

Introduction to Java

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes

Table of Contents

<i>Document Revision History</i>	<i>1</i>
<i>Table of Contents</i>	<i>3</i>
<i>Getting Started</i>	<i>4</i>
<i>Overview.....</i>	<i>4</i>
<i>Setup Checklist for Core Java.....</i>	<i>4</i>
<i>Instructions</i>	<i>4</i>
<i>Learning More (Bibliography if applicable).....</i>	<i>4</i>
<i>Lab 1. Working with Eclipse.....</i>	<i>5</i>
1.2: Create Java Project.....	10
<i>Lab 2. Language Fundamentals – Part I</i>	<i>15</i>
2.1: Write a program that displays “Hello World”.....	15
<<TO DO>>.....	20
2.2: Working with Classes and Objects.....	20
Write a program to create a Date class and UseDate class which instantiates the Date class.	20
<<TO DO>>.....	21
<<TO DO>>.....	24
<i>Lab 3. Language Fundamentals – Part II</i>	<i>26</i>
3.1: Working with Arrays.....	26
<<TO DO>>.....	26
3.2: Working with enum types	28
<<TO DO>>.....	28
3.3: Using Static variables and methods.	29
3.4: Working with Array of Objects	30
Write a program to create a Salary class.	30
<<TO DO>>.....	31
3.5: Inheritance.....	33
<<TO DO>>.....	33
<<TO DO>>.....	34

Getting Started

Overview

This lab book is a guided tour for learning Core Java version 5.0. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given which will expose you to working with Java applications. Creating Classes, Interfaces, GUI-based applications and applets, Database application, networking and mailing concepts and logging in.

Setup Checklist for Core Java

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 6.0 or higher
- MS-Access/Connectivity to Oracle database
- Apache Tomcat Version 5.0.

Please ensure that the following is done:

- A text editor like Notepad or Eclipse is installed.
- JDK 1.5 is installed. (This path is henceforth referred as <java_install_dir>)

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory java_assgn. For each lab exercise create a directory as lab <lab number>.

Learning More (Bibliography if applicable)

- Java Programming Advanced Topics: Source Book by Wigglesworth and McMilan
- Java: How to Program by Dietel n Dietel
- Java2: Beginning Java2 JDK 5 by Doug Lowe, Joel Murach and Andrea Steelman.

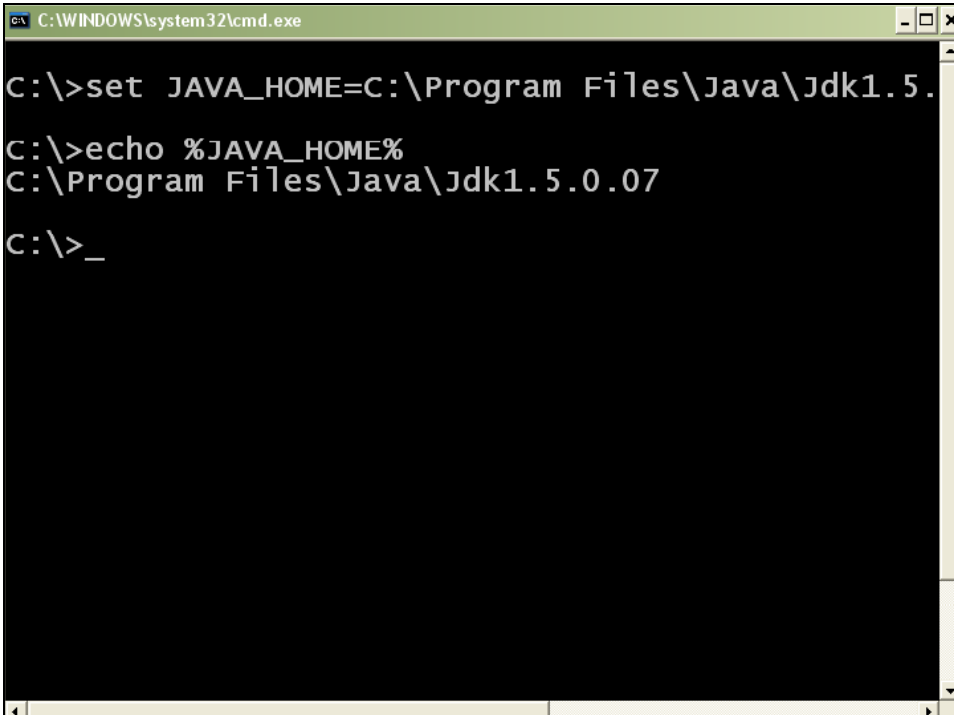
Lab 1. Working with Eclipse

Goals	<ul style="list-style-type: none">Learn and understand the process of:<ul style="list-style-type: none">Setting environment variablesCreating a simple Java Project using Eclipse 3.0
Time	20 minutes

1.1: Setting environment variables from CommandLine.Solution:

Step 1: Set JAVA_HOME to Jdk1.5 using the following command:

- Set **JAVA_HOME=C:\Program Files\Java\Jdk1.5.0.07**



```
C:\WINDOWS\system32\cmd.exe

C:\>set JAVA_HOME=C:\Program Files\Java\Jdk1.5.0.07
C:\>echo %JAVA_HOME%
C:\Program Files\Java\Jdk1.5.0.07
C:\>_
```

Figure 1: Java program

Step 2: Set PATH environment variable:

- Set **PATH=%PATH%;%JAVA_HOME%\bin;**

Step 3: Set your current working directory and set classpath.

- Set CLASSPATH=.

Note: Classpath searches for the classes required to execute the command. Hence it must be set to the directory containing the class files or the names of the jars delimited by ;

For example: C:\Test\myproject\Class;ant.jar



Alternatively follow the following steps for setting the environment variables

Alternate approach:

Step 1: Right click **My Computers**, and select **Properties** → **Environment Variables**.

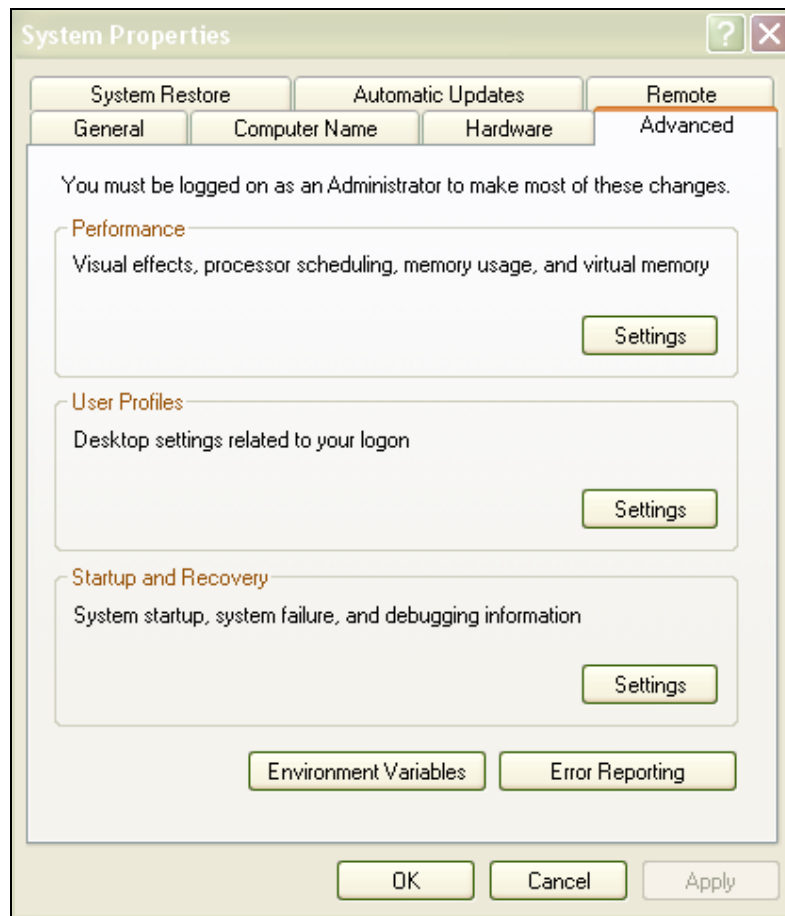


Figure 2: System Properties

Step 2: Click **Environment Variables**. The Environment Variables window will be displayed.

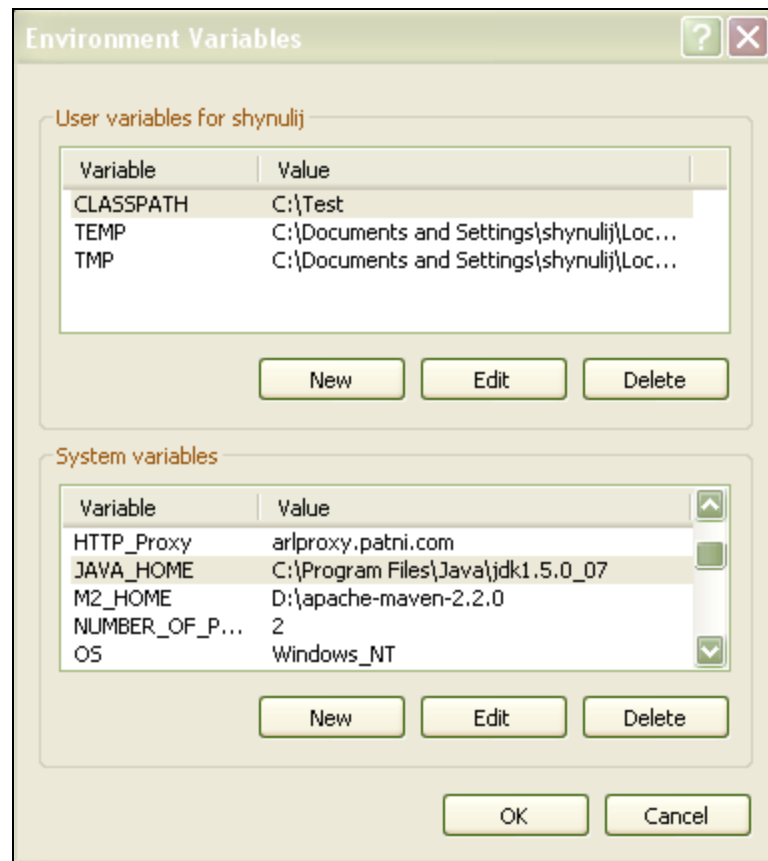


Figure 3: Environment Variables

Step 3: Click **JAVA_HOME** System Variable if it already exists, or create a new one and set the path of JDK1.5 as shown in the figure.

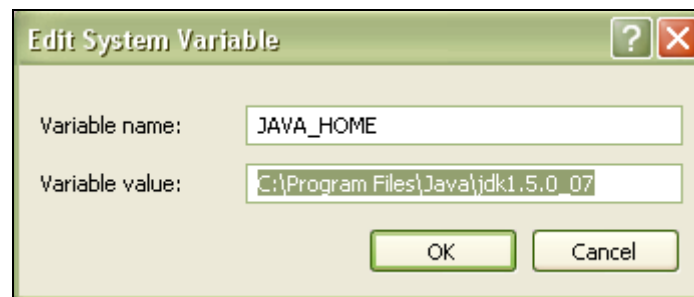


Figure 4: Edit System Variable

Step 4: Click **PATH** System Variable and set it as **%PATH%;%JAVA_HOME%\bin**.

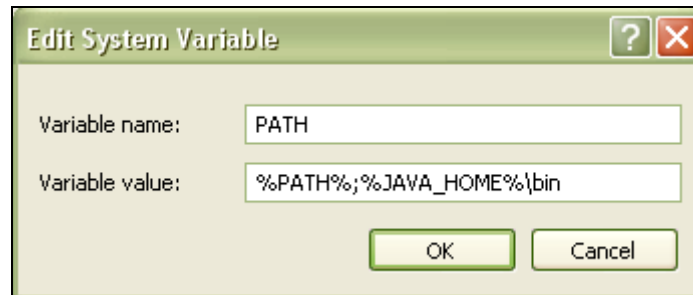


Figure 5: Edit System Variable

Step 5: Set **CLASSPATH** to your working directory in the **User Variables** tab.

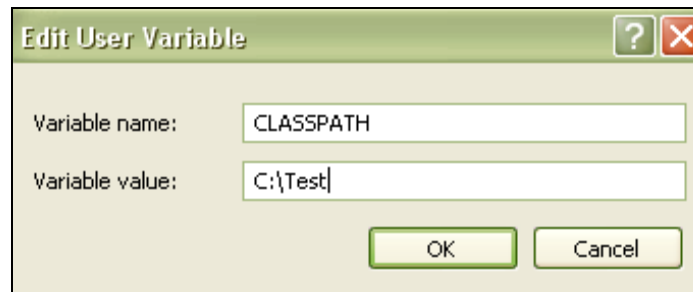


Figure 6: Edit User Variable

1.2: Create Java Project

Create a simple java project named 'myproject'..

Solution:

Step 1: Open eclipse3.3.

Step 2: Select **File** → **New** → **Project** → **Java project**.

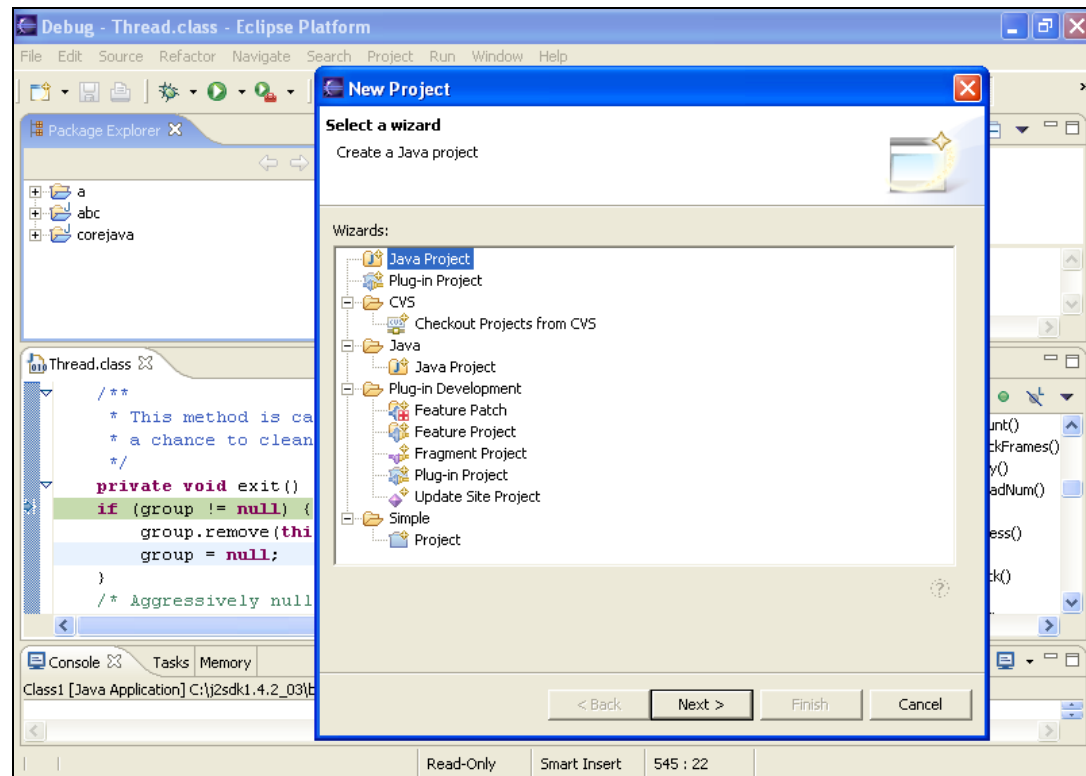


Figure 7: Select Wizard

Step 3: Click **Next** and provide name for the project.

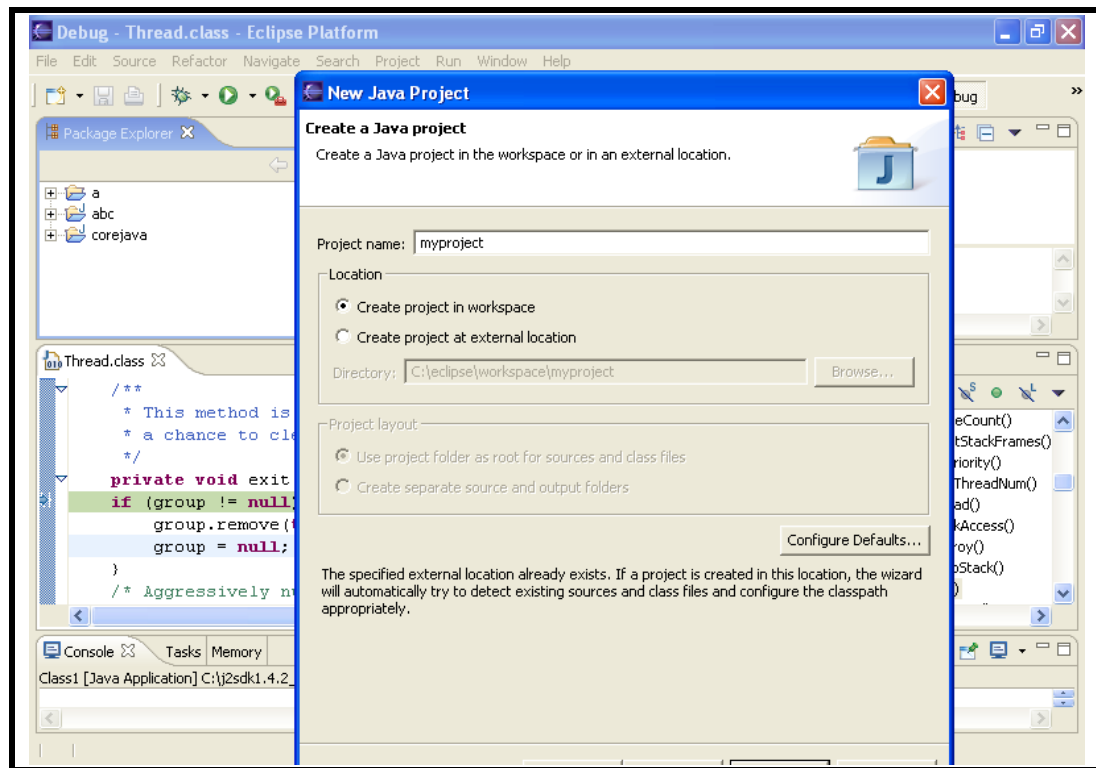


Figure 8: New Java Project

Step 4: Click **Next** and select build options for the project.

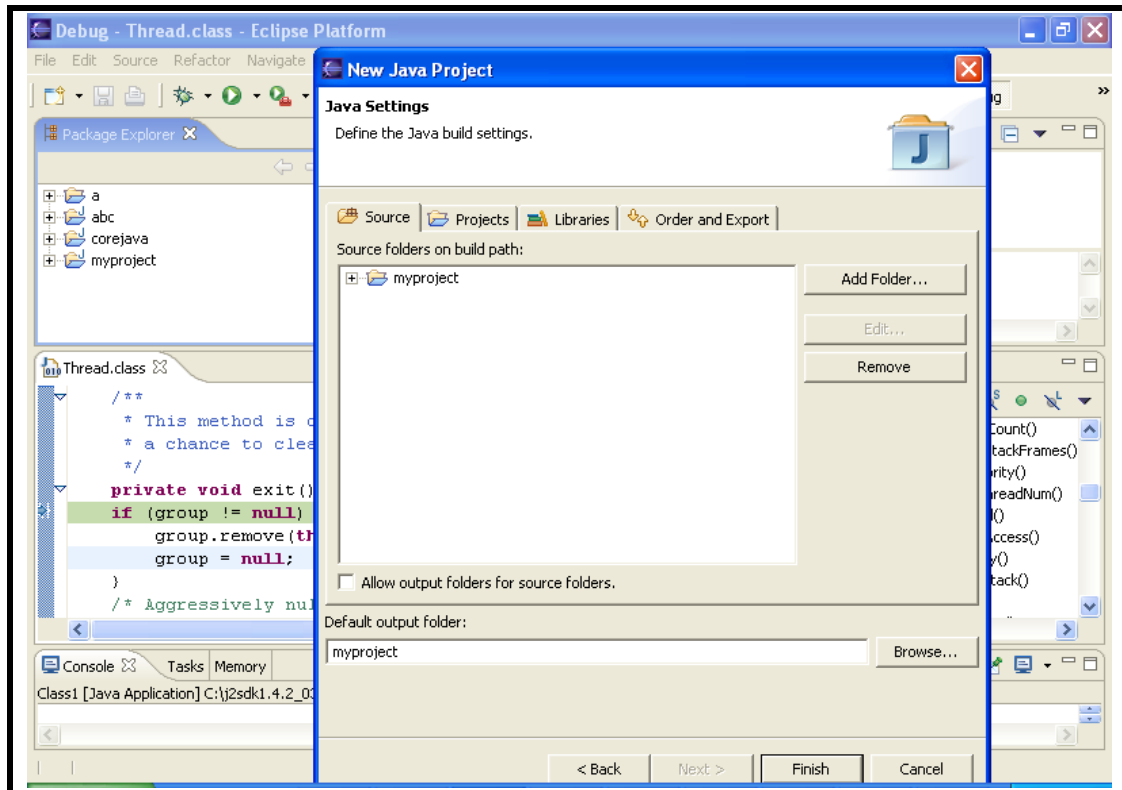


Figure 9: Java Settings

Step 5: Click **Finish** to complete the project creation.

Step 6: Right-click **myproject**, and select resource type that has to be created.

For example: Class if it is a Java class

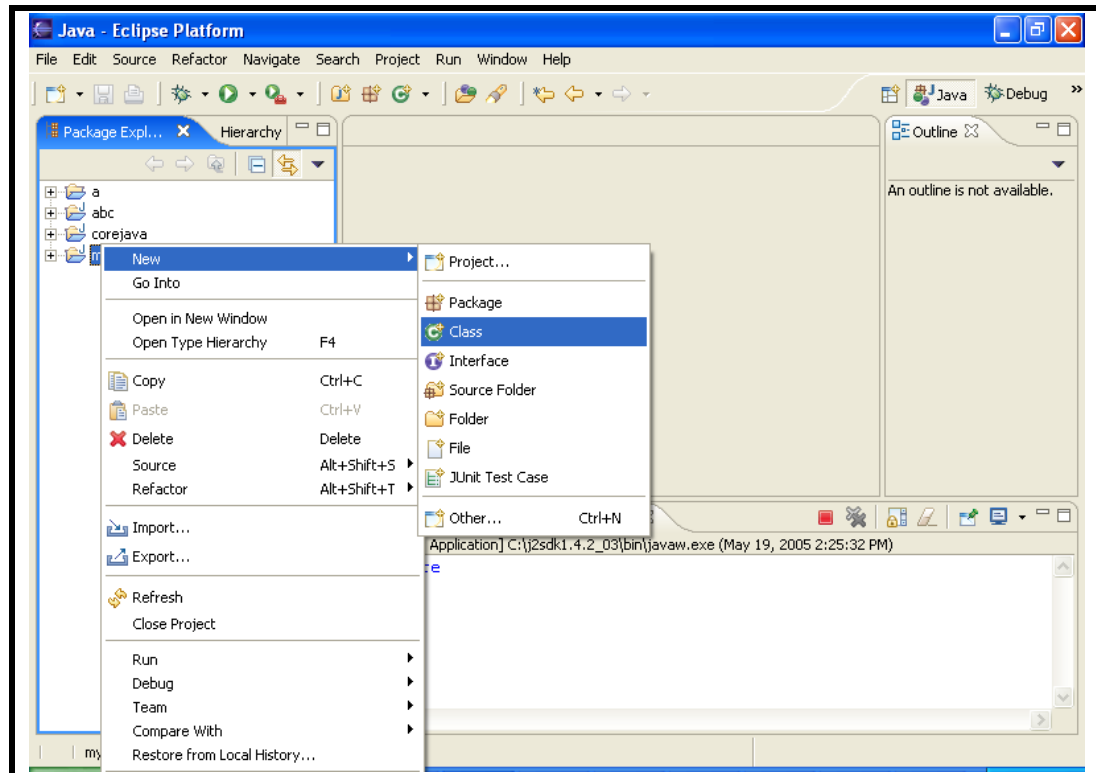


Figure 10: Select Resource

Step 7: Provide name and other details for the class, and click **Finish**.

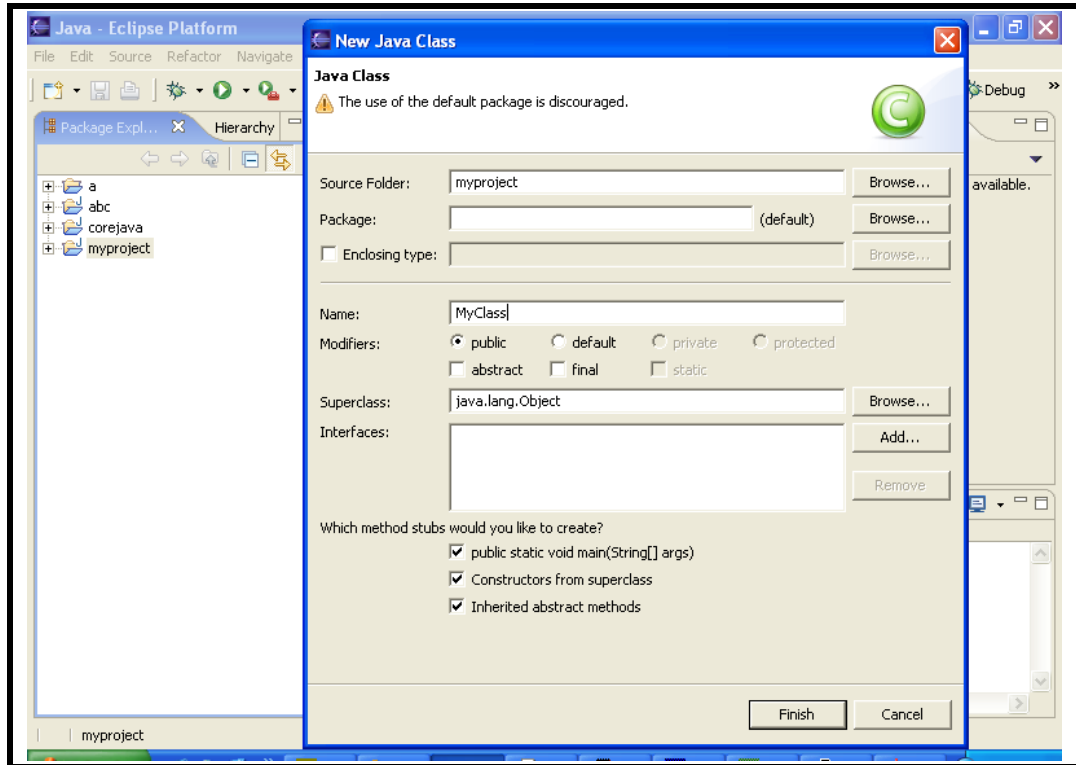


Figure 11: Java Class

This will open **MyClass.java** in the editor, with ready skeleton for the class, default constructor, **main()** method, and necessary **javadoc** comments.

To run this class, select **Run** from toolbar, or select **Run As → Java application**. Alternatively, you can select **Run..** and you will be guided through a wizard, for the selection of class containing **main()** method.

Console window will show the output.

Lab 2. Language Fundamentals – Part I

Goals	<ul style="list-style-type: none"> At the end of this lab session, you will be able to: <ul style="list-style-type: none"> Write a Java program that displays Hello world Working with Conditional Statements Create Classes and Objects
Time	1hr 30 min

2.1: Write a program that displays “Hello World”

Solution:

Step 1: Right click the Project **myproject**, and select **New → Class**.

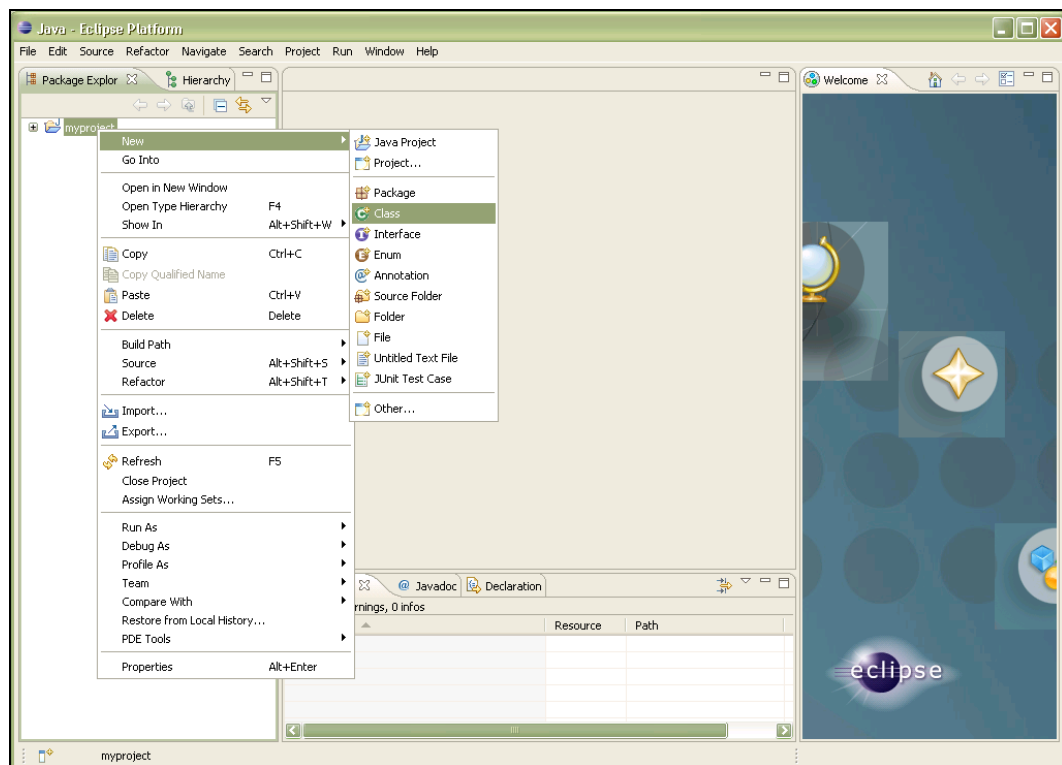


Figure 12: Creating a New Class

Step 2: In the **New Java Class** dialog, key in the name of the class as “*HelloWorld*”.

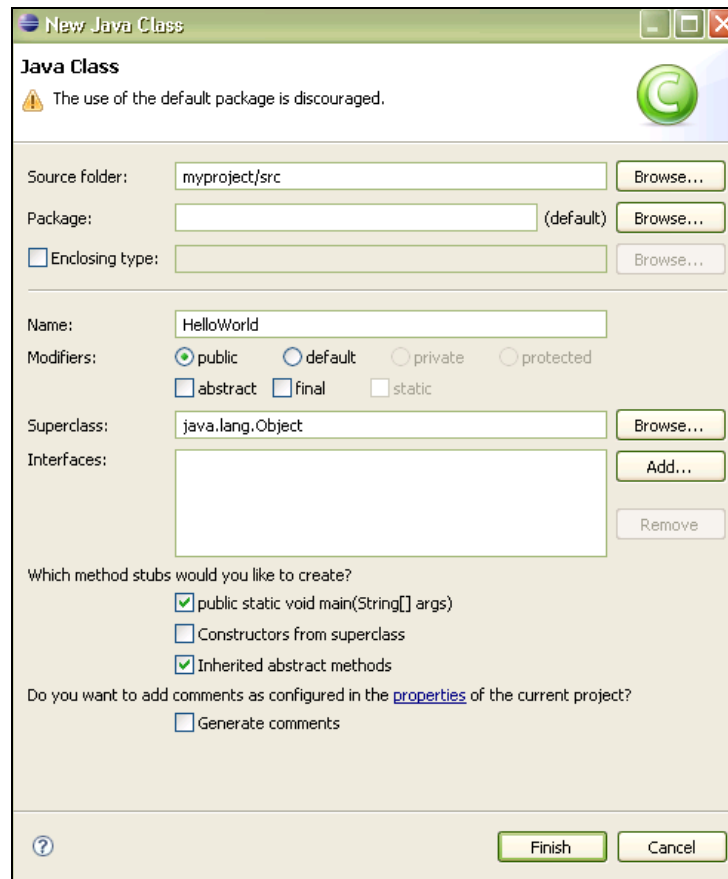


Figure 13: New Java Class dialog box

Step 3: Click **Finish**. The code-window will be displayed before you.

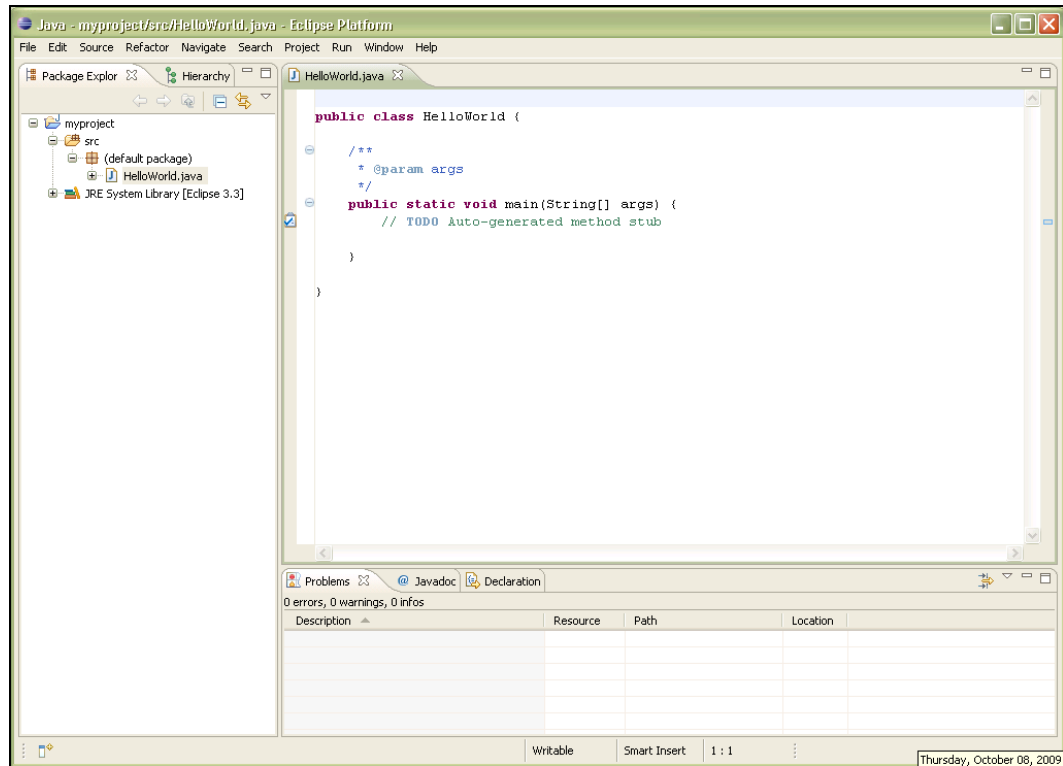


Figure 14: Code window

Step 4: Type the following code in the window and save the changes.

```
class HelloWorld {  
    /*This is a block comment */  
    public static void main(String args[]) {  
        System.out.println("Hello World");  
    } //main ends.  
} //class ends
```

Example 1: HelloWorld.java

Step 5: Run the code by right clicking in the code window, and selecting **Run As → Java Application**.

Alternatively, from the main menu, select **Run → Run As → Java Application**.

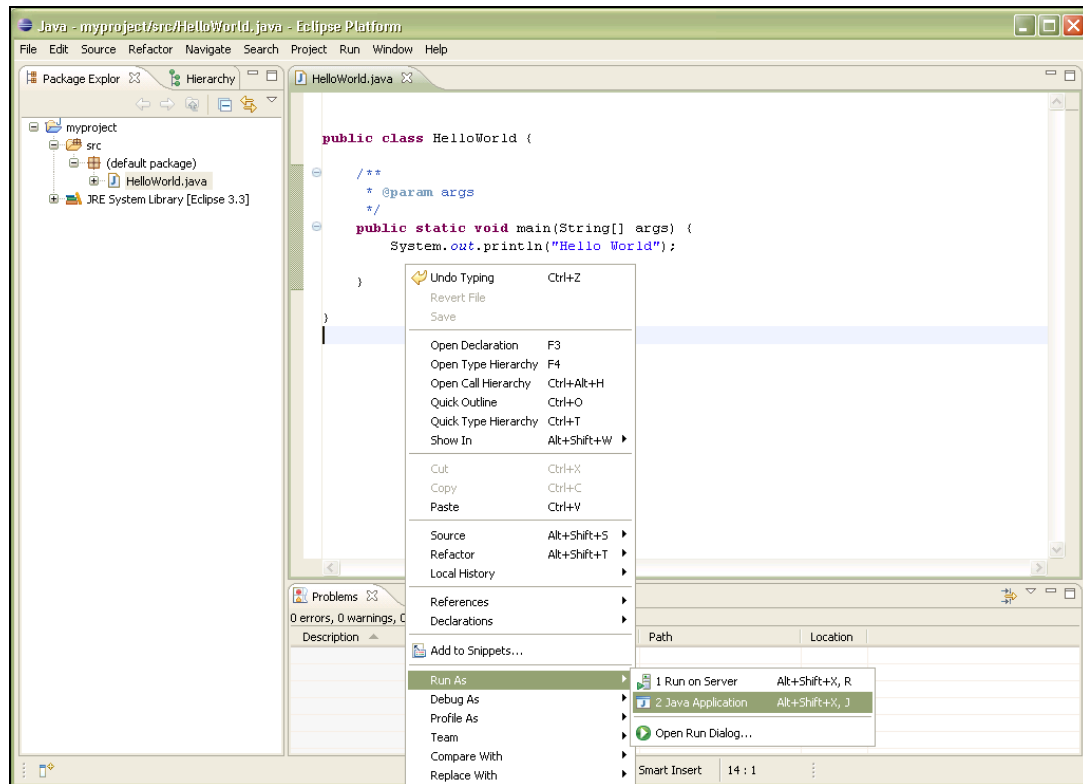


Figure 15: Selecting the Java Application

Step 6: The code will be executed and the output will be produced in the console window.

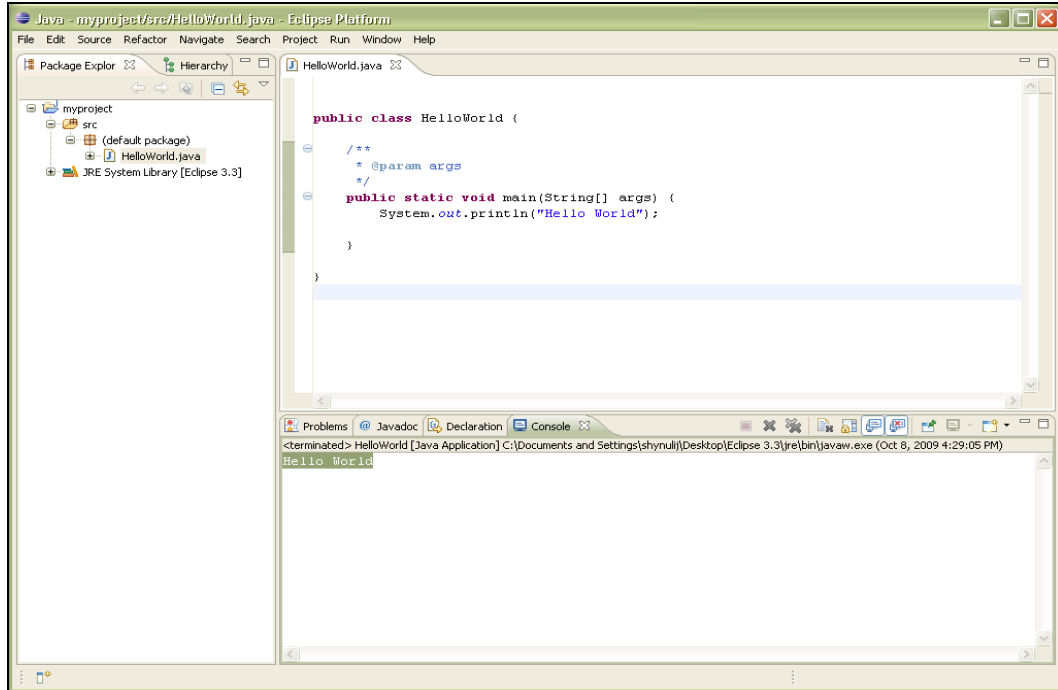


Figure 16: Console window



Alternatively to execute the same on the command prompt follow the steps given below:

Step 1: Open a notepad and type the code for HelloWorld.java.

Step 2: Open the command prompt and set your working directory.

Step 3: Compile the program by typing the following command:

Command: javac HelloWorld.java

Step 4: After successful compilation let us now execute the program using the following command:

Command: java HelloWorld

Step 5: You should get the output as shown below:

Hello World

<<TO DO>>

Assignment-1: Modify the **HelloWorld.java** program. Accept an integer value at command line and print the message that many number of times.

For example: c:\>HelloWorld 2 should print message 2 times.

(Hint: Use the **args[]** parameter of **main()**)

2.2: Working with Classes and Objects

Write a program to create a **Date** class and **UseDate** class which instantiates the **Date** class.

Solution:

Step 1: Type the following code in a text editor and editor and save the file with the name "**UseDate.java**".

```
class Date {
    int intDay, intMonth, intYear;

    Date(int intDay, int intMonth, int intYear) // Constructor
    {
        this.intDay = intDay;
        this.intMonth = intMonth;
        this.intYear = intYear;
    }

    void setDay(int intDay) // setter method for day
    {
        this.intDay = intDay;
    }

    int getDay( ) // getter method for month
    {
        return this.intDay;
    }

    void setMonth(int intMonth)
    {
        this.intMonth = intMonth;
    }

    int getMonth( )
```

```
{
    return this.intMonth;
}

void setYear(int intYear)
{
    this.intYear=intYear;
}

int getYear( )
{
    return this.intYear;
}

public String toString() //converts date obj to string.
{
    return "Date is "+intDay+"/"+intMonth+"/"+intYear;
}

} // Date class

class UseDate
{
    public static void main(String[] args)
    {
        Date d = new Date(9,5,2011);
        System.out.println(d); //invokes toString() method
    }
} //class ends
```

Example 2: UseDate.java

Step 2: Execute the class. You should get the output as follows:

Output: Date is 9/5/2011

<<TO DO>>

Assignment-3: Implement the following in the above date class

- a. Write methods to incorporate validation checks. (For example: day should be between 1 & 31, Year must be less than 1984).
- b. Call the validate methods in main. If the validation fails, restore default date.

Assignment-4: Create a class **Staff** that has members as specified below. Implement the **setter** and **getter** methods for the data members. Make the changes as required and save the file as **Staff.java**.

```
public class Staff
{
    private int staffCode;    // Employee Code
    private String staffName; // Name
    private String designation; // Designation
    private int age;          // Age
    private Date dateOfBirth; // Date of Birth (Create an object of the Date
class given in the Example 2.2 or make use of Calendar object)

    public Staff()    // Constructor
    {
        // To Do: Initialize data members
    }

    public Staff(int staffCode, String staffName)
    {
        // To Do: Initialize data members based on the paramters
    }

    public Staff(int staffCode, String staffName,String designation, int age)
    {
        // To Do: Initialize data members based on the parameters
    }

    // setter methods
    void setStaffCode(int staffCode)
    {
        // To Do: Setter method for staff code
    }

    public void setStaffName(String staffName)
    {
        // To Do: Setter method for staffName
    }

    void setDesignation(String designation)
    {
        // To Do: Setter method for designation
    }

    void setAge(int age)
    {
        // To Do: Setter method for age
    }
}
```

```
void setDateOfBirth(Date birthdate)
{
    // To Do: Setter method for date of birth
}

// getter methods
int getStaffCode()
{
    // To Do: getter method for staffCode
}

String getStaffName()
{
    // To Do: getter method for staffName
}

String getDesignation()
{
    // To Do: getter method for designation
}

int getAge()
{
    // To Do: getter method for Age
}

Date getDateOfBirth()
{
    // To Do: getter method for Salary
}

// prints the staff details on the screen
public void displayDetails()
{
    System.out.println("The staff code is "+ staffCode);
    // To Do: Print the other members in the same format
}

} // end of class Employee
Class StaffApplication
{
    public static void main(String[] args)
    {
        Staff staff = new Staff( );
        staff.displayDetails( );
    }
}
```

Example 3: Sample code

Hint : click generate setter and getter from source menu for setter and getter methods

Now, override the **toString()** method of **Object** class to return employeeCode and employeeName.

```
public String toString(){  
    // code to return a string equivalent of Employee object  
}
```

Example 4: Sample code

Change StaffApplication class as:

```
class StaffApplication  
{  
    public static void main(String[] args)  
    {  
        Staff staff = new Staff( );  
        System.out.println("Staff details: " + staff);  
    }  
}
```

Example 5: Sample code

<<TO DO>>

Assignment-5: Complete the following program. **Account.java** contains a partial definition for a class representing a bank account. Place appropriate methods to perform the tasks specified in **ManageAccounts.java**.

For withdrawal check if the balance is sufficient or print appropriate message and give three more chances to the user if the user wishes so.

```
public class Account  
{  
    private double balance;  
    private String name;  
    private long accountNumber;  
  
    public Account(double initBal, String name)  
    {  
        balance = initBal;  
        //Initialize name  
    }  
}
```



```
        accountNumber =getAccountNnumber();

//account number to be generated randomly and make sure that it is a unique number.
    }

    public String toString()
    {
        // Returns a string containing the name, account number, and balance.
    }

    public void chargeFee()
    {
        // Deducts $5 as bank charges
    }
    .....
}
public class ManageAccounts
{
    public static void main(String[] args)
    {
        Account account1, account2;
        account1 = new Account(1000, "John", 1111);

        //create account2 of Amith with $500
        //deposit $100 to Amith account
        //print Amith's new balance
        //withdraw $5000 from John's account
        //print John's new balance
        //charge fees to both accounts
        //change the name on Amith's account to John
        //print summary for both accounts
        //Test the code exhaustively.....
    }
}
```

Example 6: Sample code

Lab 3. Language Fundamentals – Part II

Goals	<ul style="list-style-type: none">• At the end of this lab session, you will be able to:<ul style="list-style-type: none">○ Use Arrays○ Working with array of objects○ Reuse classes through Inheritance○ Understand static method and variable and use it○ Work with strings
Time	2hr 30 min

3.1: Working with Arrays

<<TO DO>>

Assignment-1: Create a class **ArrayDemo**, which behaves like a wrapper around an integer array. The class should have methods to create array, add elements into it, display contents of array, search for an element in the array. Minimum size of the array should be 10.

```
class ArrayDemo{  
    int contents []    // declare array.  
    ArrayDemo (int){  
        // Create array of size <int>  
    }  
  
    void populateContents(){  
        // Populate the array  
    }  
  
    void showContents(){  
        // Display contents of array.  
    }  
  
    int searchElement(int searchElement){  
        // search for element; if found, return its index location, else return -1.  
    }  
}
```

Class UseArray

```
{  
    public static void main(String[] args)  
    {  
        ArrayDemo arrayDemo = new ArrayDemo( );  
        // call to the methods in the class.  
    }  
}
```

Example 7: Sample code

Assignment-2:

Write a java program which will take a list of values from command line and sort them using the following algorithm

- Bubble Sort
- Selection Sort

Example java SortProgram 12 4 15 9 89 90

Should display the output as 4 9 12 15 89 90

Assignment-3:

Write a java code called CurrencyConverter, which will convert Indian Rupees (INR) to

1. GBP
2. US dollars
3. Yen

[Hint: Store all conversion rates into an array]

Assignment-4: Write the program which creates an object of class **Product** that contains details of a product. Create another class called UseProduct to do the following

- a. Store 20 products
- b. Display the product which has the maximum price

```
class Product{  
    int productId;  
    String description;  
    double price;  
    int unit;
```

```
Product () {...} //default constructor

Product (int ProdID, String Descr, double Price, int Unit)    //constructor with 3 args

public int getId(){
// return the product id
}

public String getDescription(){
// returns the description of product
}

public double getPrice(){
// returns the price
}

public int getUnit(){
// returns the unit
}

Class UseProduct
{
    public static void main(String[] args)
    {
        Product product = new Product( );
        //.....

    }}
}
```

Example 8: Sample code

3.2: Working with enum types

<<TO DO>>

Assignment-5: Create an **enum** type **ArithmeticOperation** which take two numeric arguments as command line arguments, and perform the four basic arithmetic calculations PLUS, MINUS, MULTIPLY, and DIVIDE.

```
public enum ArithmeticOperation { PLUS, MINUS, TIMES, DIVIDE;    double
eval(double x, double y){ // Do arithmetic op represented by the constants}
```

```
public static void main(String args[]) {    double x =  
Double.parseDouble(args[0]);  
    double y = Double.parseDouble(args[1]);  
    // Execute the enum for Loop to perform the operations one by one and print out  
    the results on the console.  
    }  
}
```

Example 9: Sample code

3.3: Using Static variables and methods.

Solution:

Step 1: Type the following code:

```
public class Demo {  
    public Demo() {}  
  
    // Create static variable  
    static int staticNumber = 0;  
  
    // Create static method  
    static void staticMethod(int formalStaticNumber) {  
        System.out.println("staticMethod("+formalStaticNumber +") entered");  
    }  
  
    // Create static block  
    static {  
        System.out.println("Entered :static block , staticNumber = " + staticNumber);  
        staticNumber += 1;  
        System.out.println("Exiting: static block, staticNumber = " + staticNumber);  
    }  
} // class ends  
  
class StaticDemo {  
    public static void main(String[] args) {  
  
        // Access a static variable of Demo class.  
        System.out.println("Demo. staticNumber = " + Demo. staticNumber);  
  
        // Invoke a static method of Demo class  
        Demo.staticMethod(5);  
  
        // The static variable can be accessed from an object instance.  
        Demo demo1 = new Demo();
```

```
System.out.println("demo1.staticNumber = " + demo1. staticNumber);

// The static method can be invoked from an object instance.
demo1.staticMethod(0);

// The same static variable can be accessed from multiple instances.
Demo demo2 = new Demo();
System.out.println("demo2.staticNumber = " + demo2. staticNumber);
demo1.staticNumber += 3;
System.out.println("Demo. staticNumber = " + Demo. staticNumber);
System.out.println("demo1.staticNumber = " + demo1. staticNumber);
System.out.println("demo2. staticNumber = " + demo2. staticNumber);

}
} //class ends
```

Example 10: StaticDemo.java

Step 2: Execute it. You should get the output as shown below:

```
Entered :static block , staticNumber = 0
Exiting: static block, staticNumber = 1
Demo. staticNumber = 1
staticMethod(5) entered
demo1.staticNumber = 1
staticMethod(0) entered
demo2.staticNumber = 1
Demo. staticNumber = 4
demo1.staticNumber = 4
demo2. staticNumber = 4
```

3.4: Working with Array of Objects

Write a program to create a Salary class.

Solution:

Step 1: Create a class **Salary**. Unless mentioned, it is considered that **da is 25% of basic and hra is 30% of basic**.

```
public Salary
{

    private int basic;    // members of Salary class

    private double da, hra;    // members of Salary class

    public Salary()
```

```
{  
    // To Do: implement the default constructor  
}  
  
} // end of class Salary ;}
```

Example 11: Salary.java

<<TO DO>>

Assignment-6: Create a class called Employee with the following attributes. Also, add the salary class as a member of the employee class.

```
employeeId  
firstName  
lastName  
address  
phoneNumber  
dateOfBirth  
hireDate  
departmentId
```

Example 12:Employee.java

Create the following methods in the Employee class which performs the following

1. Set and Retrieve various data members, the data members should be valid. For example user cannot enter an employee who is below 21
2. Calculate the experience of the employee
3. Calculate the age of the employee
4. Calculating the expected retirement date (dob+58 years) of an employee
5. Find the probation status, an employee is in probation if he has not completed one year in the company.
6. Compare two employees in terms of experience

Create a class called UseEmployee to display the following menu and call the respective methods of Employee class

1. Create new Employees
2. Print Employee details
3. Compare the experience of two Employees
4. Exit

Assignment-7:

To the above employee class, add appropriate constructors .Create an Address class which would have the following data members

- street
- city
- country
- pincode

Replace the address data member in the employee class with the address object and modify the constructors and methods accordingly

3.5: Inheritance

<<TO DO>>

Assignment-8: Look at the Account class and write a main method in a different class to briefly experiment with some instances of the Account class.

```
public class Account
{
    private double balance; //current balance
    private int accNum; //account number
    public Account(int accNum)
    {
        this.balance=0.0f;
        this.accNum=accNum;
    }
    public void deposit(double sum)
    {
        bal-=sum;
    }

    public double getBalance()
    {
        return balance;
    }
    public double getAccountNumber()
    {
        return accNum;
    }
    public String toString()
    {
        return "Acc"+accNum+"."+balance+"balance="+balance;
    }
    public final void print()
    {
        //Don't override this,
        //override the toString method
    }
}
```

Example 13:Account.java

Inherit two classes Savings Account and Current Account from the above Account Class. Implement the following in the respective classes.

- **Savings Account**
 - a. Add a variable called minimumBalance
 - b. Add a method called withdraw (This method should check for minimum balance and allow withdraw to happen)
- **Current Account**
 - a. Add a variable called overDraftLimit
 - b. Add a method called isOverDraftLimit (checks whether overdraft limit is reached and returns a boolean value accordingly)

Create a new class called Ledger and Implement the following

- a. Variables: Array of Accounts, ledgerNumber and ledgerStartDate
- b. Methods: printStatement, which will print Account Number, Ledger Number and the balance amount for a given Account Number.

3.6: Working with Strings.

<<TO DO>>

Assignment-9: Write a program to accept a string using the **StringBuilder** class and insert a * wherever a vowel appears.

```
class UseStringBuilder{
    public static void main(String args[]) {
        // Get input from the user.
        //Create the StringBuilder

        //Append the text entered by the user to the end of the buffer

        // Loop Through to replace vowels using * and print the string . Use the insert and
        delete methods
    }
}
```

Example 14: Sample code