Jawaharlal Nehru Engineering College

Laboratory Manual

PROGRAMMING IN JAVA

For

Second Year Students
Department of Computer Science and Engineering

FOREWORD

It is my great pleasure to present this laboratory manual for Third Year engineering students for the subject of Programming In Java keeping in view the vast coverage required for Programming with Java Language

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9000 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9000 Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. H. H. Shinde
Principal

LABORATORY MANUAL CONTENTS

This manual is intended for the Third Year students of Computer Science and Engineering in the subject of Programming In Java. This manual typically contains practical/Lab Sessions related Programming In Java covering various aspects related the subject to enhanced understanding.

As per the syllabus along with Study of Java Language, we have made the efforts to cover various aspects of Programming In Java covering different Techniques used to construct and understand concepts of Java Programming.

Students are advised to thoroughly go though this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Prof. S.N. Jaiswal
CSE Department

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.

2. All the students should sit according to their roll numbers starting from their left to right.

3. All the students are supposed to enter the terminal number in the log book.

4. Do not change the terminal on which you are working.

5. All the students are expected to get at least the algorithm of the program/concept to be implemented.

6. Strictly observe the instructions given by the teacher/Lab Instructor.

Instruction for Laboratory Teachers::

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.

2. Students should be taught for taking the printouts under the observation of lab teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

SUBJECT INDEX

1. Study of syntax & Semantics of java

Aim: Write simple programs to understand Input-Output statements, Decision making, control flow & Operators in java.

Objective: The objective of this experiment is that student must familiar with java environment, understand syntax & semantics of java programming language & enhance their programming confidence. Student must use all the statements, control structure, looping statements, type casting etc.

H/w, S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

Java is true object oriented programming language with the important Features like Platform dependencies, distributed, robust, secure, multitasking, dynamic, and many more.

In this program we are finding the Prime No's between the ranges. Depends upon the value of the variable we are initializing in program. We are printing the series of prime No's within that range. The particular part is explained below with the Java programming Syntax.

Class Declaration:

As java is true Object oriented language therefore everything must be placed inside the class.

Syntax: class class-Name

E.g. class Demo

Class is a keyword, Demo is the identifier that specifies the name of the class to be defined inside opening & closing Braces.

The Main Line:

Every java application program must include the main() method.

Syntax: public static void main (String args[])

Every program begins with the main() method. All the Java applications begin execution by calling main(). The full meaning of each part of this is:

public:

The public keyword is an access specifier, which allows the programmer to control the visibility of class members. When a class member is preceded by public, then that member may be accessed by code outside the class in which it is declared. main( ) must be declared as public, since it must be called by code outside of its classe when the program is started.

static:

The keyword static allows main( ) to be called without having to instantiate a particular instance of the class. This is necessary since main ( ) is called by the JVM before any objects are made.

void:

The keyword void simply tells the compiler that main ( ) doesn't return a value.

String args [ ]:

In main ( ) , there is only one parameter, albeit a complicated one. String args [ ] declares a parameter named args, which is an array of instances of the class String. args receives any command-line arguments present when the program is executed.

The Output Line:

The executable statement in the program is

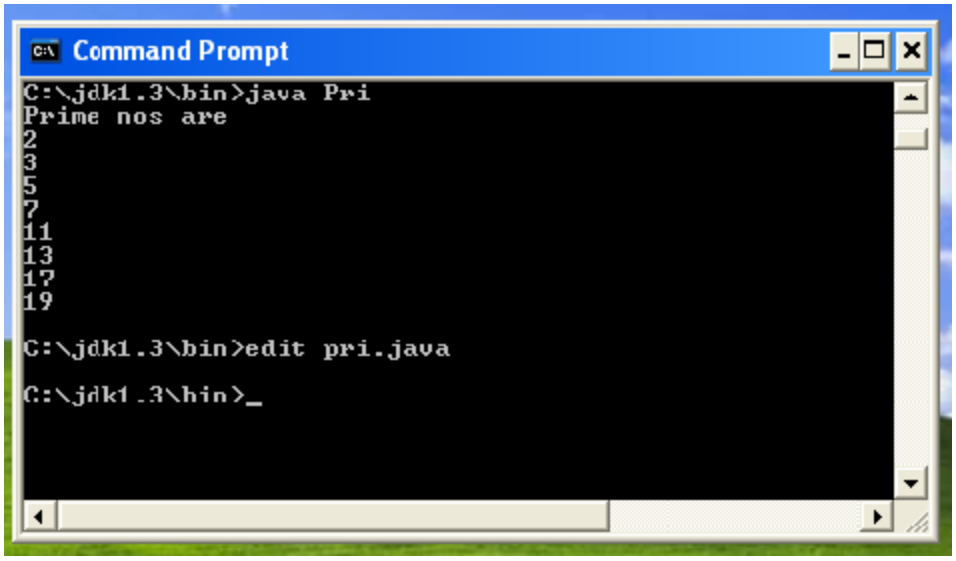
Syntax: System.out.println("Java is better then C++");

The println() method invoked with the object out of the System class. This line prints the string Java is better then C++ in Command prompt.

Algorithm:

1) Declare and define a class. Define main method .
2) Declare variables and initialize variables.
3) Implement different operators in Java for e.g. Arithmetic ,logical, conditional etc. with the help of variables.
4) Print all the values of variables, by using basic syntax of Java language.

Output:



Conclusion:

Hence we studied Java Program for finding Prime Nos.

Journal Write-up:

- History of java
- Features of java
- Java virtual machine
- Java architecture & JDK tools
- The difference between C++ & java

- The difference between java & HTML
- Data types in java
- Type casting
- Control Statements
- Looping Statements
- Java program structure

Additional List of Programs :

1. W.A.P. to find the sum of the digits of the given integer.
2. W.A.P. to obtain the **Gross salary** by calculating **DA & HRA.**
3. W.A.P. to convert **kilometers in to meters**.
4. W.A.P. to convert temperature from Fahrenheit to Celsius using the formula " c = (f - 32) * 5/9 "
5. W.A.P. to find highest of three numbers by using **If** loop.
6. W.A.P. to find highest of three numbers by using **If-Else** loop.
7. W.A.P. to find highest of three numbers and give the output in **ascending** order.
8. W.A.P. to find the sum and average of ten numbers by using **For** loop.
9. W.A.P. to find the given number is **Armstrong** or not.
10. W.A.P. to generate the **series of even numbers** solution.
11. W.A.P. to generate **factorial** of given number.
12. W.A.P. to generate **Fibonacci series**.
13. W.A.P. to find sum of the N natural numbers.
14. W.A.P. to generate the **series of odd numbers** solution.
15. W.A.P. to generate the following pattern

```
1
1   2
1   2   3
1   2   3   4
1   2   3   4   5
```

16. W.A.P. to generate the following pattern

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

17. W.A.P. to generate the following pattern

* * * * *

* * * *

* * *

* *

*

18. W.A.P. to generate the following pattern.

*

* *

* * *

* * * *

* * * * *

19. W.A.P. to generate the

A

A B

A B C

A B C D

A B C D E

20.W.A.P. to generate the

1

212

32123

4321234

543212345

21W.A.P. to find reverse of the number.

22. W.A.P. to find the sum of even and odd numbers.

23. W.A.P. to generate table of a number.

24. W.A.P. to display all the characters represented by the ASCII numbers from 25 to100.

25. W.A.P. to find the greatest common divisor of the entered number n.

26. W.A.P. to demonstrate the octal & hexadecimal representation of an integer number

27. . W.A.P. to demonstrate the use of shift operators

28. . W.A.P. to find a five digit number which on multiplication by 4 reverses its order

29. . W.A.P. to find all four digit perfect squares, where the number forming by first two digits and number forming by last two digits are also perfect.

30. W.A.P. to create a Floyd triangle of numbers

1

2 3 4 5 6 7 8 9 10 2

3 4 5 6 7 8 9 10 3

4 5 6 7 8 9 10 4

…

10 10

31. W. A.P. to print I,J,K,L such that I <J<K<L , L=I+J+K where I,J,K <100

32. W.A.P. to print 4 digit number into word.

2. Study of Arrays & Vectors in java

Aim: Program for Matrix Operations using Arrays in java

Objective : Student should understand the concept of one & multi-dimensional arrays, use vectors appropriately.

H/w,S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

Arrays:

An array is a group of like-typed variables. To create an array, you have to follow fllowing steps:

1. Declaring an Array.
2. Creating memory location
3. Putting values in memory location

One-dimensional Arrays:

A list of items can be given one variable name using only one subscript and such a variable called a single subscript variable or One-dimensional array.

The general form of a one-dimensional array declaration is:

Syntax: type var-name[ ] ;

Here, type declares the base type of the array. The base type determines the data type of each element that comprises the array. Thus the base type for the array determines what type of data the array will hold.

E.g : int month_days [ ] ;

The above line declares an array name month days with the type “array of int” .

new operator:

It is a special operator that allocates memory. The general form of new as it applies to one-dimensional arrays appears as follows:

Syntax: array-var = new type [size] ;

here, type specifies the type of data being allocated, size specifies the number of elements in the array, and the array-var is the array variable that is linked to the array. That is, to use new to allocate an array, you must specify the type and the number of elements to allocate. The elements in an array allocated by new will automatically be initialize to zero.

E.g: Month_days = new int [12] ;

This example allocates a 12 – element array of integers and links them to month_days.

Multidimensional Arrays:

In Java, multidimensional arrays are actually arrays of arrays. These, as you might expect, look and act like regular multidimensional arrays. However, as you will see, there are a couple of subtle differences. To declare this variable, specify each additional index using another set of square brakets. For example, the following declares a two-dimensional array variable called twoD

int twoD [] [] = new int [4] [5] ;

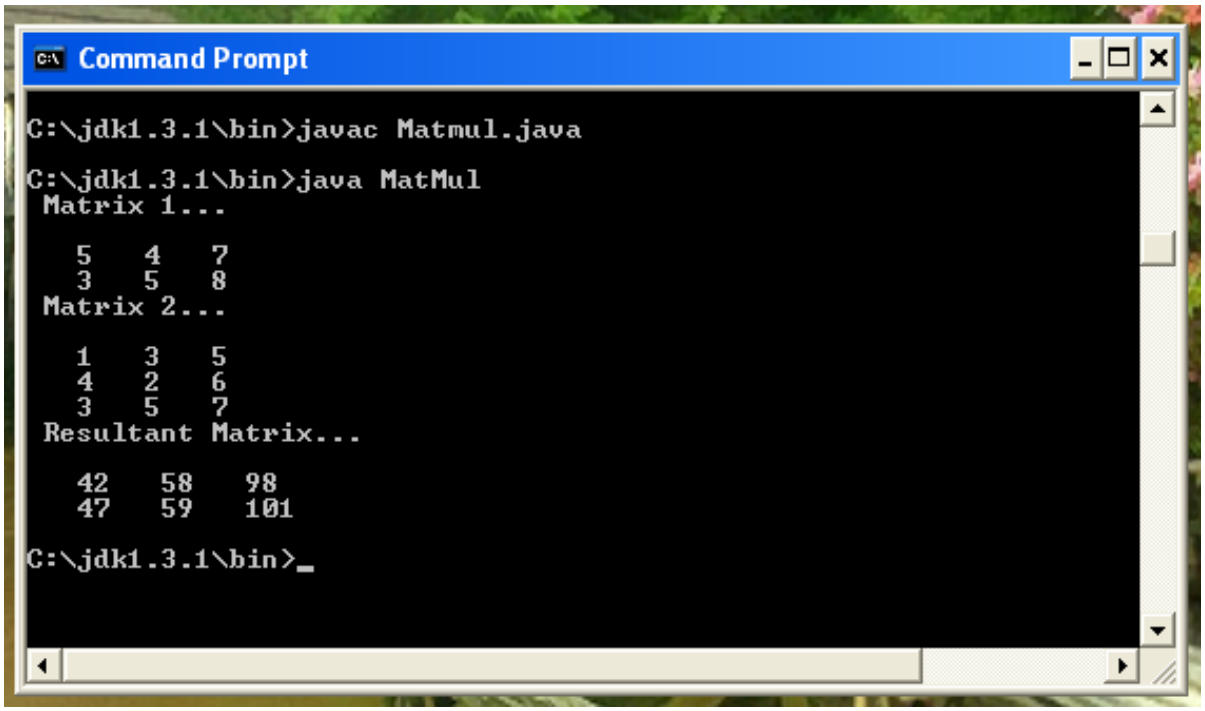
This allocates a 4 by 5 array and assigns it to twoD. Internally this matrix is implements as an array of arrays of int.

Algorithm:

1) *Define a class* multidimensional arrays

2) *Initialize the* multidimensional arrays
3) *Declare variables.*
4) Implement for loops to access specific location of array elements.
5) Print array.

Output:



Conclusion: Hence we studied Arrays & Vectros in java.

Journal Write-up:

- Introduction
- Array Initialization
- Copying Arrays
- Multidimensional Arrays
- Vectors
- Common Vector Methods

Additional List of Programs :

1. WAP to print highest & lowest number from an array
2. WAP to implement different sorting techniques
3. WAP to implement different searching techniques
4. WAP to find sum & average of numbers
5. W.A.P. to display even & odd numbers from an array.
6. W.A.P. to find out prime numbers from given array.
7. W.A.P. to find first & second biggest number from an array

8. W.A.P. to implement the concept of interpolation
9. W.A.P. to create an array of company name & the price quoted. Fetch the company name who has quoted the lowest amount.
10. W.A.P. to find out sum of diagonal element of a matrix.
11. WAP to perform matrix operations
12. WAP to check the equality of two matrices
13. WAP for Eight Queen problem
14. WAP to accept name of five students from the command line & store them in a vector. Perform the following operations-

- Display the size & capacity
- Delete an item in the list
- Add an item at a specified location
- Print the contents of the vector

15. Use a two-dimensional array to solve the following problem: A company has four salespeople(1to 5).Once a day, each salesperson passes in a slip for each type of product sold. Each slip contains the following.

a) The salesperson number

b) The product number

c) The total dollar value of that product sold that day

Thus, each salesperson passes in between 0 to 5 sales slip per day. Assume that the information from all the slip for the last month is available. Write an Application that will read all the information for last month's sale and summarize the total sale by the salesperson and by the product. All totals should be store in two dimensional array sales. After processing all the information for last month .display the result in tabular format, with each column represent the particular salesperson and each row represent particular product. Cross-total each row to get the total

sales of each product for last month. Cross-total each column to get the total sales by salesperson for last month your tabular output should include these cross-totals to the right of the totaled rows and to the bottom of the totaled columns.

16. W.A.P. to create student class with attribute roll number, name, mark1, mark2, total marks & result. Create array of student object.
17. W.A.P. to create student class with attribute roll number, name, dob, weight, height, mark. Write a suitable constructor and a method to display the details of student object. In the main method create an array of 10 student objects & display the roll number and name of students who are 19 years old with weight above 90.5 kg but height less than 175.0 cm.
18. W. A. P. for generation and display of spiral matrix of order 5 as shown below

```
1   2   3   4   5
16  17  18  19  6
15  24  25  20  7
14  23  22  21  8
13  12  11  10  9
```

3. Program to demonstrate Classes and Objects in java

Aim: Write a program which demonstrate Classes and objects.

H/w,S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

Java is true object oriented programming language so everything is placed inside the class. Class is user defined data type containing Members of class.

Class Definition:

Syntax

```
class classname {
        type instance-variable1;
        type instance-variable2;
        //……
        type instance-variableN;
        type methodName1 ( parameter-list) {
            // body of method
        }
        type methodName2 ( parameter-list) {
            // body of method
        }
        //……
        type methodNameN ( parameter-list ) {
            // body of method
        }
}
```

E.g:

```
class Demo
{  int i;
   void getdata();
}
```

Class is keyword and declares that a new class definition follows. Demo is a java identifier that specifies the name of class to be defined with the class Members .

The general form of a method declaration is

```
type methodname ( parameter – list )
{
        method – body ;
}
```

Method declaration have four basic parts:

1. The name of the method
2. The type of the value the method returns
3. A list of parameters
4. The body of the method

Creating Object:

Object is instance of class, is block of memory that contains space to store all the instance variables. Object in java is created using new operator. It is a special operator that allocates memory.

Syntax: Obj-name=new Class-name();

E.g: Student S1; // Declare the object

S1 = new Student() ; //instantiate the object

S1 is the object of Student class.

Accessing class Members:

Once we create object of class,each containing its set of variables we should assign values to those variables in order to use them in our program. All variables must assign values before they used.

Syntax:

```
classname classobjname  = new classname ( ) ;
    ObjectName.Variablename = Value;
  ObjectName.MethodName(Parameter List);
```

Here Object name is name of object,variable name is name of instance variable inside the object that we wish to access,Method name is the method we wish to call.

E.g: S1.Marks = 80;

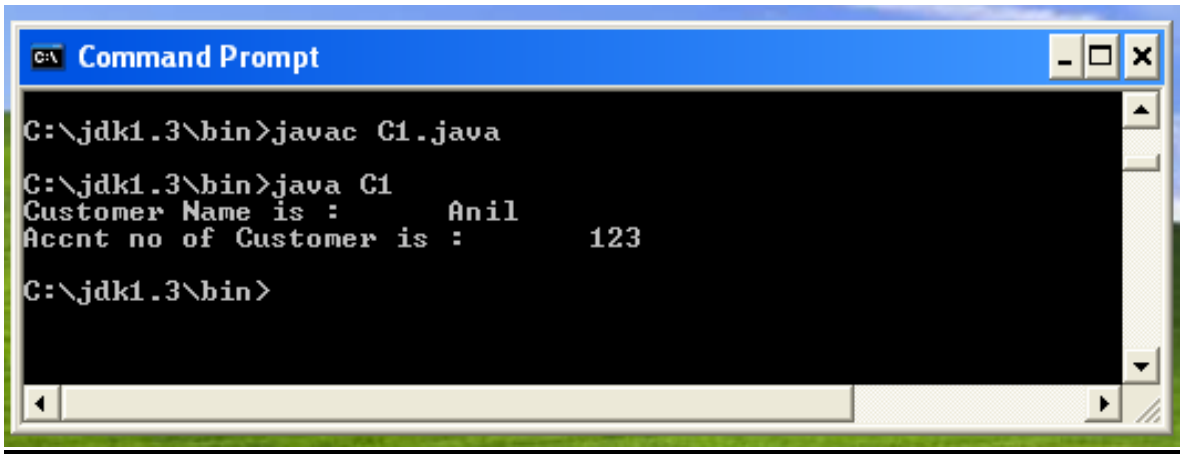
S1.getdata();

S1 is object name, Marks is variable name,getdata is Methos which is accessed with the object and dot operator.

Algorithm:

1) Create any class , define class variables, member functions of class.
2) Initialize the class variables
3) Define main class , create object of above class with the help of (.) Dot operator and obj name.
4) Access members of above class in main class.

Output:



Conclusion:

Hence we studied how to retrieve bank information with the help of classes and objects in java.

Journal Write-up:

- What are classes ?
- Defining a class
- Rules for Naming Classes
- Creating an Object
- Methods : Declaration & Invoking
- *New* Operator
- Constructors & Their types
- *This* keyword
- Access Specifiers
- *Static* Keyword
- Abstract class

Additional List of Programs :

1. Create a class of object interest with a constructor. WAP to find the simple interest using the formula

   Simple Interest=PNR/100 Where P –principal amount N – No of years R –rate of interest
2. WAP using different type of constructors to find the area of rectangle.
3. WAP to find average of the marks of students. Use methods & constructors.
4. WAP to find product of two numbers using Default constructor.
5. WAP to implement complex number operations
6. WAP to implement Stack
7. WAP to implement Link list
8. WAP to demonstrate Method Overloading
9. Create a class Employee that includes three instances variables – a first name, a last name & a monthly salary. Provide a constructor that initializes three variables. Provide a set & get method for each

instance variable. If the monthly salary is not positive, do not set its value. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects & display each object's yearly salary. Then give each employee a 10% raise and display each Employee's yearly salary again.

10. Create a class called Invoice that a hardware store might use to represent for an item sold at the store. An Invoice should include four pieces of information as instance variable – a part number, a part description, a quantity of the item being purchased & a price per item. Your class should have a constructor which initializes the four instance variables. Provide a set & get method for each instance variable. In addition, provide a method named getInvoiceAmount that calculates the invoice amount, then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application name InvoiceTest that demonstrates class Invoice's capabilities.

4. To study methods, method overloading, constructors in java

Aim: To study methods, method overloading, constructors in java

Theory:

Method overloading: Method having same name but different parameter.

Constructor overloading: Constructor having different parameters.

Example:-

Program explains the concept of method overloading and constructor overloading.

```
class Cs
{
int p,q;
public Cs()
{
}
public Cs(int x, int y)
{
p=x;
q=y;
}
public int add(int i, int j)
{
return (i+j);
}
public int add(int i, int j, int k)
{
return (i+j+k);
}
public float add(float f1, float f2)
{
return (f1+f2);
}
public void printData()
{
System.out.print("p = "+p);
System.out.println(" q = "+q);
```

```
}
}
class Hari
{
public static void main(String args[ ])
{
int x=2, y=3, z=4;
Cs c=new Cs();
Cs c1=new Cs(x, z);
c1.printData();
float m=7.2, n=5.2;
int k=c.add(x,y);
int t=c.add(x,y,z);
float ft=c.add(m, n);
System.out.println("k = "+k);
System.out.println("t = "+t);
System.out.println("ft = "+ft);
}
}
```

Output:

p=2

q=4

k=5

t=9

ft=12.400000

Exercise:

1. Wap to find out area of a triangle,rectangle,square and circle using method overloading and constructor overloading.

2. Write a JAVA program which contains a method square() such that square(3) returns 9, square(0.2) returns 0.04.

3. Write a JAVA program which contains a method cube() such that cube(3) returns 27, cube(0.2) returns 0.008.

4. Write a JAVA program which contains a method fun() such that fun(x) returns x and fun(x,y) returns x2+ y2 (where x and y are integers).

5. Write a JAVA program which contains a method fun() such that fun(x) returns x and fun(x,y) returns x + y and fun(x,y,z) returns x*y*z. (where x, y and z are integers).

6. Write a set of overloaded methods min() that returns the smaller of two numbers passed to them as arguments. Make versions for int and float.

7. Write a set of overloaded methods power() that returns the Xn where n is integer and X maybe int and float.

8. Write a set of overloaded methods max() that returns the biggest of two numbers passed to them as arguments. Make versions for int and float.

5. To study Inheritance and types of inheritance in java

Aim: Write a program which demonstrates multilevel inheritance in java.

H/w,S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

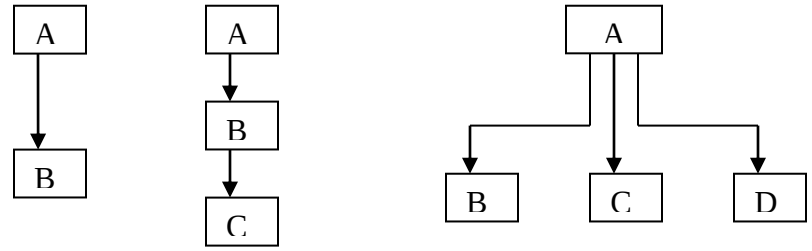
Reusability is important aspect of object oriented programming. It is always nice that we would reuse something that is already exists rather then creating same all over again. Java supports this concept. Java classes can be reused in several ways. This is done by creating new classes reusing the properties of existing once.

The mechanism of deriving new class from old class is called Inheritance. The old class is known as base class (Parent class) and new class is known as Derived class (child class).

The inheritance allows child class to inherit all the variables and methods of their parent classes.

Types of inheritance:

- Single inheritance
- Multilevel inheritance
- Hierarchical inheritance



Single Multilevel Hierarchical

Defining a Sub-class:

A sub-Class is defined as follows:

Syntax: class sub-classname extends superclassname

{ Variables declarartion;

Method declaration;

}

The extends keyword signifies that the properties of the superclass name are extended to the subclass name. The subclass also contain its own variables and methods along with super class.

A derived class with multilevel base class is as follows:

```
Class A
{
}
Class B extends A       //First Level
{
}
Class C extends B       //Second Level
{
}
```

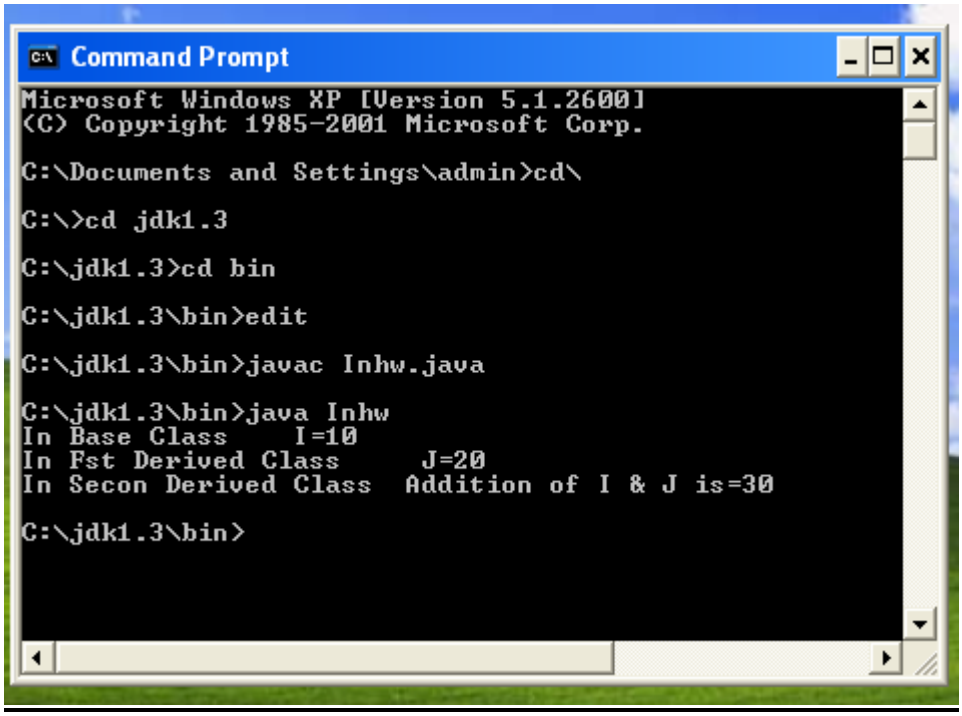
This process may be extended to any members of level. The C class can inherit members of both class A and B.

Java does not directly implement Multiple inheritance. This concept is implemented using a secondary inheritance path in the form of interfaces.

Algorithm:

1) Define base class/ Parent class , Define members of base class
2) Derive child class from base class with the help of extend keyword.
3) Derived another class from already derived one.
4) Declare and define members of classes.
5) Define main class, create object of class which you want to access .
6) You can access members of base class in derived class just by creating object of that class.

Output:



Conclusion:

Hence we studied how to implement Multilevel Inheritance in java.

6. To study interface in java

Aim: *To study interface in java*

Theory:

An interface in java is a blueprint of a class. It has static constants and abstract methods only.

The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship. It cannot be instantiated just like abstract class.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.:

It is used to achieve fully abstraction.

By interface, we can support the functionality of multiple inheritance.

It can be used to achieve loose coupling.

Declaring Interfaces:

The interface keyword is used to declare an interface. Here is a simple example to declare an interface:

Example:

Let us look at an example that depicts encapsulation:

```
public interface NameOfInterface
{
//Any number of final, static fields
//Any number of abstract method declarations
```

}

Interfaces have the following properties:

An interface is implicitly abstract.

You do not need to use the abstract keyword when declaring an interface.

Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.

Methods in an interface are implicitly public.

Example:

```
interface Animal
{
public void eat();
public void travel();
}
```

Implementing Interfaces:

When a class implements an interface, you can think of the class as signing a contract, agreeing to

perform the specific behaviors of the interface. If a class does not perform all the behaviors of the interface, the class must declare itself as abstract.

A class uses the implements keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.

```
public class Mammal implements Animal
{
public void eat()
{
System.out.println("Mammal eats");
}
```

```
public void travel()
{
System.out.println("Mammal travels");
}
public int noOfLegs()
{
return 0;
}
public static void main(String args[])
{
Mammal m =new Mammal();
m.eat();
m.travel();
}
}
```

This would produce the following result:

```
Mammal eats
Mammal travels
```

Example:-

```
interface printable
{
void print();
}
class A6 implements printable
{
public void print()
{
```

```
System.out.println("Hello");
}
public static void main(String args[])
{
A6 obj = new A6();
obj.print();
}
}
```

Output:

Hello

Exercise:-

1. WAP to find out factorial of a number using interface ?

2. WAP to find out greatest among three numbers using interface?

3. WAP to find out gcd and lcm using interface ?

4. WAP to find out area and perimeter of rectangle using interface ?

5. WAP to find out area and perimeter of square using interface ?

6. WAP to find out area and perimeter of traingle using interface ?

7. WAP to find out area and perimeter of circle using interface ?

8. WAP to find out area and perimeter of rectangle ,square and triangle using interface and method overloading ?

9. WAP to find out area and perimeter of rectangle ,square and triangle using interface and method overriding ?

7. To study packages in JAVA

Aim: Write a program to find balance of an account using Package which demonstrates package.

H/w,S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

Package:

Packages are containers for classes that are used to keep the class name space compartmentalized. It is defined as a collection of classes. Packages are similar to the directories and classes within a package are similar to files in the directory. Two categories of packages are in java

1. Java API Package / Predefined Package / Built in Package

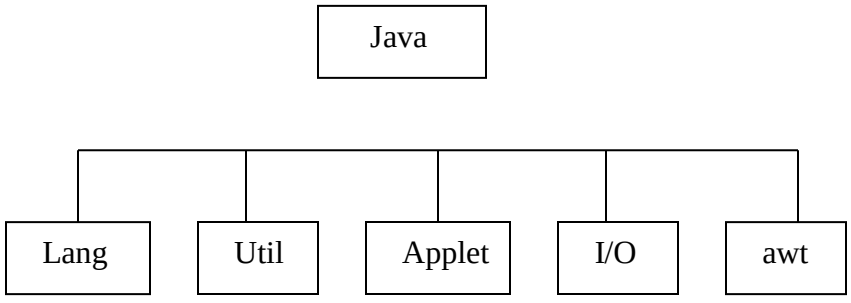


Fig: Frequently used API Packages

Java API provides a large no of Classes grouped in different packages according to functionality.

2. User defined Package:

Creating Package:

To create a package, you choose a name for the package and put a package statement with that name at the top of every source file that contains the types (classes, interfaces, enumerations, and annotation types) that you want to include in the package.

The package statement (for example, package graphics;) must be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

Syntax: package package-Name;

```
Type Class Class-name
{

}
```

Accessing a Package:

The import statement can be used to search a list of packages for a particular class. The general form of import statement for searching a class is as follows:

Import package1 [.package2] [.package3] . Class name;

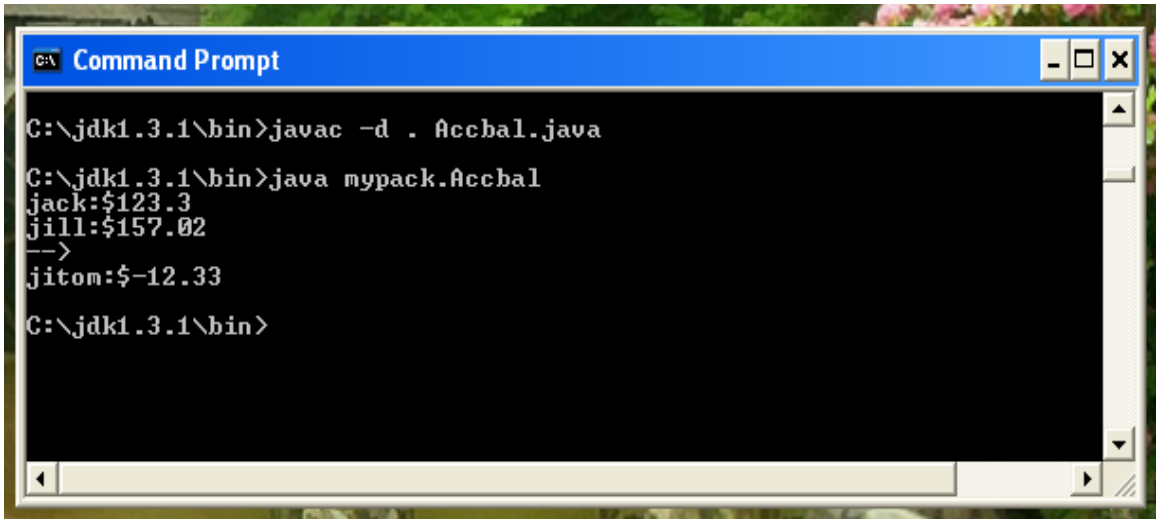
Here, package1 is the name of the top level package,, package2 is the name of the package that is inside the package, and so on. The following is an example of importing a particular class:

import firstPackage . secondPackage . MyClass ;

Algorithm:

1) a directory that contains one or more class files.
2) creating a folder
3) moving the class file to the new folder (or creating them there)
4) calling the package from any applications
5) the location of packages are defined by using the CLASSPATH environmental variable
6) each class in the package must have the following as its first line package <package name>;
7) The classes in the package are loaded into a Java application by using import <package name>.* ;

Output:



Conclusion:

Hence we studied how to implement Package in java.

8. To study exception handling in JAVA

Aim: Program to Program to Create user defined Exceptions

H/w, S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

An exception is a condition that is caused by run time error in the program. An exception is a point in the code where something out of the ordinary has happened and the regular flow of the program needs to be interrupted; an exception is not necessarily an error. For e.g. errors like divide by zero, ArrayIndexOutOfBoundsException etc. If java interpreter encounters such errors then it creates an exception object and throws it. If an exception object is not caught and handled properly then the interpreter will display an error Message. If we want program to continue with execution without an error message then we should try to catch the exception code. Them we should try to catch exception object thrown by error condition and then display the error message for taking corrective Action. This Procedure is known as Exception Handling.

Purpose of Exception Handling:

The Purpose of exception handling is to be able to define the regular flow of the program in part of the code without worrying about all the special cases. Then, in a separate block of code, you cover the exceptional cases.

It does the task like

- Find the problem (Hit the exception)
- Inform that the error is encountered (Throw exception)
- Run the error handling (exception Catch) code.
- Take corrective action.

Handling an Exceptions:

While programming what you do is write blocks of code that may generate exceptions inside try-catch blocks. You try the statements that generate the exceptions. Within your try block you are free to act as if nothing has or can go wrong. Then, within one or more catch blocks, you write the program logic that deals with all the special cases.

Syntax for Handling an Exceptions:

The basic concept of exception handling are throwing an exception and catching it . java uses the keyword Catch that catches the exception and throws by Try block. The catch block is added immediately after try block. Start a try clause:

```
try
{
  // Section of code which might fail
}
```

Multiple Catch Statements:

After the try section there must exist one or more catch statements to catch some or all of the exceptions that can happen within the try block. Exceptions that are not caught will be passed up to the next level, such as the function that called the one which threw the exception, and so on.

```
try
{
  // Section of code which might fail
}
catch (Exception1ThatCanHappen E)
{
  // things to do if this exception was thrown..
}
catch (Exception2ThatCanHappen E)
```

```
{
  // things to do if this exception was thrown..
}
```

The catch blocks work like a method definition. In which a single parameter is passed a s a reference to exception and is not caught, default exception handler will be caused the execution to terminate.

Using Finally statement:

Java support another statement called finally statement which is used to handle exception i. e not caught by any of the pervious catch statement.

Syntax:

```
try
{
  // Section of code which might fail
}

finally

{

}
catch (Exception1ThatCanHappen E)
{
  // things to do if this exception was thrown..
}
finally

{

}
```

Throwing our own Exceptions:

We can also like to throw our own exception, we can do this by using throw keyword as follows:

Syntax: Throw new Throwable_subclass;

Algorithm:

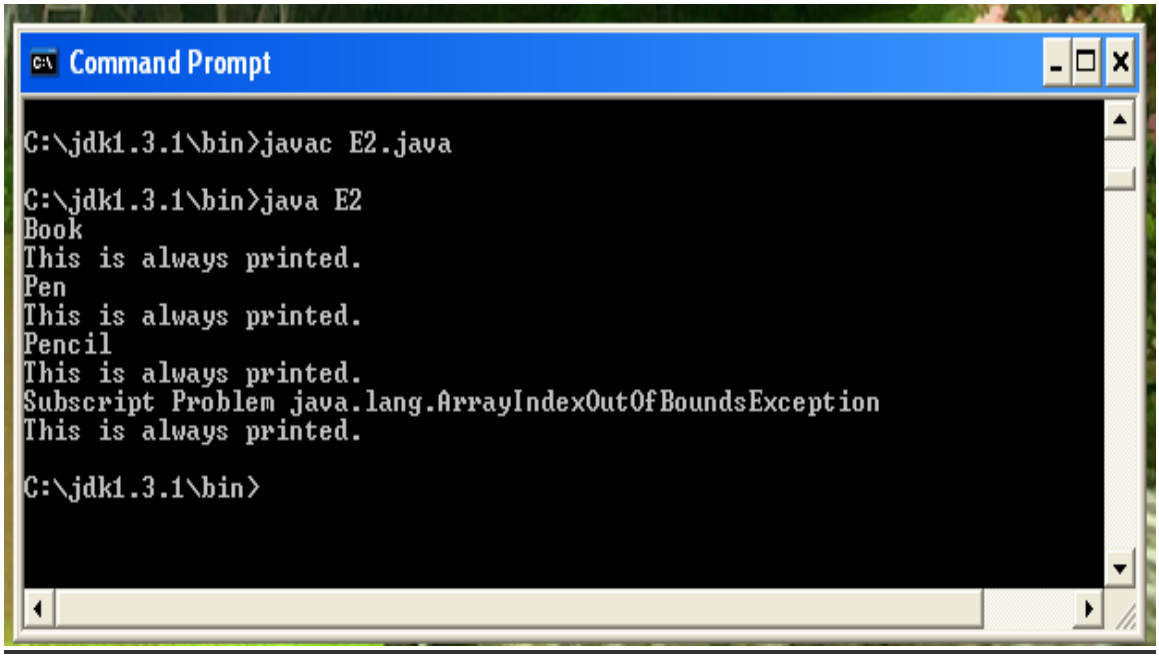
1)Define a class , define variables and members of class.

2)write down the code which might throw an exception at a time of program execution in try block which throws exceptions.

3)The user defined exception class should extend from Exception class.

4) Write down multiple catch statements for handling all exceptions thrown by try block.

Output:



Conclusion:

Hence we studied how to handle user defined Exceptions in java.

1. WAP for handling Arithmetic exception and Input mismatch Exceptions.
2. WAP to demonstrate the use of StringIndexOutOfBoundException
3. WAP for catching Exception using class Exception

   Define classes ExceptionA(Which inherits from class Exception) and ExceptionB(which inherits from class ExceptionA).In your program, create try blocks that throw exception of types Exception, ExceptionB, NullpointerException and IOException. All exception should be caught with catch blocks specifying type Exception.
4. Use inheritance to create an exception superclass(A) and two exception subclasses(A and B),where B inherits form A and C inherits from B. WAP to demonstrate that the catch block for type A catches the exceptions of type A and B
5. WAP that shows the order of catch block is important. If you try to catch a superclass exception type before a subclass type. The compiler should generate the errors.

9. Program For Applets in Java

Aim: Program For Drawing an different shapes in Java Applet
H/w,S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

Theory:

Applet: An applet is a special kind of Java program that a browser enabled with Java technology can download from the internet and run. An applet is typically embedded inside a web-page and runs in the context of the browser. An applet must be a subclass of the java.applet.Applet class, which provides the standard interface between the applet and the browser environment.

Life Cycle of an Applet:

Basically, there are four methods in the `Applet` class on which any applet is built.

- init: This method is intended for whatever initialization is needed for your applet. It is called after the param attributes of the applet tag.
- start: This method is automatically called after init method. It is also called whenever user returns to the page containing the applet after visiting other pages.
- stop: This method is automatically called whenever the user moves away from the page containing applets. You can use this method to stop an animation.
- destroy: This method is only called when the browser shuts down normally.

Thus, the applet can be initialized once and only once, started and stopped one or more times in its life, and destroyed once and only once.

After you enter the source code for Simple Applet, compile in the same way that you have been compiling programs. How ever, running Simple Applet involves a different process. In fact, there are two ways in which you can run an applet.

Executing the applet within a Java-compatible Web Browser.

Using an applet viewer, such as the standard SDK tool, applet viewer. An applet viewer executes your applet in a window. This s generally the fastest and easiest way to test your applet

Commonly used Methods for creating an Applets:

Drawing Lines :-

Lines are drawn by means of the **drawline()** method

Syntax: Void drawLine(int startX,int startY, int endX, int endY)

drawLine() displays a line in the current drawing color that begins at **startX, StartY** and ends at **endX, endY.**

Drawing Rectangles :-

The **drawrect()** and **fillRect()** methods display an outlined and filled rectangle, respectively **.**

Syntax: Void drawRect(int top, int left, int width, int height)

Void fillRect(int top, int left, int width, int height)

The upper-left corner of the rectangle is at top left. The dimensions of the Rectangle are specified by Width and Height.

Drawing Ellipse and Circles :-

To draw an Ellipse , use **DrawOval().** To fill an ellipse, use fillOval(). These methods are :

Syntax: Void drawOval(int top,int left,int width,int height)

Void fillOval(int top,int left,int width, int height)

The Ellipse is drawn within a bounding rectangle whose upper-left corner is specified by top, left and whose width and height are specified by width and height. To draw a circle, specify a square as the bounding rectangle.

Drawing Arcs :-

Arcs can be drawn with drawArc() and FillArc()

Syntax: Void drawArc(int top,int width, int height, int startAngle,int sweepAngle)

Void FillArc(int top,int left,int width,int height,int startAngle,int sweepAngle)

DrawString :

Inside paint ( ) is a call to drawString ( ), which is a member of the Graphics class. This method outputs a string beginning at the specified X,Y location. It has the following general form:

Syntax: Void drawstring (String message , int x , int y)

Here, message is the string to be output beginning at x,y. In a Java Window, the upper-left corner is location 0,0.

Algorithm: -

1) import applet packages in program e.g. import java.awt.Image

import java.applet.*;

import java.awt.*;

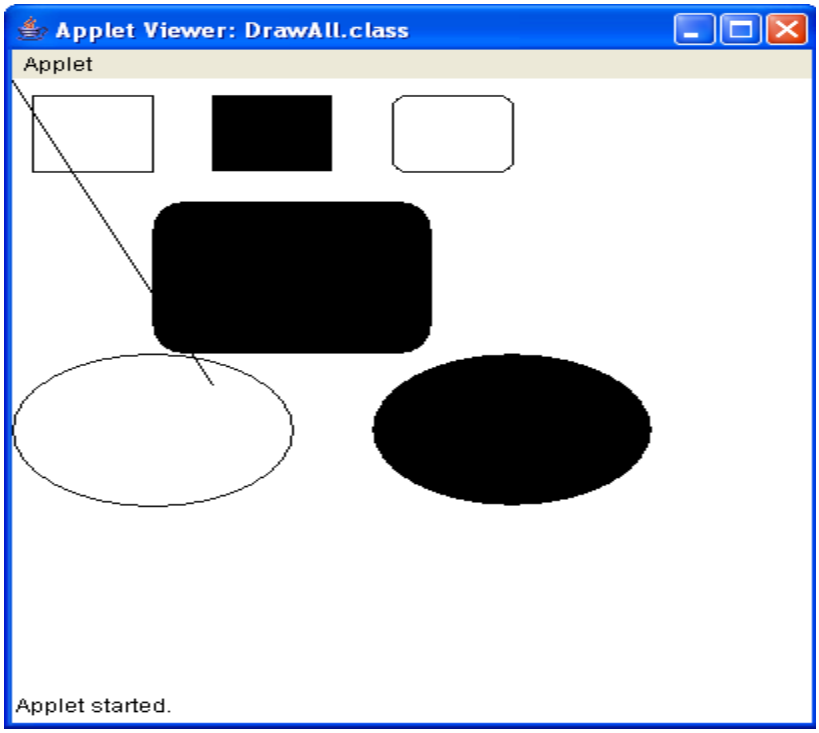
2) To draw images, import the class Image. Type this at the top of your code (not in your class)

3) Invoke the methods of classes with the Object.

4) Write applet code

5) See the O/P on Applet viewer

O***UT PUT : -***



Conclusion: Hence we studied how to implement Applets in java.

10. Program to retrieve data from Employee database

Aim: Program to retrieve data from Employee database
H/w, S/w Requirement: IBM-compatible 486 System, a hard drive, Min 8Mb memory, Win 98 S/w.

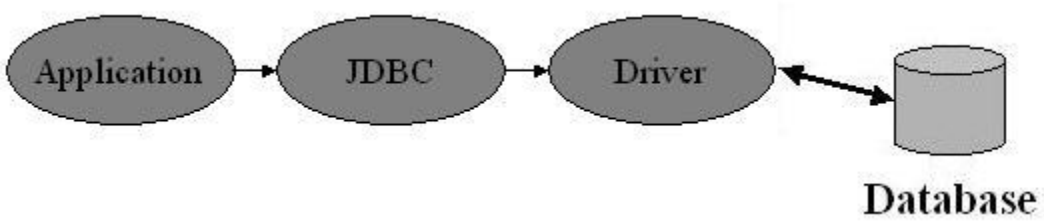
Theory:
A file system does not work well for data storage applications for e.g. Business applications so we required power of database. Java provide file system access with database connectivity called JDBC.

JDBC:

The JDBC (Java Database Connectivity) API defines interfaces and classes for writing database applications in Java by making database connections. Using JDBC, you can send SQL, PL/SQL statements to almost any relational database. JDBC is a Java API for executing SQL statements and supports basic SQL functionality. It provides RDBMS access by allowing you to embed SQL inside Java code. Because Java can run on a thin client, applets embedded in Web pages can contain downloadable JDBC code to enable remote database access.

We can also create a table, insert values into it, query the table, retrieve results, and update the table with the help of a JDBC Program example.
Although JDBC was designed specifically to provide a Java interface to relational databases, we may find that you need to write Java code to access non-relational databases as well.

JDBC Architecture



Java application calls the JDBC library. JDBC loads a driver that talks to the database. We can change database engines without changing database code.

JDBC Basics - Java Database Connectivity Steps

Before you can create a java jdbc connection to the database, you must first import the java.sql package.

import java.sql.*; The star (*) indicates that all of the classes in the package java.sql are to be imported.

1. Loading a database driver,

In this step of the jdbc connection process, we load the driver class by calling Class.forName() with the Driver class name as an argument. Once loaded, the Driver class creates an instance of itself. A client can connect to Database Server through JDBC Driver. Since most of the Database servers support ODBC driver therefore JDBC-ODBC Bridge driver is commonly used.

The return type of the Class.forName (String ClassName) method is “Class”. java.lang package.

Syntax:

```
try {
        Class.forName(”sun.jdbc.odbc.JdbcOdbcDriver”); //Or any other
driver
}
catch(Exception x){
        System.out.println( “Unable to load the driver class!” );
```

2.Creating aoracle jdbc Connection

The JDBC DriverManager class defines objects which can connect Java applications to a JDBC driver. DriverManager is considered the backbone of JDBC architecture. DriverManager class manages the JDBC drivers that are installed on the system. Its getConnection() method is used to establish a connection to a database. It uses a username, password, and a jdbc url to establish a connection to the database and returns a connection object. A jdbc Connection represents a session/connection with a specific database. Within the context of a Connection, SQL, PL/SQL statements are executed and results are returned. An application can have one or more connections with a single database, or it can have many connections with different databases. A Connection object provides metadata i.e. information about the database, tables, and fields. It also contains methods to deal with transactions.

3. Creating a jdbc Statement object,

Once a connection is obtained we can interact with the database. Connection interface defines methods for interacting with the database via the established connection. To execute SQL statements, you need to instantiate a

Statement object from your connection object by using the createStatement() method.

Statement statement = dbConnection.createStatement();

A statement object is used to send and execute SQL statements to a database.

4. Executing a SQL statement with the Statement object, and returning a jdbc resultSet.

Statement interface defines methods that are used to interact with database via the execution of SQL statements. The Statement class has three methods executeQuery(), executeUpdate(), and execute().

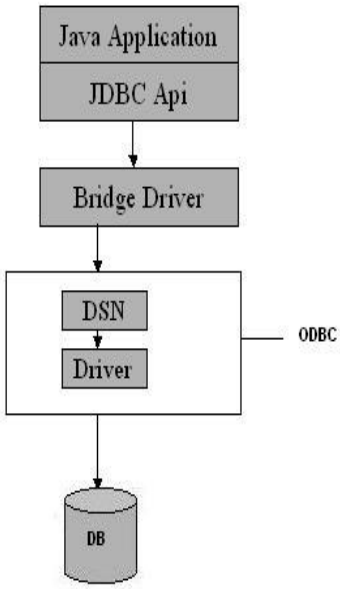
JDBC driversTypes:

JDBC drivers are divided into four types or levels. The **different types of jdbc drivers** are:

Type 1: JDBC-ODBC Bridge driver (Bridge)
Type 2: Native-API/partly Java driver (Native)
Type 3: All Java/Net-protocol driver (Middleware)
Type 4: All Java/Native-protocol driver (Pure)

Type 1 JDBC Driver: **JDBC-ODBC Bridge driver**

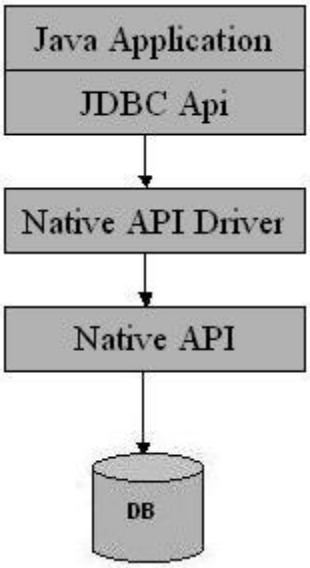
The Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge



Type 1: JDBC-ODBC Bridge

Native-API/partly Java driver

The distinctive characteristic of type 2 jdbc drivers are that Type 2 drivers convert JDBC calls into database-specific calls i.e. this driver is specific to a particular database. Some distinctive characteristic of type 2 jdbc drivers are shown below. Example: Oracle will have oracle native api.



Type 2: Native api/ Partly Java Driver

E.g.:

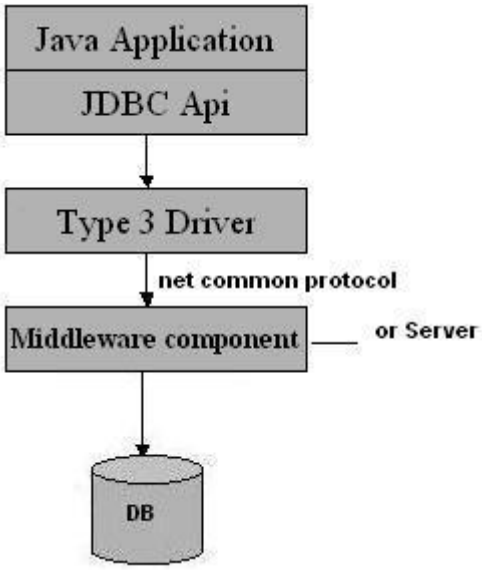
Orders Table:

```
CREATE TABLE Orders (
Prod_ID INTEGER,
ProductName VARCHAR(20),
Employee_ID INTEGER
);
```

Type 3 JDBC Driver

All Java/Net-protocol driver

Type 3 database requests are passed through the network to the middle-tier server. The middle-tier then translates the request to the database. If the middle-tier server can in turn use Type1, Type 2 or Type 4 drivers.

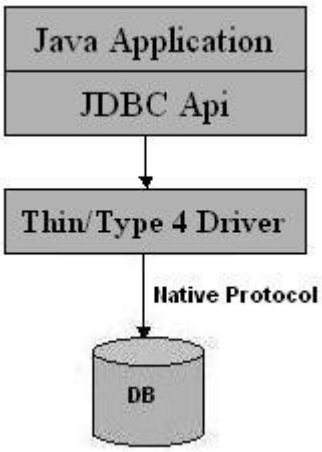


Type 3: All Java/ Net-Protocol Driver

Type 4 JDBC Driver

Native-protocol/all-Java driver

The Type 4 uses java networking libraries to communicate directly with the database server.



Type 4: Native-protocol/all-Java driver

Advantage

The JDBC-ODBC Bridge allows access to almost any database, since the database's ODBC drivers are already available.

Disadvantages

1. Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable.
   2. A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types.
   3. The client system requires the ODBC Installation to use the driver.
   4. Not good for the Web.

Algorithm:

1)Cerate a Table in Ms Access,save it in data base.

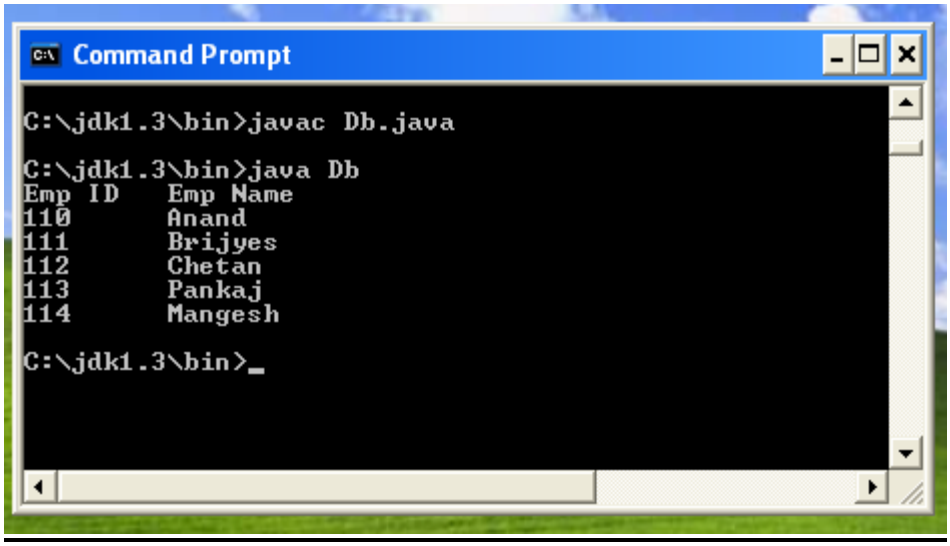
2)write java program, Import java Packages in program.

3) Define class establish ODBC connection with the help of java code.

4) Insert record in to database with the help of SQL queries.

5) Execute program, see the Op on console prompt.

Output:



Conclusion: Hence, we studied how to implement Jdbc

100 Possible Question on Java:

VIVA VOCE QUESTIONS

1) What is a method? And What is OOPS?
2) What is the signature of a method?
3) What is the difference between an instance variable and a class variable?
4) What is an abstract method?
5) What is an abstract class?
6) What is an object reference?
7) What is an exception?
8) Why does the compiler complain about Interrupted Exception when I try to use Thread's sleep method?
9) Why do methods have to declare the exceptions they can throw?
10) What is the difference between a runtime exception and a plain exception-why don't you runtime exceptions have to be declared?
11) What is an applet?
12) How do applets differ from applications?
13) Can I write Java code that works both as an applet and as a stand-alone application?
14) What is the difference between an application, an applet, and a
15) Several applet methods seem special, in that I need to define them even if my own code doesn't invoke them--what are the methods, and when (and by whom) are they invoked?
16) Should applets have constructors?
17) How can my applet tell when a user leaves or returns to the web page containing my applet?
18) How do I read number information from my applet's parameters, given that Applet's getParameter method returns a String?
19) When I subclass Applet, why should I put setup code in the init() method? Why not just a constructor for my class?
20) Can I use an http URL to write to a file on the server from an applet?
21) Can applets launch programs on the server?
22) Can applets launch programs on the client?
23) How do you do file I/O from an applet?
24) How do I access remote machine's file system through Java Applet?
25) What is a thread?
26) How do I create a thread and start it running?

27) How many threads can I create?
28) How does Thread's stop method work--can I restart a stopped thread?
29) If I create a thread, and then null out the reference to it, what happens to the thread? Does it get interrupted or what?
30) How should I stop a thread so that I can start a new thread later in its place?
31) How do I specify pause times in my program?
32) Why is thread synchronization important for multithreaded programs?
33) What is a monitor?
34) How does the synchronized keyword work?
35) What objects do static synchronized methods use for locking?
36) How do the wait and notifyAll/notify methods enable cooperation between threads?
37) How do I achieve the effect of condition variables if the Java platform provides me with only wait and notifyAll/notify methods?
38) How do I make one thread wait for one or more other threads to finish?
39) What do I use the yield method for?
40) Does the Java Virtual Machine protect me against deadlocks?
41) I have several worker threads. I want my main thread to wait for any of them to complete, and take action as soon as any of them completes. I don't know which will complete soonest, so I can't just call Thread.join on that one. How do I do it?
42) How do I do keyboard (interactive) I/O in Java?
43) Is there a way to read a char from the keyboard without having to type carriage-return?
44) How do I read a line of input at a time?
45) How do I read input from the user (or send output) analogous to using standard input and standard output in C or C++?
46) Is there a standard way to read in int, long, float, and double values from a string representation?
47) How do I read a String/int/boolean/etc from the keyboard?
48) I try to use "int i = System.in.read();" to read in an int from the standard input stream. It doesn't work. Why?
49) I use the following to read an int. It does not work. Why?
50) I'm trying to read in a character from a text file using the DataInputStream's readChar() method. However, when I print it out, I get?'s.

51) Why do I get garbage results when I use DataInputStream's readInt or readFloat methods to read in a number from an input string?
52) How do I read data from a file?
53) How do I write data to a file?
54) How do I append data to a file?
55) When do I need to flush an output stream?
56) Why do I see no output when I run a simple process, such as r.exec("/usr/bin/ls")?
57) Can I write objects to and read objects from a file or other stream?
58) How do I format numbers like C's printf()?
59) How do I do file I/O in an applet?
60) How do I do I/O to the serial port on my computer
61) How do I do formatted I/O like printf and scanf in C/C++?
62) How do I read a file containing ASCII numbers?
63) Why do I have trouble with System.out.println()?
64) How do I write to the serial port on my PC using Java?
65) Is it possible to lock a file using Java ?
66) How do I make the keyboard beep in Java?
67) How do I make I/O faster? My file copy program is slow.
68) How do I do formatted I/O of floating point numbers?
69) How do I read numbers in exponential format in Java?
70) How do I delete a directory in Java? 71). How do I tell how much disk space is free in Java?
71) How do I get a directory listing of the root directory C:\ on a PC?
72) I did a read from a Buffered stream, and I got fewer bytes than I specified
73) How do I redirect the System.err stream to a file?
74) What are the values for the Unicode encoding schemes?
75) How do I print from a Java program?
76) What are the properties that can be used in a PrintJob?
77) How do I get Java talking to a Microsoft Access database?
78) How do I do I/O redirection in Java using exec()?
79) What is the signature of a method?
81) How do I create an instance of a class?
82) Why do I get the java.lang.UnsatisfiedLinkError when I run my Java program containing Native Method invocations?
83) Given a method that doesn't declare any exceptions, can I override that method in a subclass to throw an exception?

84) What's the difference between a runtime exception and a plain exception-why don't runtime exceptions have to be declared?
85) Why do methods have to declare the exceptions they can throw?
86) Why does the compiler complain about Interrupted Exception when I try to use Thread's sleep method?
87) What is an exception?
88) I can't seem to change the value of an Integer object once created.
89) How can I safely store particular types in general containers?
90) Why is the String class final? I often want to override it.
91) How do static methods interact with inheritance?
92) Where can I find examples of the use of the Java class libraries?
93) How can I find the format of a .class file/any file?
94) What are "class literals"?
95) What is an object reference?
96) What does it mean that a method or class is abstract?
97) What is an abstract class?
98) What is an abstract method?
99) How do I create an instance of a class?
100) what is an object reference?

Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become popular amongst the students. It is a wrong approach or concept to award the students by way of easy marking to get cheap popularity among the students to which they do not deserve. It is a primary responsibility of the teacher that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with unjust circumstances.

The assessment is done according to the directives of the Principal/ Vice-Principal/ Dean Academics.