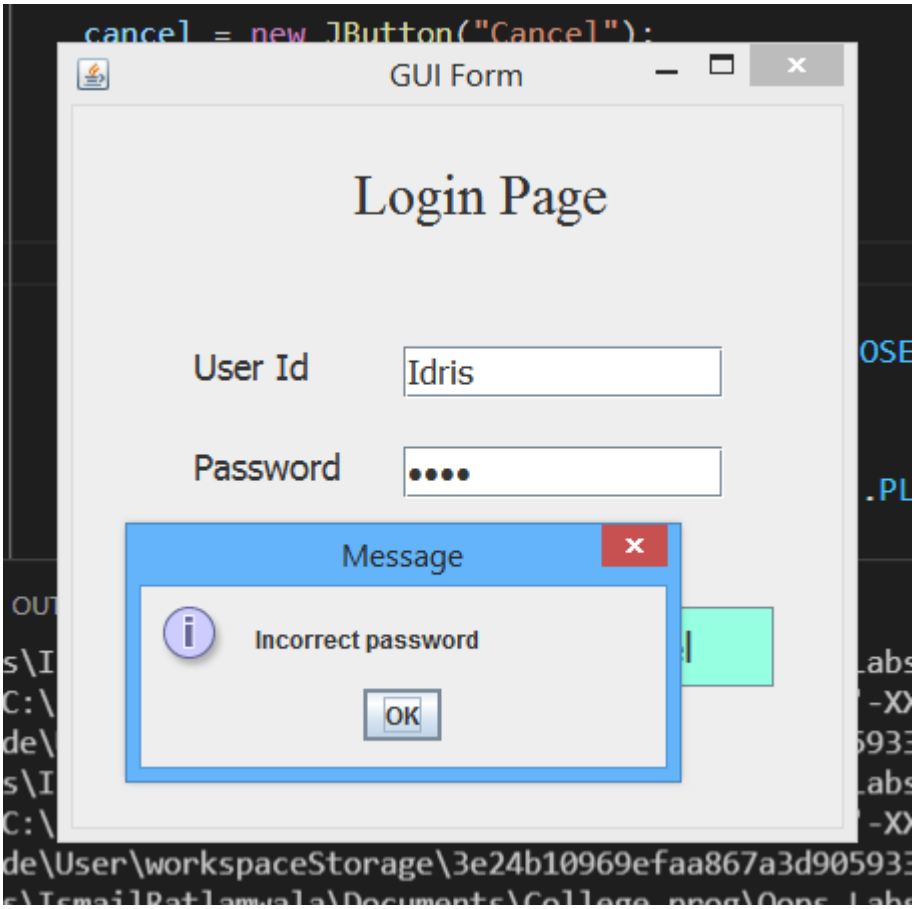
Program :

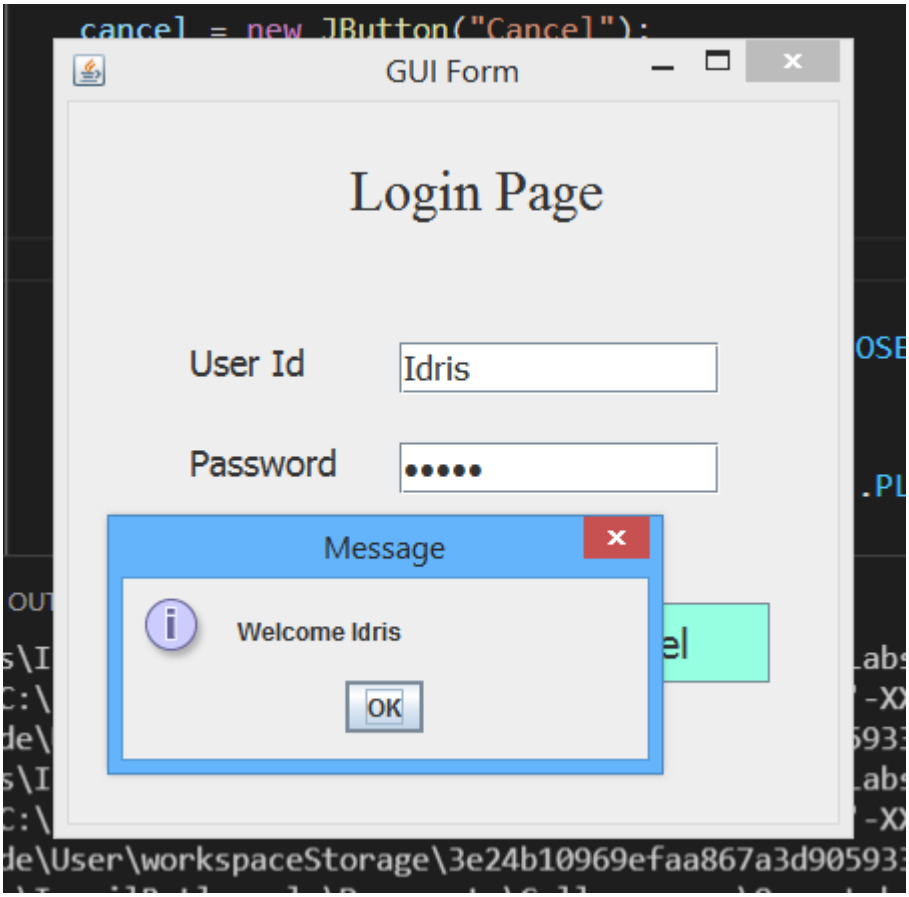
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Login_page {
    public static void main(String[] args) {
        new Windows();
    }
}

class Windows extends JFrame implements ActionListener {
    JTextField password, id;
    JLabel l, l1, l2;
    JButton login,cancel;

    public Windows() {
        password = new JPasswordField(15);
        id = new JTextField(15);
        login = new JButton("Login");
        cancel = new JButton("Cancel");
        l = new JLabel("Login Page");
        l1 = new JLabel("User Id");
        l2 = new JLabel("Password");

        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("GUI Form");
        setBounds(200,200,400, 400);
        l.setFont(new Font("Times New Roman", Font.PLAIN, 29));
        l.setBounds(140, 30, 190, 30);

        l1.setFont(new Font("Tahoma", Font.PLAIN, 18));
        l1.setBounds(60, 120, 80, 20);

        l2.setFont(new Font("Tahoma", Font.PLAIN, 18));
```

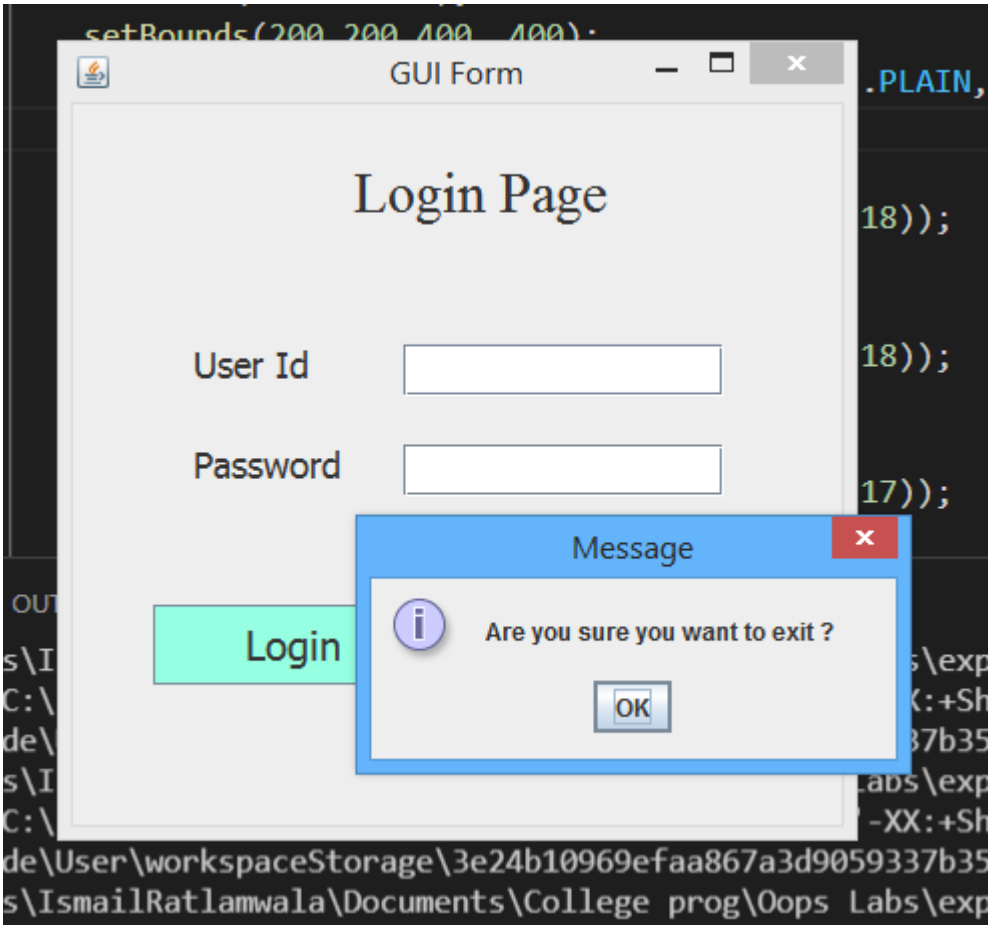
```
        l2.setBounds(60, 170, 80, 20);

        id.setFont(new Font("Tahoma", Font.PLAIN, 17));
        id.setBounds(165, 120, 160, 26);

        password.setFont(new Font("Tahoma", Font.PLAIN, 17));
        password.setBounds(165, 170, 160, 26);

        login.setFont(new Font("Tahoma", Font.PLAIN, 20));
        login.setBackground(new Color(150, 255, 225));
        login.setBounds(40, 250, 140, 40);
        login.addActionListener(this);

        cancel.setFont(new Font("Tahoma", Font.PLAIN, 20));
        cancel.setBackground(new Color(150, 255, 225));
        cancel.setBounds(210, 250, 140, 40);
        cancel.addActionListener(this);

        add(l);
        add(l1);
        add(id);
        add(l2);
        add(password);
        add(login);
        add(cancel);
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==login){
            String pass = password.getText();
            String name = id.getText();
            if (pass.equals("hello")) {
                JOptionPane.showMessageDialog(login, "Welcome "+name);
                dispose();
            } else {
                JOptionPane.showMessageDialog(login, "Incorrect password");
            }
        }
        else if(e.getSource()==cancel){
            JOptionPane.showMessageDialog(cancel, "Are you sure you want to exit ?");
            dispose();
        }
    }
}
```

Output :


GUI Form
Login Page
User Id
Idris
Password
Login
Cancel


GUI Form
Login Page
User Id
Idris
Password
Message
Incorrect password
OK


GUI Form
Login Page
User Id
Idris
Password
Message
Welcome Idris
OK


GUI Form
Login Page
User Id
Password
Login
Message
Are you sure you want to exit ?
OK

Idris Ratlamwala
2003145 - C31

PAGE NO.
DATE / /

JAVA ASSIGNMENT. 1

Write short note on:

1. Features of java:

1) Simple: Java is easy to learn and its syntax is quite simple, clean and easy to understand. The ambigous concepts of c++ are either left out in java or they have been re-implemented in cleaner way.

2) Object Oriented: In java everything is an object which has some data and behaviours. Java can be easily extended as its based on Object Model. following are basic concepts of OOPs.
   i) Object
   ii) Class
   iii) Inheritance
   iv) Polymorphism
   v) Abstraction
   vi) Encapsulation

3) Robust: Java makes an effort to eliminate error prone codes by emphesising mainly on compile time error checking and runtime checking. But the main areas which java improved were Memory management and mishandeled exceptions.

4) Platform independent: Unlike other programing languages such as C, C++, which are compiled into platform specific machines, Java is guaranteed to be write-once & run anywhere language.

● On compilation, java program is compiled into bytecode. This bytecode is platform independent. and can be run on any device.

PAGE No.
DATE / /

5) Secure : When it comes to security, java is always the first choice. With java secure features, it enable us to develop virus free, temper free system. It always runs in java runtime environment with almost null interactions with system OS, hence more secure.

6) Multi threading : Java multithreading makes it possible to write program that can do many task simultaneously. Benifit of multithreading is that it utilizes same memory / other resources to execute multiple threads at same time.

7) Potable : Java byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example :- size of primitive data types.

8) Distributed : Java is also distributed language. Programs can be designed to run on computer networks. Java has a special class library for communicating ~~protocol~~ using TCP / IP protocols.

PAGE NO.
DATE / /

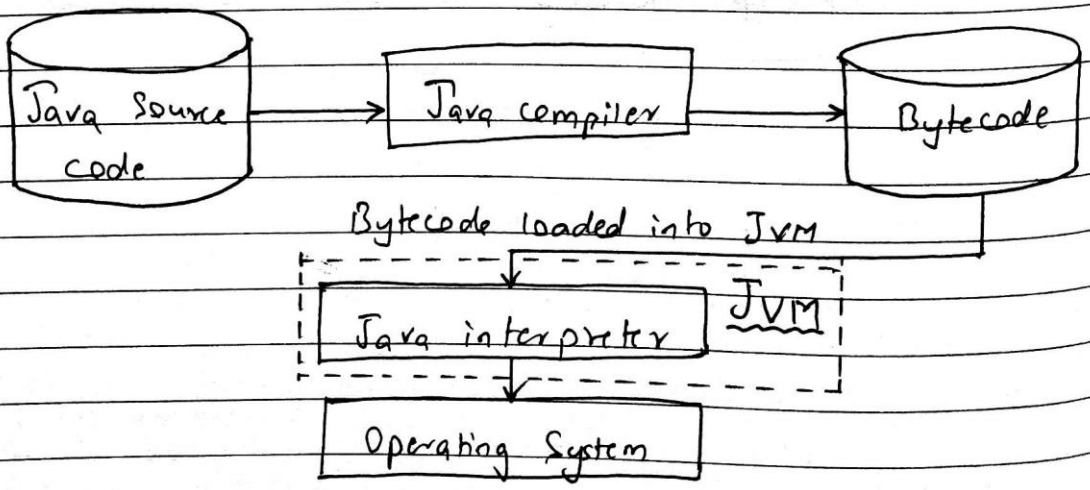
2. JVM :

The java virtual machine is called as JVM, is an abstract computing machine or virtual machine ~~that~~ interface that drives the java code.

- Mostly in other compiling languages, compiler produces a code for a particular system, but java ~~produces~~ compiler produces byte code for a java virtual machine.
- When we compile a Java program, then a bytecode is generated. Bytecode is the source code that can be used to run on any platform.
- Bytecode is an intermediate language between java source and the host systems.
- It is the medium which compiles javacode to bytecode which gets interpreted on different machines and hence makes it platform/OS independent.

JVM's work can be explained in the following manner:

- Reading the bytecode.
- Verifing bytecode.
- Linking the code with the library.



PAGE No.
DATE / /

Platform independent :

Java is called platform independent because of JVM As different computers with different OS have their JVM, when we submit a .class file to any OS, JVM interprets the bytecode into machine level language

- JVM is the main component of Java architecture, and it is the part of the JRE
- A program of JVM is written in 'C' and JVM is OS dependent.
- JVM is responsible for allocating the necessary memory needed by the Java program.
- JVM is also responsible for deallocating the memory.

3) Wrapper class :

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

- **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
- **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
- **Synchronization:** Java synchronization works with objects in Multithreading.

- **java.util package:** The java.util package provides the utility classes to deal with objects.
- **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

The eight classes of the *java.lang* package are known as wrapper classes in Java. The list of eight wrapper classes are given below:

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Autoboxing

The automatic conversion of primitive data types into its wrapper class objects is known as autoboxing.

```
class Autoboxing {
    public static void main( String args[] ) {
     int a = 15; // Primitive data type
     Integer I = a; // Autoboxing will occur internally.
```

```
    }
}
```

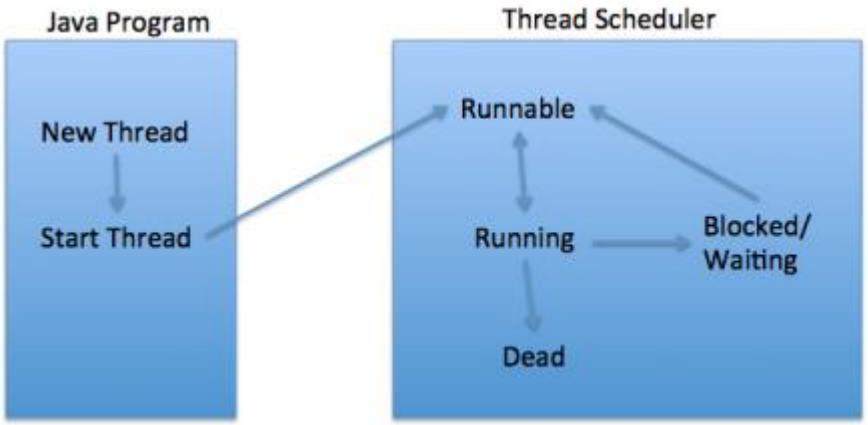
Unboxing

The automatic conversion of wrapper class objects into its primitive data types is known as unboxing.

```
class Autoboxing {
    public static void main( String args[] ) {
      Integer a = new Integer(15); // Wrapper class object
      int I = a;// Unboxing will occur internally.
    }
}
```

4) Life cycle of a thread :

Below diagram shows different states of thread life cycle in java. We can create a thread in java and start it but how the thread states change from Runnable to Running to Blocked depends on the OS implementation of thread scheduler and java doesn't have full control on that.



New

When we create a new Thread object using *new* operator, thread state is New Thread. At this point, thread is not alive and it's a state internal to Java programming.

Runnable

When we call start() function on Thread object, it's state is changed to Runnable. The control is given to Thread scheduler to finish it's execution. Whether to run this thread instantly or keep it in runnable thread pool before running, depends on the OS implementation of thread scheduler.

Running

When thread is executing, it's state is changed to Running. Thread scheduler picks one of the thread from the runnable thread pool and change it's state to Running. Then CPU starts executing this thread. A thread can change state to Runnable, Dead or Blocked from running state depends on time slicing, thread completion of run() method or waiting for some resources.

Blocked/Waiting

A thread can be waiting for other thread to finish using thread join or it can be waiting for some resources to available. For example producer consumer problem or waiter notifier implementation or IO resources, then it's state is changed to Waiting. Once the thread wait state is over, it's state is changed to Runnable and it's moved back to runnable thread pool.

Dead

Once the thread finished executing, it's state is changed to Dead and it's considered to be not alive.

Above are the different **states of thread**. It's good to know them and how thread changes it's state. That's all for thread life cycle in java.

Idris Ratlamwala
C31 - 2003145

Java Assingment 2

a. Thread synchronization:

When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally they can produce unforeseen result due to concurrency issues. For example, if multiple threads try to write within a same file then they may corrupt the data because one of the threads can override data or while one thread is opening the same file at the same time another thread might be closing the same file.

As Java is a multi_threaded language, thread synchronization has a lot of importance in Java as multiple threads execute in parallel in an application.

We use keywords “synchronized” and “volatile” to achieve Synchronization in Java

We need synchronization when the shared object or resource is mutable. If the resource is immutable, then the threads will only read the resource either concurrently or individually.

In this case, we do not need to synchronize the resource. In this case, JVM ensures that Java synchronized code is executed by one thread at a time.

Most of the time, concurrent access to shared resources in Java may introduce errors like “Memory inconsistency” and “thread interference”. To avoid these errors we need to go for synchronization of shared resources so that the access to these resources is mutually exclusive.

We use a concept called Monitors to implement synchronization. A monitor can be accessed by only one thread at a time. When a thread gets the lock, then, we can say the thread has entered the monitor.

When a monitor is being accessed by a particular thread, the monitor is locked and all the other threads trying to enter the monitor are suspended until the accessing thread finishes and releases the lock.

Going forward, we will discuss synchronization in Java in detail in this tutorial. Now, let us discuss some basic concepts related to synchronization in Java.

Race Condition In Java
In a multithreaded environment, when more than one thread tries to access a shared resource for writing simultaneously, then multiple threads race each other to finish accessing the resource. This gives rise to 'race condition'.

One thing to consider is that there is no problem if multiple threads are trying to access a shared resource only for reading. The problem arises when multiple threads access the same resource at the same time.

Race conditions occur due to a lack of proper synchronization of threads in the program. When we properly synchronize the threads such that at a time only one thread will access the resource, and the race condition ceases to exist.

b. Abstract classes :

A class that is declared using "abstract" keyword is known as abstract class. It can have abstract methods(methods without body) as well as concrete methods (regular methods with body). A normal class(non-abstract class) cannot have abstract methods.

Lets say we have a class Animal that has a method sound() and the subclasses(see inheritance) of it like Dog, Lion, Horse, Cat etc. Since the animal sound differs from one animal to another, there is no point to implement this method in parent class. This is because every child class must override this method to give its own implementation details, like Lion class will say "Roar" in this method and Dog class will say "Woof".

So when we know that all the animal child classes will and should override this method, then there is no point to implement this method in parent class. Thus, making this method abstract would be the good choice as by making this method abstract we force all the sub classes to implement this method(

otherwise you will get compilation error), also we need not to give any implementation to this method in parent class.

Since the Animal class has an abstract method, you must need to declare this class abstract.

Now each animal must have a sound, by making this method abstract we made it compulsory to the child class to give implementation details to this method. This way we ensures that every animal has a sound.

Rules

1: As there are cases when it is difficult or often unnecessary to implement all the methods in parent class. In these cases, we can declare the parent class as abstract, which makes it a special class which is not complete on its own.
A class derived from the abstract class must implement all those methods that are declared as abstract in the parent class.

2: Abstract class cannot be instantiated which means you cannot create the object of it. To use this class, you need to create another class that extends this this class and provides the implementation of abstract methods, then you can use the object of that child class to call non-abstract methods of parent class as well as implemented methods(those that were abstract in parent but implemented in child class).

3: If a child does not implement all the abstract methods of abstract parent class, then the child class must need to be declared abstract as well.

c. JDBC Drivers and Architecture

The term JDBC stands for Java Database Connectivity. JDBC is a specification from Sun microsystems. JDBC is an API(Application programming interface) in Java that helps users to interact or communicate with various databases.

The classes and interfaces of JDBC API allow the application to send the request to the specified database.

Using JDBC, we can write programs required to access databases. JDBC and the database driver are capable of accessing databases and spreadsheets. JDBC API is also helpful in accessing the enterprise data stored in a relational database(RDB).

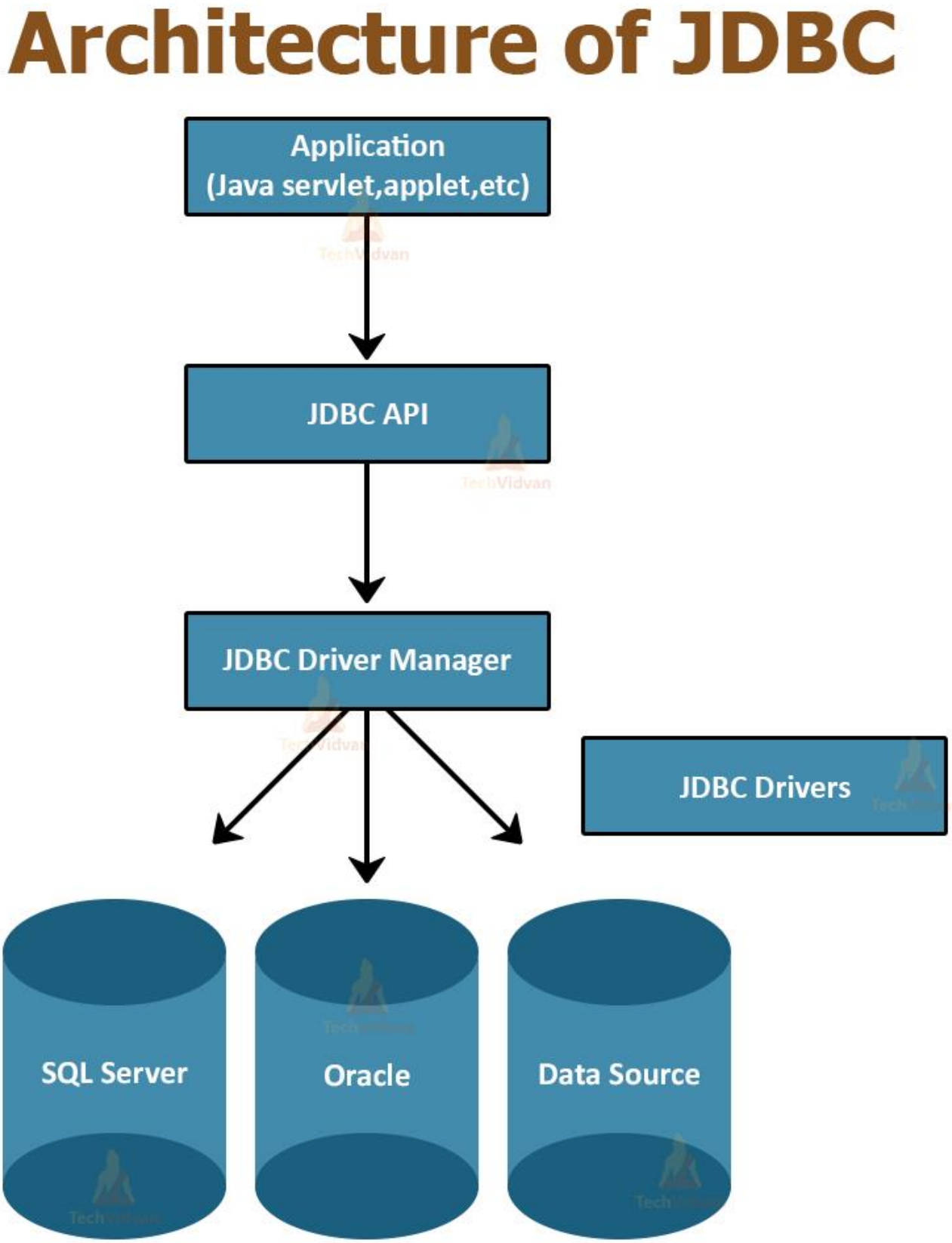
There are some enterprise applications created using the JAVA EE(Enterprise Edition) technology. These applications need to interact with databases to store application-specific information.

Interacting with the database requires efficient database connectivity, which we can achieve using ODBC(Open database connectivity) driver. We can use this ODBC Driver with the JDBC to interact or communicate with various kinds of databases, like Oracle, MS Access, Mysql, and SQL, etc.

Applications of JDBC
JDBC is fundamentally a specification that provides a complete set of interfaces. These interfaces allow for portable access to an underlying database.

We can use Java to write different types of executables, such as:

Java Applications
Java Applets
Enterprise JavaBeans (EJBs)
Java Servlets
Java ServerPages (JSPs)
All these different executables can use a JDBC driver to access a database, and take advantage of the stored data. JDBC provides similar capabilities as ODBC by allowing Java programs to contain database-independent code.
Architecture of JDBC
The following figure shows the JDBC architecture:



Description of the Architecture:

1. Application: Application in JDBC is a Java applet or a Servlet that communicates with a data source.

2. JDBC API: JDBC API provides classes, methods, and interfaces that allow Java programs to execute SQL statements and retrieve results from the database. Some important classes and interfaces defined in JDBC API are as follows:

- DriverManager
- Driver
- Connection
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- SQL data

3. Driver Manager: The Driver Manager plays an important role in the JDBC architecture. The Driver manager uses some database-specific drivers that effectively connect enterprise applications to databases.

4. JDBC drivers: JDBC drivers help us to communicate with a data source through JDBC. We need a JDBC driver that can intelligently interact with the respective data source.

Types of JDBC Architecture

There are two types of processing models in JDBC architecture: two-tier and three-tier. These models help us to access a database. They are:

1. Two-tier model

In this model, a Java application directly communicates with the data source. JDBC driver provides communication between the application and the data source. When a user sends a query to the data source, the answers to those queries are given to the user in the form of results.

We can locate the data source on a different machine on a network to which a user is connected. This is called a client/server configuration, in which the user machine acts as a client, and the machine with the data source acts as a server.

2. Three-tier model

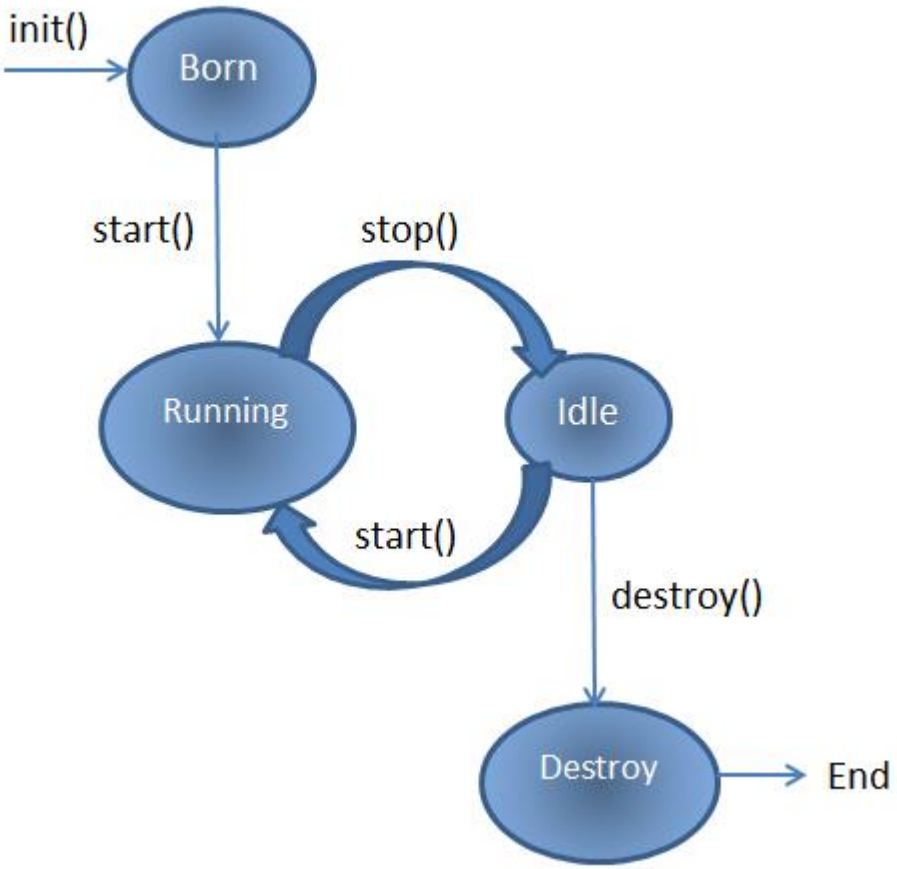
In the three-tier model, the query of the user queries goes to the middle-tier services. From the middle-tier service, the commands again reach the data source. The results of the query go back to the middle tier.

d. Life Cycle of an applet

Applets are small java programs that are primarily used in internet computing. It can be transported over the internet from one computer to another and run using the Applet Viewer or any web browser that supports Java.

An applet is like an application program, it can do many things for us. It can perform **arithmetic operations, display graphics, create animation & games, play sounds** and accept user input, etc. Every Java applet inherits a set of default behaviours from the Applet class. As a result, when an applet is loaded, it undergoes a series of changes in its states.

Applet Life Cycle:
Applet Life Cycle is defined as how the applet created, started, stopped and destroyed during the entire execution of the application. The methods to execute in the Applet Life Cycle are **init()**, **start()**, **stop()**, **destroy()**. The Applet Life Cycle Diagram is given as below:



- Born or Initialization State
- Running State

- Idle State
- Dead State
-

1. Born or Initialization State: Applets enters the initialization state when it is first loaded. It is achieved by **init()** method of Applet class. Then applet is born.

```
public void init()
{
.....
.....
(Action)
}
```

2. Running State: Applet enters the running state when the system calls the **start()** method of Applet class. This occurs automatically after the applet is initialized.

```
public void start()
{
.....
.....
(Action)
}
```

3. Idle State: An applet becomes idle when it is stopped from running. Stopping occurs automatically when we leave the page containing the currently running applet. This is achieved by calling the **stop()** method explicitly.

```
public void stop()
{
.....
.....
(Action)
}
```

4. Dead State: An applet is said to be dead when it is removed from memory. This occurs automatically by invoking the **destroy()** method when we want to quit the browser.

```
public void destroy()
{
.....
.....
```

(Action)
}

GROUP MEMBERS
BATCH C31

LAVIN RUPANI -2003147
IDRIS RATLAMWALA -2003145
PRIYANSH SALIAN -2003148

Mini Project Report (JAVA OOPM)

PIL Canteen Management System

1. Introduction

This canteen management system discards the old billing system of writing everything down and taking orders to an advanced application.Where an authenticated user can order his/her meal and can have his/her account to debt balance on thus, can pay the total bill after the specified time, without worrying about the book account.

2. Motivation

Having to stand in long queues, sometimes errors in account balance lead us to think of making a software which is capable of ordering items and can hold all the data of orders and transactions.

3. CODE-

Main.java

```
import java.awt.EventQueue;

public class Main {
    public static void main(String[] args) {
         EventQueue.invokeLater(new Runnable() {
             public void run() {
                  new Signin();
             }
         });
     }
}
```

Signin.java

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

public class Signin extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField username;
    private JPasswordField passwordField;
    private JButton loginButton,newUserButton;

    public Signin() {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setIconImage(Toolkit.getDefaultToolkit().getImage("img/bg.png"));
        setTitle("PIL CANTEEN");
        setBounds(200, 100, 400, 500);
        setResizable(false);
        setVisible(true);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
```

```
        contentPane.setLayout(null);
        getContentPane().setBackground(new Color(150, 255, 225));


        JLabel welcome = new JLabel("Welcome to");
        welcome.setFont(new Font("Times New Roman", Font.PLAIN, 23));
        welcome.setBounds(129, 10, 300, 60);
        contentPane.add(welcome);

        JLabel welcome1 = new JLabel(" PIL CANTEEN");
        welcome1.setFont(new Font("Felix Titling", Font.PLAIN, 27));
        welcome1.setBounds(92, 42, 300, 60);
        contentPane.add(welcome1);

        JLabel lblsignin = new JLabel("Login");
        lblsignin.setFont(new Font("Berlin Sans FB", Font.PLAIN, 25));
        lblsignin.setBounds(154, 102, 200, 50);
        contentPane.add(lblsignin);


        JLabel lblUsername = new JLabel("Username");
        lblUsername.setFont(new Font("Tahoma", Font.PLAIN, 18));
        lblUsername.setBounds(20, 180, 90, 18);
        contentPane.add(lblUsername);

        JLabel lblPassword = new JLabel("Password");
        lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 18));
        lblPassword.setBounds(20, 240, 90, 18);
        contentPane.add(lblPassword);

        username = new JTextField();
        username.setFont(new Font("Tahoma", Font.PLAIN, 18));
        username.setBounds(110, 175, 228, 30);
        username.setBorder(new EmptyBorder(0, 5, 0, 0));
        contentPane.add(username);
        username.setColumns(10);

        passwordField = new JPasswordField();
        passwordField.setFont(new Font("Tahoma", Font.PLAIN, 18));
        passwordField.setBounds(110, 235, 228, 30);
        passwordField.setBorder(new EmptyBorder(0, 5, 0, 0));
        contentPane.add(passwordField);

        loginButton = new JButton("Sign in");
        loginButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String userName = username.getText();
                String password = String.valueOf(passwordField.getPassword());
                String query = "select password_,first_name from account where user_name =
\""+userName+"\"";
                try {
                    Class.forName("com.mysql.cj.jdbc.Driver");
```

```
                    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/canteen_db", "root", "1234");
                    Statement sta = connection.createStatement();

                    ResultSet rs = sta.executeQuery(query);
                    rs.next();
                    String db_password = rs.getString("password_");
                    //System.out.println(db_password+" entered :"+password);

                    if (db_password.equals(password)) {
                        JOptionPane.showMessageDialog(loginButton,"Welcome, "+
rs.getString("first_name"));
                        dispose();
                        Order o =new Order(userName); //----------------------------------
---------------
                        o.setVisible();
                    } else {
                        JOptionPane.showMessageDialog(loginButton, "Invalid Password");
                    }
                    connection.close();
                } catch (Exception exception) {
                    //exception.printStackTrace();
                    JOptionPane.showMessageDialog(loginButton, "Invalid User");
                    System.out.println(exception);
                }
            }
        });
        loginButton.setFont(new Font("Tahoma", Font.PLAIN, 18));
        loginButton.setBounds(56, 310, 270, 38);
        loginButton.setFocusPainted(false);
        contentPane.add(loginButton);

        newUserButton = new JButton("New user");
        newUserButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                dispose();
                new UserRegistration();
            }
        });
        newUserButton.setFont(new Font("Tahoma", Font.PLAIN, 18));
        newUserButton.setBounds(56, 370, 270, 38);
        newUserButton.setFocusPainted(false);
        contentPane.add(newUserButton);
    }
}
```

UserRegistration.java

```
import java.awt.Color;
import java.awt.Font;
```

```java
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

public class Signin extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField username;
    private JPasswordField passwordField;
    private JButton loginButton,newUserButton;

    public Signin() {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setIconImage(Toolkit.getDefaultToolkit().getImage("img/bg.png"));
        setTitle("PIL CANTEEN");
        setBounds(200, 100, 400, 500);
        setResizable(false);
        setVisible(true);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        getContentPane().setBackground(new Color(150, 255, 225));


        JLabel welcome = new JLabel("Welcome to");
        welcome.setFont(new Font("Times New Roman", Font.PLAIN, 23));
        welcome.setBounds(129, 10, 300, 60);
        contentPane.add(welcome);

        JLabel welcome1 = new JLabel(" PIL CANTEEN");
        welcome1.setFont(new Font("Felix Titling", Font.PLAIN, 27));
        welcome1.setBounds(92, 42, 300, 60);
        contentPane.add(welcome1);

        JLabel lblsignin = new JLabel("Login");
        lblsignin.setFont(new Font("Berlin Sans FB", Font.PLAIN, 25));
        lblsignin.setBounds(154, 102, 200, 50);
        contentPane.add(lblsignin);


        JLabel lblUsername = new JLabel("Username");
```

```
        lblUsername.setFont(new Font("Tahoma", Font.PLAIN, 18));
        lblUsername.setBounds(20, 180, 90, 18);
        contentPane.add(lblUsername);

        JLabel lblPassword = new JLabel("Password");
        lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 18));
        lblPassword.setBounds(20, 240, 90, 18);
        contentPane.add(lblPassword);

        username = new JTextField();
        username.setFont(new Font("Tahoma", Font.PLAIN, 18));
        username.setBounds(110, 175, 228, 30);
        username.setBorder(new EmptyBorder(0, 5, 0, 0));
        contentPane.add(username);
        username.setColumns(10);

        passwordField = new JPasswordField();
        passwordField.setFont(new Font("Tahoma", Font.PLAIN, 18));
        passwordField.setBounds(110, 235, 228, 30);
        passwordField.setBorder(new EmptyBorder(0, 5, 0, 0));
        contentPane.add(passwordField);

        loginButton = new JButton("Sign in");
        loginButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String userName = username.getText();
                String password = String.valueOf(passwordField.getPassword());
                String query = "select password_,first_name from account where user_name =
\""+userName+"\"";
                try {
                    Class.forName("com.mysql.cj.jdbc.Driver");
                    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/canteen_db", "root", "1234");
                    Statement sta = connection.createStatement();

                    ResultSet rs = sta.executeQuery(query);
                    rs.next();
                    String db_password = rs.getString("password_");
                    //System.out.println(db_password+" entered :"+password);

                    if (db_password.equals(password)) {
                        JOptionPane.showMessageDialog(loginButton,"Welcome, "+
rs.getString("first_name"));
                        dispose();
                        Order o =new Order(userName); //----------------------------------
---------------
                        o.setVisible();
                    } else {
                        JOptionPane.showMessageDialog(loginButton, "Invalid Password");
                    }
                    connection.close();
                } catch (Exception exception) {
                    //exception.printStackTrace();
```

```
                    JOptionPane.showMessageDialog(loginButton, "Invalid User");
                    System.out.println(exception);
                }
            }
        });
        loginButton.setFont(new Font("Tahoma", Font.PLAIN, 18));
        loginButton.setBounds(56, 310, 270, 38);
        loginButton.setFocusPainted(false);
        contentPane.add(loginButton);

        newUserButton = new JButton("New user");
        newUserButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                dispose();
                new UserRegistration();
            }
        });
        newUserButton.setFont(new Font("Tahoma", Font.PLAIN, 18));
        newUserButton.setBounds(56, 370, 270, 38);
        newUserButton.setFocusPainted(false);
        contentPane.add(newUserButton);
    }
}
```

Order.java

```
import java.awt.Color;
import java.awt.Font;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
//import java.sql.*;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;

public class Order extends JFrame implements ActionListener{
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    JTextArea bill1,bill2;
    JButton cart;

    Cart c ;

    public Order(String username) {
```

```
        c = new Cart(username);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Canteen-MS-Place Order");
        setBounds(60, 10, 1200, 700);
        setResizable(false);
        setIconImage(Toolkit.getDefaultToolkit().getImage("img/bg.png"));

        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        getContentPane().setBackground(new Color(52, 46, 45)); //black

        JLabel lblorder = new JLabel("Place your Order");
        lblorder.setFont(new Font("Times New Roman", Font.PLAIN, 35));
        lblorder.setBounds(110, 19, 500, 50);
        lblorder.setForeground(Color.white);  //red
        contentPane.add(lblorder);

        JPanel panel = new JPanel();
        panel.setLayout(null);
        panel.setPreferredSize(new Dimension(115, 1000));

        //System.out.println(new File("img/item1.jpg").getCanonicalPath());

        JLabel lrow1 = new JLabel("Snacks");
        lrow1.setFont(new Font("Times New Roman", Font.PLAIN, 32));
        lrow1.setBounds(525,11,100,32);
        lrow1.setForeground(Color.white);
        panel.add(lrow1);

        JLabel lrow2 = new JLabel("Meals");
        lrow2.setFont(new Font("Times New Roman", Font.PLAIN, 32));
        lrow2.setBounds(525,350,100,32);
        lrow2.setForeground(Color.white);
        panel.add(lrow2);

//        int yrow1=52 ,yrow2=360,  pheight=246,pwidth=262;
//        int xcol1=16 ,xcol2=295 ,xcol3=574 ,xcol4=853;
//        int lx=100,ly=218,lwid=250,lhei=20,by=4,bxy=4,bwid=254,bhei=210;

        int yrow1=52 ,yrow2=390,  pheight=270,pwidth=262;
        int xcol1=16 ,xcol2=295 ,xcol3=574 ,xcol4=853;
        int lx=100,ly=243,lwid=250,lhei=20,by=30,bxy=5,bwid=252,bhei=210;

        JButton bitem1,bitem2,bitem3,bitem4,bitem5,bitem6,bitem7,bitem8;
        JLabel litem1,litem2,litem3,litem4,litem5,litem6,litem7,litem8;
        JLabel lname1,lname2,lname3,lname4,lname5,lname6,lname7,lname8;
        JPanel pitem1,pitem2,pitem3,pitem4,pitem5,pitem6,pitem7,pitem8;

        Color blue =new Color(138, 241, 255); //greenish
        Font menuFont= new Font("Tahoma", Font.PLAIN, 20);
```

```
        Border blackline = BorderFactory.createLineBorder(Color.black,2);
        Font lblname = new Font("Times New Roman", Font.PLAIN, 22);

        pitem1 = new JPanel();
        pitem1.setBounds(xcol1,yrow1,pwidth,pheight);
        pitem1.setBackground(blue);
        litem1 = new JLabel("Rs.25");
        litem1.setFont(menuFont);
        litem1.setBounds(lx,ly,lwid,lhei);
        lname1 = new JLabel("Samosa");
        lname1.setFont(lblname);
        lname1.setBounds(90,6,lwid,lhei);
        bitem1 = new JButton(new ImageIcon("img/samosa.jpg"));
        bitem1.setBounds(bxy,by,bwid,bhei);

        pitem2 = new JPanel();
        pitem2.setBounds(xcol2,yrow1,pwidth,pheight);
        pitem2.setBackground(blue);
        litem2 = new JLabel("Rs.20");
        litem2.setFont(menuFont);
        litem2.setBounds(lx,ly,lwid,lhei);
        lname2 = new JLabel("Vada Pav");
        lname2.setFont(lblname);
        lname2.setBounds(90,6,lwid,lhei);
        bitem2 = new JButton(new ImageIcon("img/vadapav.jpg"));
        bitem2.setBounds(bxy,by,bwid,bhei);

        pitem3 = new JPanel();
        pitem3.setBounds(xcol3,yrow1,pwidth,pheight);
        pitem3.setBackground(blue);
        litem3 = new JLabel("Rs.30");
        litem3.setFont(menuFont);
        litem3.setBounds(lx,ly,lwid,lhei);
        lname3 = new JLabel("Misal Pav");
        lname3.setFont(lblname);
        lname3.setBounds(90,6,lwid,lhei);
        bitem3 = new JButton(new ImageIcon("img/misal.jpg"));
        bitem3.setBounds(bxy,by,bwid,bhei);

        pitem4 = new JPanel();
        pitem4.setBounds(xcol4,yrow1,pwidth,pheight);
        pitem4.setBackground(blue);
        litem4 = new JLabel("Rs.30");
        litem4.setFont(menuFont);
        litem4.setBounds(lx,ly,lwid,lhei);
        lname4 = new JLabel("Pav Bhaji");
        lname4.setFont(lblname);
        lname4.setBounds(90,6,lwid,lhei);
        bitem4 = new JButton(new ImageIcon("img/pavbhaji.jpg"));
        bitem4.setBounds(bxy,by,bwid,bhei);

        pitem5 = new JPanel();
        pitem5.setBounds(xcol1,yrow2,pwidth,pheight);
```

```
        pitem5.setBackground(blue);
        litem5 = new JLabel("Rs.120");
        litem5.setFont(menuFont);
        litem5.setBounds(lx,ly,lwid,lhei);
        lname5 = new JLabel("Chicken Biryani");
        lname5.setFont(lblname);
        lname5.setBounds(66,6,lwid,lhei);
        bitem5 = new JButton(new ImageIcon("img/chikenbiryani.jpg"));
        bitem5.setBounds(bxy,by,bwid,bhei);

        pitem6 = new JPanel();
        pitem6.setBounds(xcol2,yrow2,pwidth,pheight);
        pitem6.setBackground(blue);
        litem6 = new JLabel("Rs.160");
        litem6.setFont(menuFont);
        litem6.setBounds(lx,ly,lwid,lhei);
        lname6 = new JLabel("Idli Sambar");
        lname6.setFont(lblname);
        lname6.setBounds(90,6,lwid,lhei);
        bitem6 = new JButton(new ImageIcon("img/idli.jpg"));
        bitem6.setBounds(bxy,by,bwid,bhei);

        pitem7 = new JPanel();
        pitem7.setBounds(xcol3,yrow2,pwidth,pheight);
        pitem7.setBackground(blue);
        litem7 = new JLabel("Rs.130");
        litem7.setFont(menuFont);
        litem7.setBounds(lx,ly,lwid,lhei);
        lname7 = new JLabel("Chole Bature");
        lname7.setFont(lblname);
        lname7.setBounds(80,6,lwid,lhei);
        bitem7 = new JButton(new ImageIcon("img/cholebhature.jpg"));
        bitem7.setBounds(bxy,by,bwid,bhei);

        pitem8 = new JPanel();
        pitem8.setBounds(xcol4,yrow2,pwidth,pheight);
        pitem8.setBackground(blue);
        litem8 = new JLabel("Rs.140");
        litem8.setFont(menuFont);
        litem8.setBounds(lx,ly,lwid,lhei);
        lname8 = new JLabel("Veg Biryani");
        lname8.setFont(lblname);
        lname8.setBounds(82,6,lwid,lhei);
        bitem8 = new JButton(new ImageIcon("img/vegbiryani.jpg"));
        bitem8.setBounds(bxy,by,bwid,bhei);

        pitem1.setLayout(null);
        pitem2.setLayout(null);
        pitem3.setLayout(null);
        pitem4.setLayout(null);
        pitem5.setLayout(null);
        pitem6.setLayout(null);
        pitem7.setLayout(null);
```

```
        pitem8.setLayout(null);

        pitem1.add(litem1);
        pitem2.add(litem2);
        pitem3.add(litem3);
        pitem4.add(litem4);
        pitem5.add(litem5);
        pitem6.add(litem6);
        pitem7.add(litem7);
        pitem8.add(litem8);

        pitem1.add(lname1);
        pitem2.add(lname2);
        pitem3.add(lname3);
        pitem4.add(lname4);
        pitem5.add(lname5);
        pitem6.add(lname6);
        pitem7.add(lname7);
        pitem8.add(lname8);

        pitem1.add(bitem1);
        pitem2.add(bitem2);
        pitem3.add(bitem3);
        pitem4.add(bitem4);
        pitem5.add(bitem5);
        pitem6.add(bitem6);
        pitem7.add(bitem7);
        pitem8.add(bitem8);

        pitem1.setBorder(blackline);
        pitem2.setBorder(blackline);
        pitem3.setBorder(blackline);
        pitem4.setBorder(blackline);
        pitem5.setBorder(blackline);
        pitem6.setBorder(blackline);
        pitem7.setBorder(blackline);
        pitem8.setBorder(blackline);

        bitem1.addActionListener(this);
        bitem2.addActionListener(this);
        bitem3.addActionListener(this);
        bitem4.addActionListener(this);
        bitem5.addActionListener(this);
        bitem6.addActionListener(this);
        bitem7.addActionListener(this);
        bitem8.addActionListener(this);

        bitem1.setActionCommand("1");
        bitem2.setActionCommand("2");
        bitem3.setActionCommand("3");
        bitem4.setActionCommand("4");
        bitem5.setActionCommand("5");
        bitem6.setActionCommand("6");
```

```
        bitem7.setActionCommand("7");
        bitem8.setActionCommand("8");

        panel.add(pitem1);
        panel.add(pitem2);
        panel.add(pitem3);
        panel.add(pitem4);
        panel.add(pitem5);
        panel.add(pitem6);
        panel.add(pitem7);
        panel.add(pitem8);

        cart= new JButton("Cart");
        cart.setBounds(1000,25,100,40);
        cart.setFont(menuFont);
        cart.setBackground(new Color(92, 85, 84));
        cart.addActionListener(this);
        cart.setFocusPainted(false);
        cart.setForeground(Color.white);
        cart.setBorder(BorderFactory.createLineBorder(Color.white,2));
        contentPane.add(cart);

        panel.setBackground(new Color(92, 85, 84));
        JScrollPane scrollPane = new
JScrollPane(panel,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_A
S_NEEDED);
        scrollPane.setBounds(20,90,1150,500);
        scrollPane.getVerticalScrollBar().setUnitIncrement(16);
        add(scrollPane);
    }

    public void actionPerformed(ActionEvent e){
        switch(e.getActionCommand()) {
        case "1":
            c.addToCart("1");
            break;
        case "2":
            c.addToCart("2");
            break;
        case "3":
            c.addToCart("3");
            break;
        case "4":
            c.addToCart("4");
            break;
        case "5":
            c.addToCart("5");
            break;
        case "6":
            c.addToCart("6");
            break;
        case "7":
            c.addToCart("7");
```

```
            break;
        case "8":
            c.addToCart("8");
            break;
        }

        if(e.getSource()==cart) {
            c.setVisible(true);
        }
    }

    public void setVisible() {
        setVisible(true);
    }

}
```

Cart.java

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;

public class Cart extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextArea itemList, quantity, amt,rate;
    private JButton Customize, calculate, debit ,paynow, Finish;
    private JTextField tftotal;
    private String username;
    private int amtsum=0, paid=0;

    Cart(String Username) {
```

```
        username=Username;
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Canteen-MS-Cart");
        setBounds(60, 10, 1200, 700);
        setResizable(false);

        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        contentPane.setBackground(new Color(52, 46, 45));

        Color blue = new Color(79, 170, 255);
        Font menuFont = new Font("Tahoma", Font.PLAIN, 20);

        JPanel pitemlist,prate,pquan,pamt;
        JPanel panel = new JPanel();
        panel.setLayout(null);
        panel.setBounds(200, 100, 390, 430);
        panel.setBackground(new Color(92, 85, 84));

        Border blackborder = BorderFactory.createLineBorder(Color.black,2);
        int py=36,pos=-19;

        itemList = new JTextArea();
        itemList.setBounds(5, 5, 170, 300);
        itemList.setLineWrap(true);
        itemList.setEditable(false);
        itemList.setBorder(new EmptyBorder(5, 5, 5, 5));
        itemList.setFont(new Font("Tahoma", Font.PLAIN, 17));
        pitemlist = new JPanel();
        pitemlist.setLayout(null);
        pitemlist.setBounds(40+pos, py, 180, 310);
        pitemlist.setBorder(blackborder);
        pitemlist.add(itemList);

        rate = new JTextArea();
        rate.setBounds(05,05, 50, 300);
        rate.setLineWrap(true);
        rate.setEditable(false);
        rate.setBorder(new EmptyBorder(5, 5, 5, 5));
        rate.setFont(new Font("Tahoma", Font.PLAIN, 17));
        rate.setBorder(new EmptyBorder(5, 5, 5, 5));
        prate = new JPanel();
        prate.setLayout(null);
        prate.setBounds(218+pos, py, 60, 310);
        prate.setBorder(blackborder);
        prate.add(rate);

        quantity = new JTextArea();
        quantity.setBounds(05, 05, 40, 300);
        quantity.setLineWrap(true);
        quantity.setCaretColor(Color.WHITE);
```

```
        quantity.setBorder(new EmptyBorder(5, 5, 5, 5));
        quantity.setFont(new Font("Tahoma", Font.PLAIN, 17));
        pquan = new JPanel();
        pquan.setLayout(null);
        pquan.setBounds(276+pos, py, 50, 310);
        pquan.setBorder(blackborder);
        pquan.add(quantity);

        amt = new JTextArea();
        amt.setBounds(05, 05, 50, 300);
        amt.setLineWrap(true);
        amt.setEditable(false);
        amt.setBorder(new EmptyBorder(5, 5, 5, 5));
        amt.setFont(new Font("Tahoma", Font.PLAIN, 17));
        pamt = new JPanel();
        pamt.setLayout(null);
        pamt.setBounds(324+pos, py, 60, 310);
        pamt.setBorder(blackborder);
        pamt.add(amt);

        panel.add(pitemlist);
        panel.add(prate);
        panel.add(pquan);
        panel.add(pamt);


        int ly = 8;
        JLabel litems, lquantity, user, lrate;
        Font lblfont = new Font("Times New Roman", Font.PLAIN, 20);

        litems = new JLabel("Items in cart");
        litems.setFont(lblfont);
        litems.setForeground(Color.white);
        litems.setBounds(70+pos, ly, 200, 32);
        panel.add(litems);

        lrate = new JLabel("Rate");
        lrate.setFont(lblfont);
        lrate.setForeground(Color.white);
        lrate.setBounds(230+pos, ly, 130, 32);
        panel.add(lrate);

        lquantity = new JLabel("Qty");
        lquantity.setFont(lblfont);
        lquantity.setForeground(Color.white);
        lquantity.setBounds(289+pos, ly, 130, 32);
        panel.add(lquantity);

        user = new JLabel("Amt");
        user.setFont(lblfont);
        user.setForeground(Color.white);
        user.setBounds(338+pos, ly, 100, 32);
        panel.add(user);
```

```
        tftotal = new JTextField(" Total : ");
        tftotal.setBounds(242, 350, 123, 32);
        tftotal.setEditable(false);
        tftotal.setFont(new Font("Tahoma", Font.PLAIN, 19));
        tftotal.setBorder(blackborder);
        panel.add(tftotal);

        user = new JLabel("Dear "+username+", confirm your order");
        user.setFont(new Font("Tahoma", Font.PLAIN, 21));
        user.setForeground(Color.white);
        user.setBounds(70, 30, 600, 32);
        add(user);

        Customize = new JButton("Edit Cart");
        Customize.setBounds(900, 130, 170, 40);
        Customize.setFont(menuFont);
        Customize.setBackground(blue);
        Customize.addActionListener(this);
        Customize.setFocusPainted(false);
        add(Customize);

        calculate = new JButton("Calculate Bill");
        calculate.setBounds(900, 210, 170, 40);
        calculate.setFont(menuFont);
        calculate.setBackground(blue);
        calculate.addActionListener(this);
        calculate.setFocusPainted(false);
        add(calculate);

        debit = new JButton("Pay Later");
        debit.setBounds(900, 320, 170, 40);
        debit.setFont(menuFont);
        debit.setBackground(blue);
        debit.addActionListener(this);
        debit.setFocusPainted(false);
        add(debit);

        paynow = new JButton("Pay Now");
        paynow.setBounds(900, 370, 170, 40);
        paynow.setFont(menuFont);
        paynow.setBackground(blue);
        paynow.addActionListener(this);
        paynow.setFocusPainted(false);
        add(paynow);

        Finish = new JButton("Place Order");
        Finish.setBounds(900, 420, 170, 40);
        Finish.setFont(menuFont);
        Finish.setBackground(new Color(0, 20, 153));
        Finish.setEnabled(false);
        Finish.addActionListener(this);
        Finish.setFocusPainted(false);
```

```
        add(Finish);

        add(panel);
    }

    int count=0;
    public void addToCart(String inp) {

        switch (inp) {
            case "1":
                count++;
                itemList.append(count+". Samosa\n");
                rate.append("25\n");
                break;
            case "2":
                count++;
                itemList.append(count+". Vada Pav\n");
                rate.append("20\n");
                break;
            case "3":
                count++;
                itemList.append(count+". Misal Pav\n");
                rate.append("30\n");
                break;
            case "4":
                count++;
                itemList.append(count+". Pav Bhaji\n");
                rate.append("30\n");
                break;
            case "5":
                count++;
                itemList.append(count+". Chicken Biryani\n");
                rate.append("150\n");
                break;
            case "6":
                count++;
                itemList.append(count+". Idli Sambar\n");
                rate.append("160\n");
                break;
            case "7":
                count++;
                itemList.append(count+". Chole Bature\n");
                rate.append("130\n");
                break;
            case "8":
                count++;
                itemList.append(count+". Veg Biryani\n");
                rate.append("140\n");
                break;
        }
    }

    public void setVisible() {
```

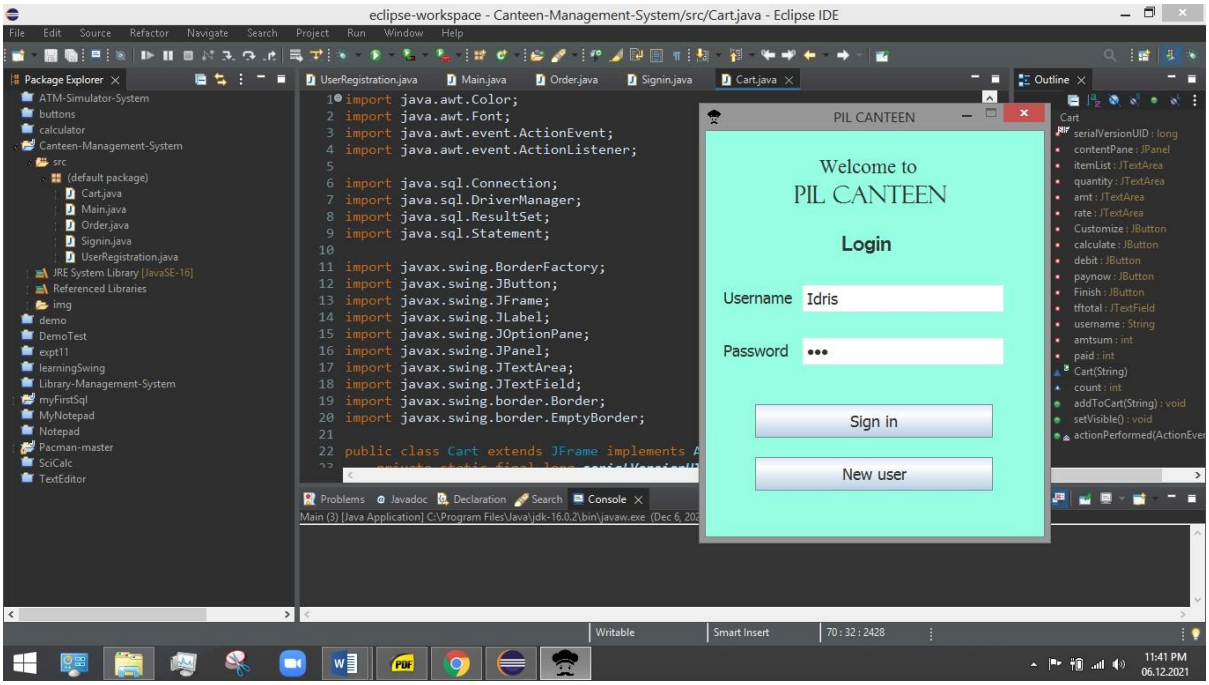
```
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == Customize) {
            setVisible(false);

        }

        else if (e.getSource() == calculate) {
            amt.setText(null);
            tftotal.setText(null);
            tftotal.replaceSelection(" Total : ");

            String sQuan = quantity.getText();
            String sItem = rate.getText();

            int[][] arrQuan = new int[10][3];

            StringBuffer tempQuan = new StringBuffer();
            StringBuffer tempItem = new StringBuffer();

            int lengthQuan=0;
            for(int i=0;i<sQuan.length();i++){
                while(i<sQuan.length() && sQuan.charAt(i)!='\n'){
                    tempQuan.append(Character.toString(sQuan.charAt(i)));
                    i++;
                }
                arrQuan[lengthQuan][0]=Integer.parseInt(tempQuan.toString());
                lengthQuan++;
                tempQuan.delete(0, tempQuan.length());
            }

            lengthQuan=0;
            for(int i=0;i<sItem.length();i++){
                while(i<sItem.length() && sItem.charAt(i)!='\n'){
                    tempItem.append(Character.toString(sItem.charAt(i)));
                    i++;
                }
                arrQuan[lengthQuan][1]=Integer.parseInt(tempItem.toString());
                lengthQuan++;
                tempItem.delete(0, tempItem.length());
            }

            for(int i=0;i<lengthQuan;i++){
                //System.out.println(arrQuan[i][0]+" - "+arrQuan[i][1]);
                amtsum = amtsum + arrQuan[i][0]*arrQuan[i][1];
                amt.append(Integer.toString(arrQuan[i][0]*arrQuan[i][1])+"\n");
            }
            tftotal.replaceSelection(Integer.toString(amtsum));
```

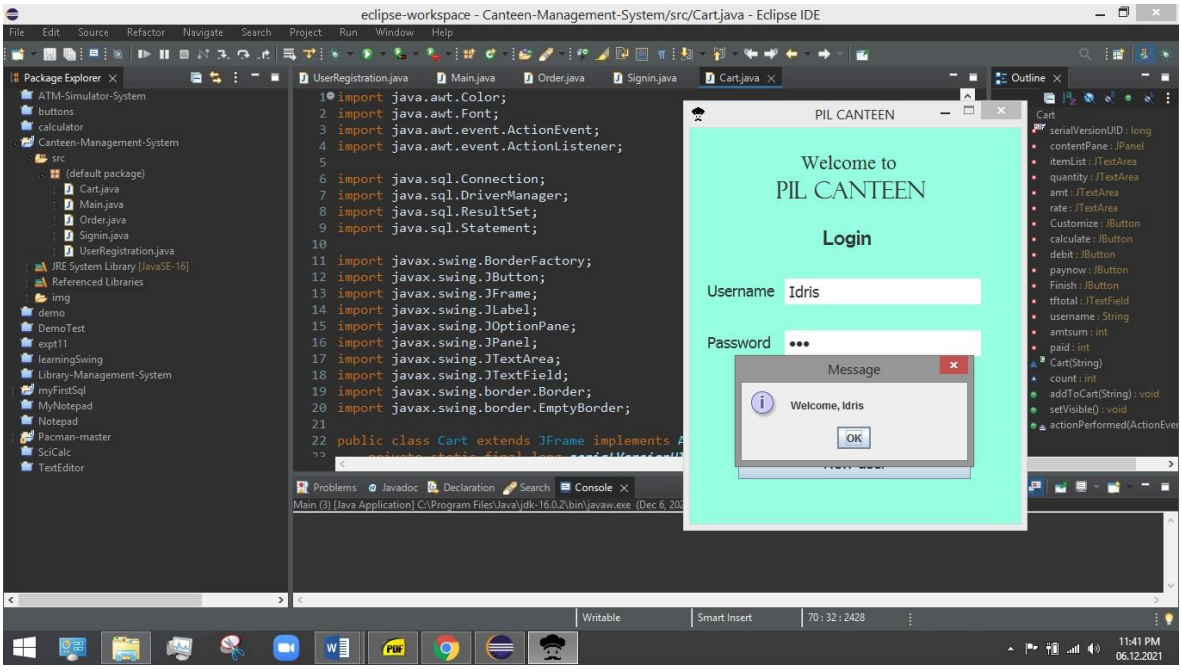
```
        }
        else if (e.getSource() == debit) {

            quantity.setEditable(false);
            String query1 = "select debt from account where user_name = \""+username+"\"";
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/canteen_db", "root", "1234");
                Statement sta = connection.createStatement();

                ResultSet rs = sta.executeQuery(query1);
                rs.next();
                int prevDebt = rs.getInt("debt");

                String query2 = "UPDATE account SET debt="+ (amtsum+prevDebt) +" WHERE
user_name = \""+username+"\"";
                sta.executeUpdate(query2);
                //System.out.println();

                connection.close();
            } catch (Exception exception) {
                //exception.printStackTrace();
                System.out.println(exception);
            }
            JOptionPane.showMessageDialog(debit, "Dear "+username+", Rs. "+amtsum+" have
been\n added to your balance");
            Finish.setEnabled(true);
            Finish.setBackground(new Color(79, 170, 255));
        }
        else if (e.getSource() == paynow) {
            quantity.setEditable(false);
            paid=1;
            Finish.setEnabled(true);
            Finish.setBackground(new Color(79, 170, 255));
            JOptionPane.showMessageDialog(Finish, "Dear "+username+", pay Rs. "+amtsum+"
to counter");
        }
        else if(e.getSource() == Finish) {

            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/canteen_db", "root", "1234");
                Statement sta = connection.createStatement();

                String sit = itemList.getText();
                String squ = quantity.getText();
                String sam = amt.getText();
                StringBuffer tempIt = new StringBuffer();
                StringBuffer tempqu = new StringBuffer();
                StringBuffer tempam = new StringBuffer();
```

```
                int k=0,l=0,m=0; //=============================================
                for(int j=0;j<count;j++){

                    while(k<sit.length() && sit.charAt(k)!='\n'){
                        tempIt.append(Character.toString(sit.charAt(k)));
                        k++;
                    }
                    String itemname=tempIt.toString()+"  ";
                    tempIt.delete(0, tempIt.length());
                    k++;

                    while(l<squ.length() && squ.charAt(l)!='\n'){
                        tempqu.append(Character.toString(squ.charAt(l)));
                        l++;
                    }
                    int quanno =Integer.parseInt(tempqu.toString());
                    tempqu.delete(0, tempqu.length());
                    l++;

                    while(m<sam.length() && sam.charAt(m)!='\n'){
                        tempam.append(Character.toString(sam.charAt(m)));
                        m++;
                    }
                    int amount =Integer.parseInt(tempam.toString());
                    tempam.delete(0, tempam.length());
                    m++;
                    //System.out.println(itemname+"-"+quanno+amount);
                    String query = "INSERT INTO order_record values(now(),'" + username +
"','" + itemname + "'," + quanno + "," + amount + "," + paid + ")";
                    sta.executeUpdate(query);
                }

                connection.close();
            } catch (Exception exception) {
                //exception.printStackTrace();
                System.out.println(exception);
            }
            dispose();
            Order o = new Order(username);
            o.setVisible();
        }

    }
}
```

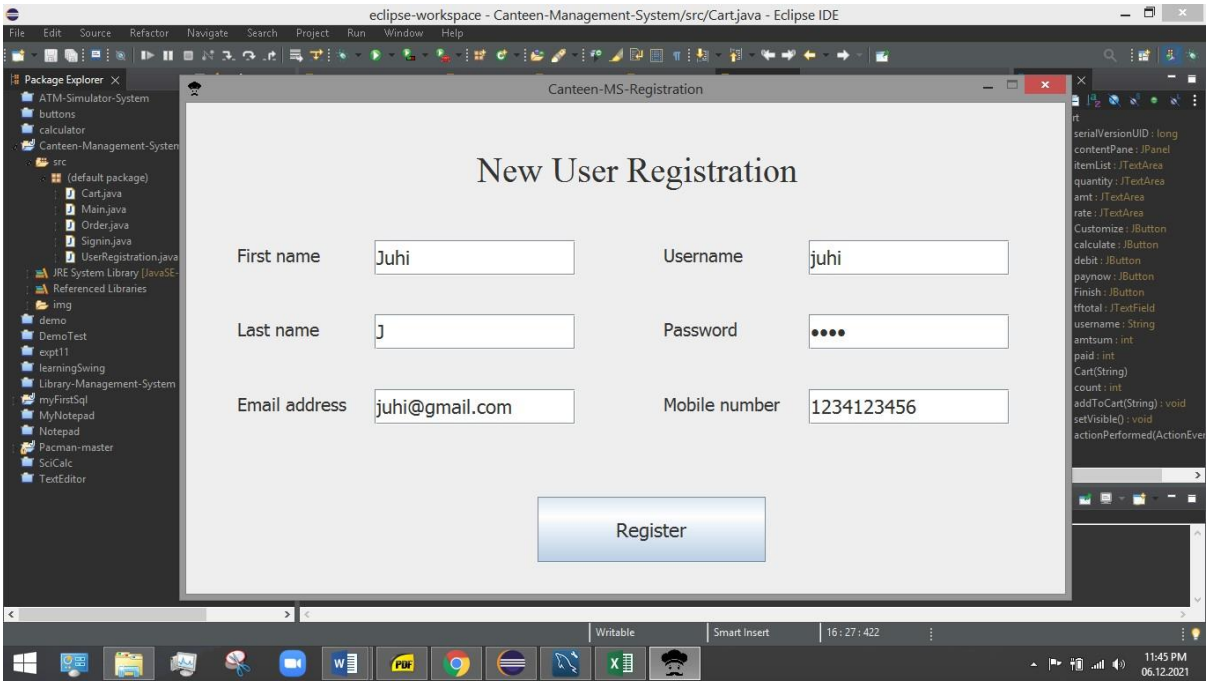
1. OUTPUT

Sign-in page-



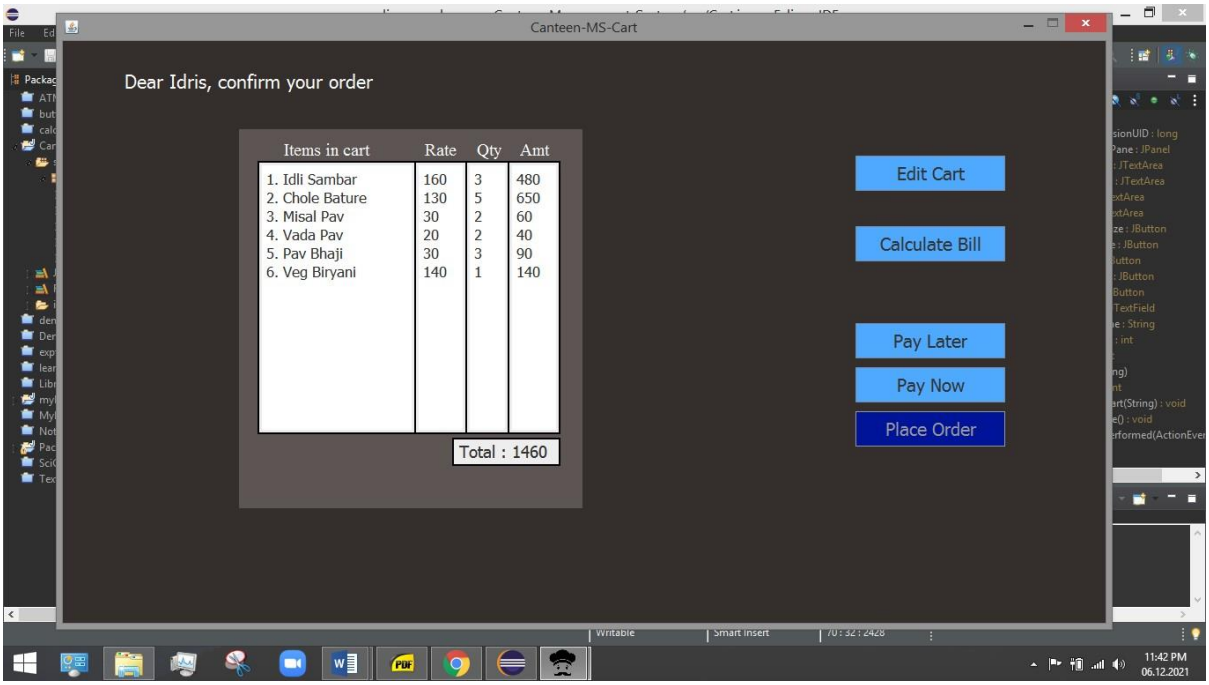


New User Registraton-

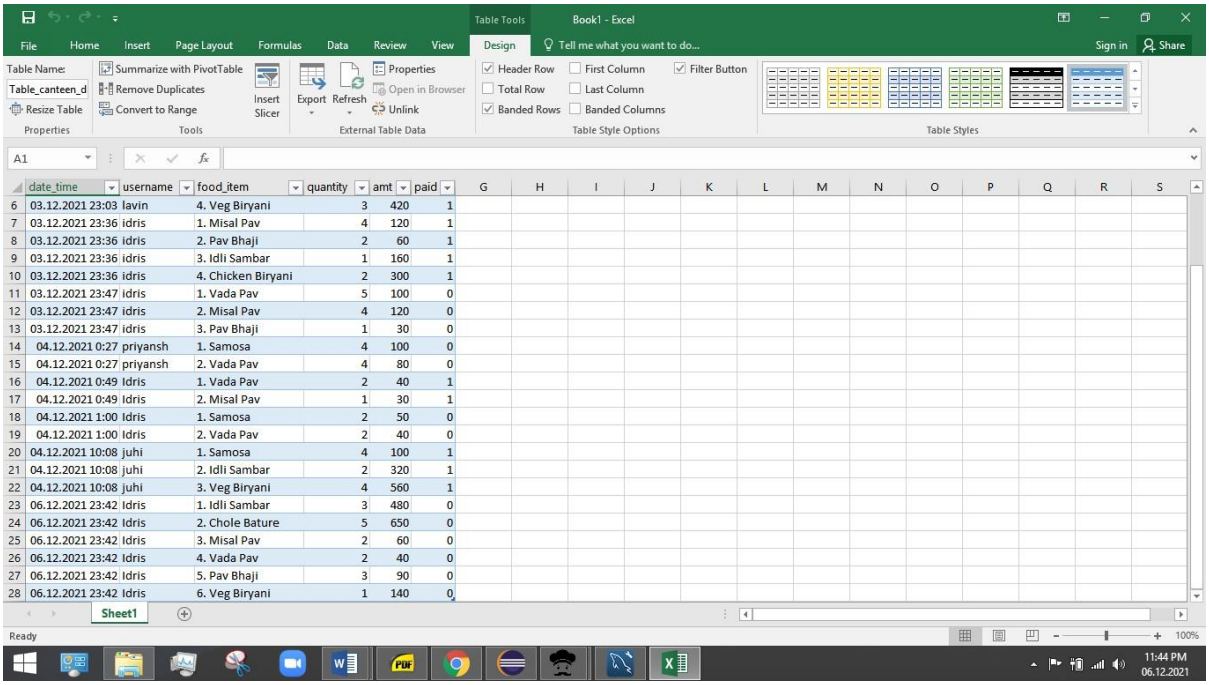


Menu Cum Place your Order page-

Cart Page-



Database Result in Spreadsheet-



Database for Calculating Debt-