Core Java 8 and Development Tools
Lab Book

Capgemini Public

Document Revision History

Date	Revision No.	Author	Summary of Changes
17-11-2013	1.0	Rathnajothi P	As of updated module content, designed lab book
28-05-2015	2.0	Vinod Satpute	Updated to include new features of Java SE 8, Junit 4 and JAXB 2.0
25-05-2016	3.0	Tanmaya K Acharya	Updated as per the integrated ELT TOC
10-01-2017	4.0	Vaishali Srivastava	TOC Updated as per BU Requirements for VNV Automation

The information contained in this document is proprietary and confidential. For Capgemini only. | **2** / 35

Capgemini Public

|

Capgemini Public

Table of Contents

|

Getting Started

Overview

This lab book is a guided tour for learning Core Java version 8 and development tools. It comprises of assignments to be done. Refer the demos and work out the assignments given by referring the case studies which will expose you to work with Java applications.

Setup Checklist for Core Java

Here is what is expected on your machine in order to work with lab assignment.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 7 or higher.
- Memory: (1GB or more recommended)
- Internet Explorer 9.0 or higher or Google Chrome 43 or higher
- Connectivity to Oracle database

Please ensure that the following is done:

- A text editor like Notepad or Eclipse is installed.
- JDK 1.8 or above is installed. (This path is henceforth referred as <java_home>)

Instructions

- For all Naming conventions, refer Appendix A. All lab assignments should adhere to naming conventions.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory java_assignments. For each lab exercise create a directory as lab <lab number>.

Learning More (Bibliography if applicable)

- https://docs.oracle.com/javase/8/docs/
- Java, The Complete Reference; by Herbert Schildt
- Thinking in Java; by Bruce Eckel
- Beginning Java 8 Fundamentals by KishoriSharan

The information contained in this document is proprietary and confidential. For Capgemini only. | **5** / 35

Capgemini Internal

Problem Statement/ Case Study (If applicable)

1. **Bank Account Management System:**

   ➢ Funds Bank needs an application to feed new Account Holder information. AccountHolder will be a person. There are two types of accounts such as SavingsAccount, CurrentAccount.

2. **Employee Medical Insurance Scheme:**

   ➢ By default, all employees in an organization will be assigned with a medical insurance scheme based on the salary range and designation of the employee. Refer the below given table to find the eligible insurance scheme specific to an employee.

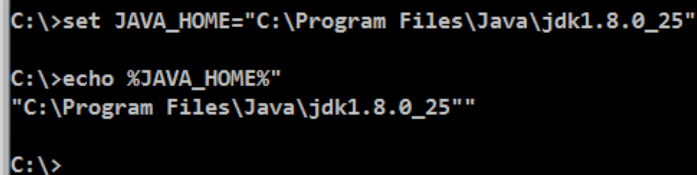| Salary	Designation	Insurance scheme
>5000 and < 20000	System Associate	Scheme C
>=20000 and <40000	Programmer	Scheme B
>=40000	Manager	Scheme A
<5000	Clerk	No Scheme

Lab 1: Working with Java and Eclipse IDE

Goals	Learn and understand the process of: ➢ Setting environment variables ➢ Creating a simple Java Project using Eclipse 3.0 or above
Time	195 minutes

1.1: Setting environment variables from CommandLineSolution:

Step 1: Set **JAVA_HOME** to Jdk1.8 using the following command:

- **Set JAVA_HOME=**C:\Program Files\Java\jdk1.8.0_25

```
C:\>set JAVA_HOME="C:\Program Files\Java\jdk1.8.0_25"

C:\>echo %JAVA_HOME%"
"C:\Program Files\Java\jdk1.8.0_25""

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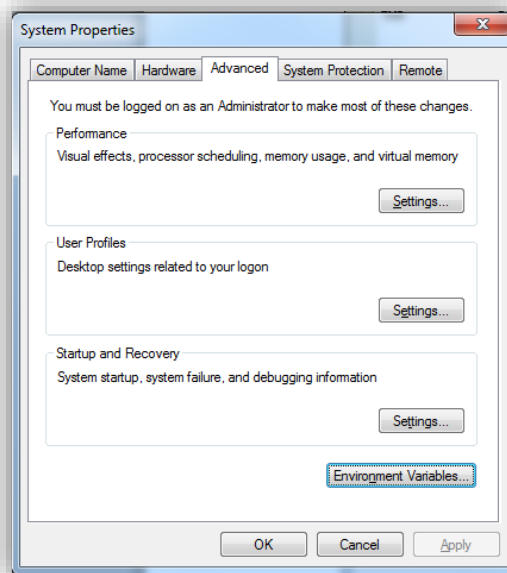
For example: C:\Test\myproject\Class;ant.jar

Capgemini Internal

⚠️   Alternatively follow the following steps for setting the environment variables
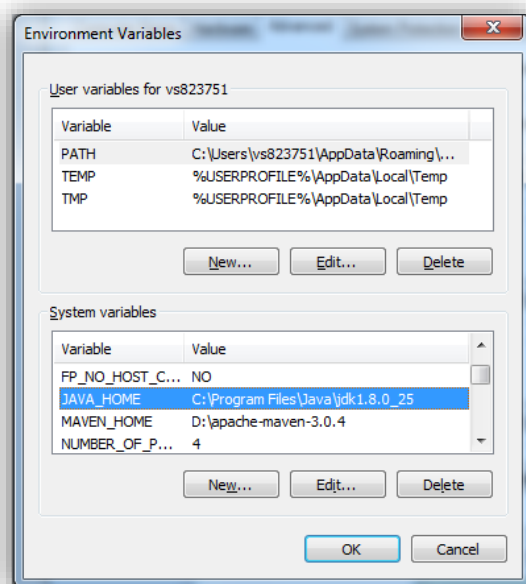
Alternate approach:

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

Figure 2: System Properties
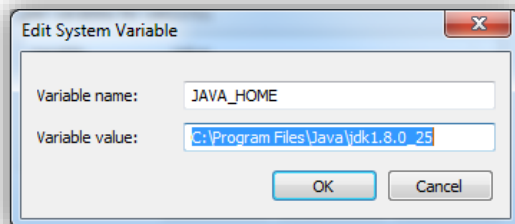
Step 2: Click **Environment Variables**. The Environment Variables window will bedisplayed.
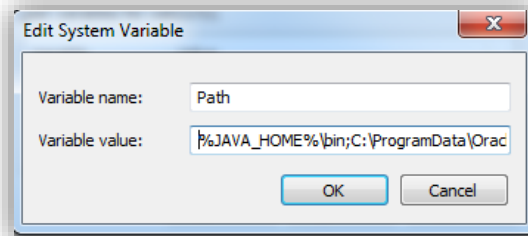
Capgemini Internal

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 JDK1.8 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.
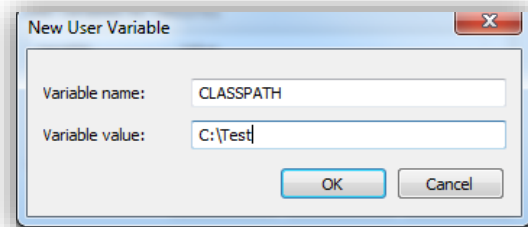
Figure 6: Edit User Variable

1.2: Create Java Project

Create a simple java project named 'MyProject'.

Solution:

Step 1: Open **eclipse 4.4**(or above)

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

Capgemini Internal
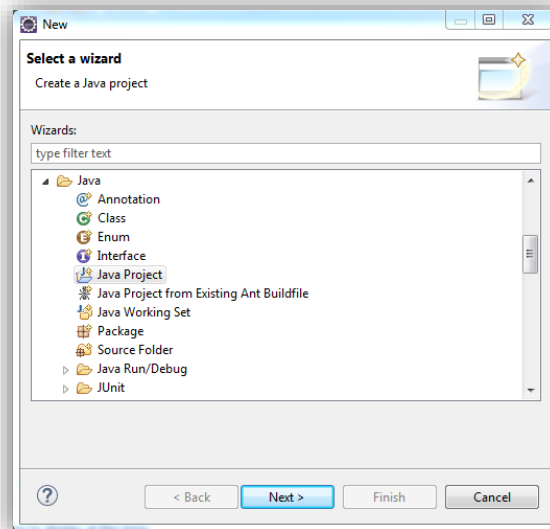
Figure 7: Select Wizard

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

Capgemini Internal
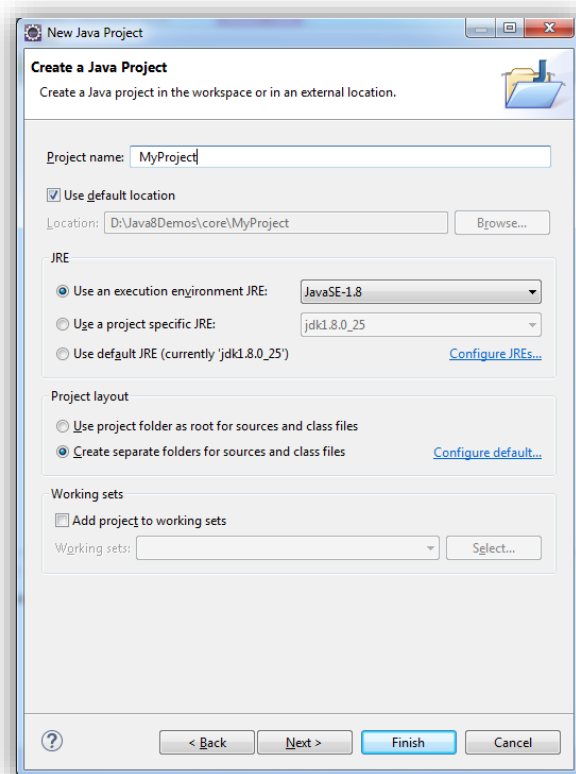
Figure 8: New Java Project

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

Capgemini Internal
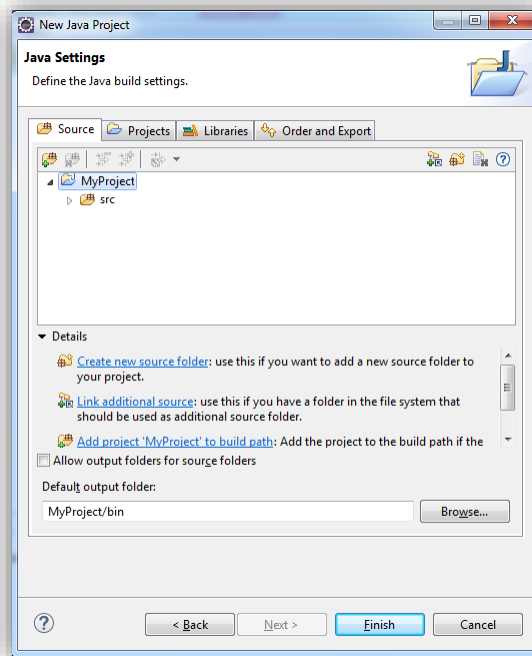
Figure 9: Java Settings

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

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

|

Capgemini Internal
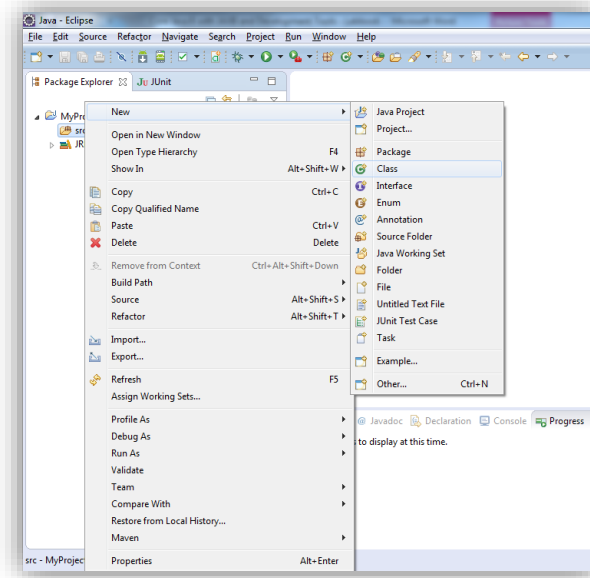
Figure 10: Select Resource

©2016 Capgemini. All rights reserved.

The information contained in this document is proprietary and confidential. For Capgemini only. | **14** / 35

Capgemini Internal

Step 7: Provide name and other details for the class, and click **Finish**.
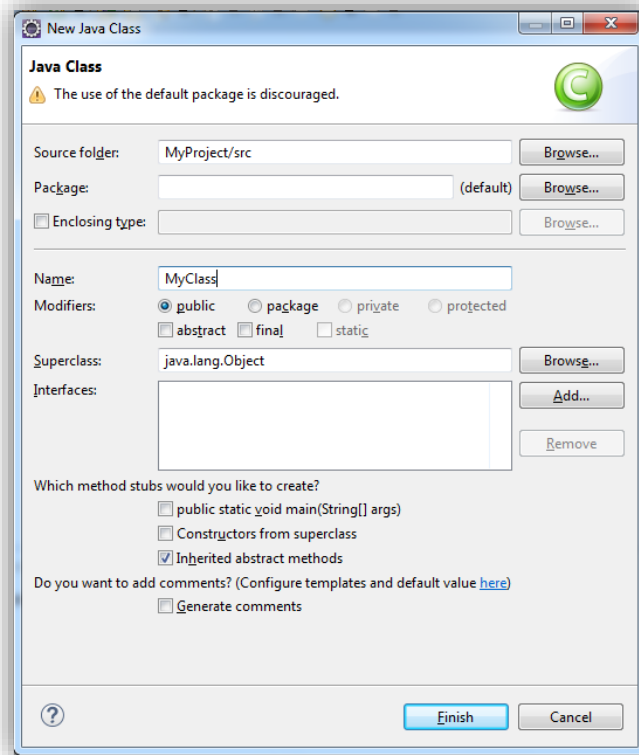
Figure 11: Java Class

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.

1.3: Using offline Javadoc API in Eclipse

Step 1: Open **eclipse 4.4**(or above)

Step2: From eclipse Window → Preferences →Java → "Installed JREs" select available JRE (jdk1.8.0_25 for instance) and click Edit.

|

Capgemini Internal

Step3: Select all the "JRE System libraries" using Control+A.

Step 4: Click "Javadoc Location"

Step 5: Change "Javadoc location path:" from
http://download.oracle.com/javase/8/docs/api/ to "file:/E:/Java/docs/api/".

Capgemini Internal
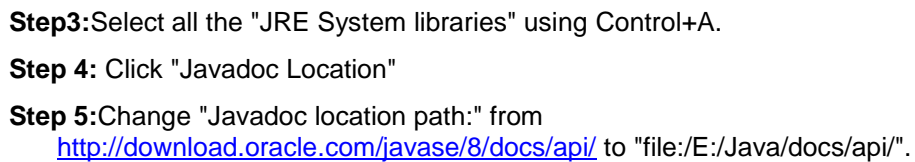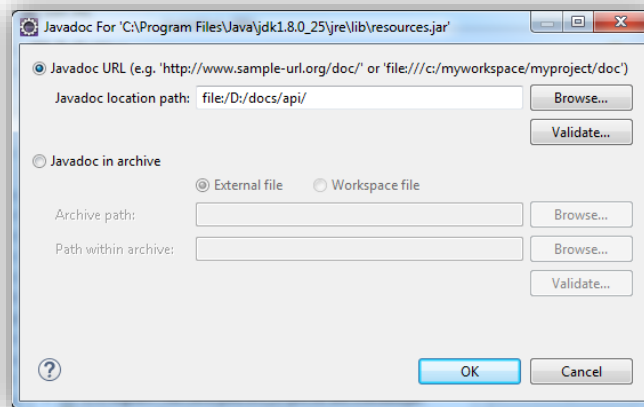
Step 6: Close all windows by either clicking on ok/apply.

Step 7: Open the Javadoc view from Window → Show View → Javadoc.
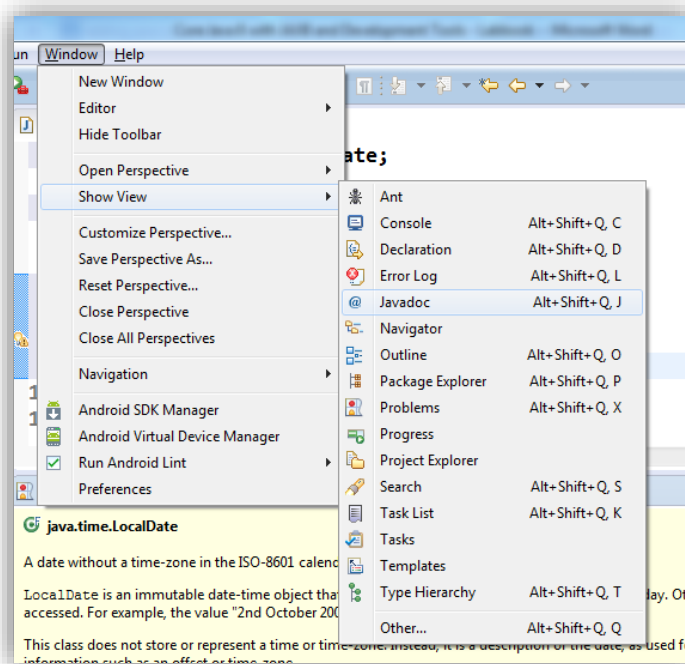
Note: Henceforth whenever you select any class or method in Editor Window, it Javadoc view will display the reference documentation.
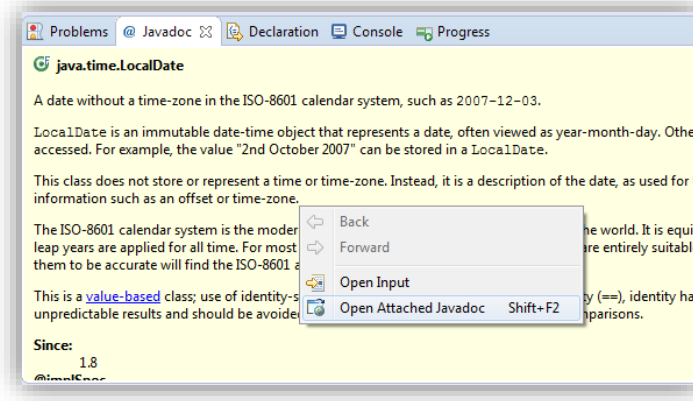
If you want to open the Java documentation for specified resource as html page, right click in the Javadoc view → Open Attached Javadoc.

1.4: Write a java program to print person details in the format as shown below:

Person Details:
_____

First Name: Divya
Last Name: Bharathi
Gender: F
Age: 20
Weight: 85.55

Figure 12: Sample output of Person details

1.5: Write a program to accept a number from user as a command line argument and check whether the given number is positive or negative number.

1.6: Refer the class diagram given below and create person class.

Capgemini Internal
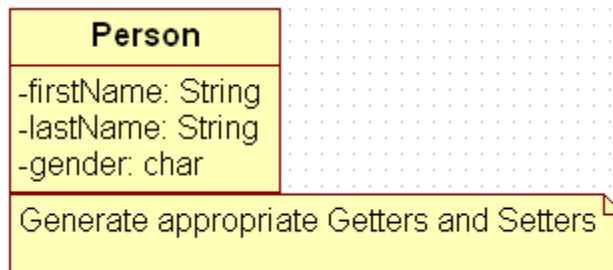
Figure 13: Class Diagram of Person

Create default and parameterized constructor for Person class.

Also Create "PersonMain.java" program and write code for following operations:

      a) Create an object of Person class and specify person details through constructor.

      b) Display the details in the format given in Lab assignment 1.4

1.7: Modify Lab assignment 1.6 to accept phone number of a person. Create a new method to implement the same and also define method for displaying person details.

1.8: Modify the above program, to accept only 'M' or 'F' as gender field values. Use Enumeration for implementing the same.

1.9: Create a method which can perform a particular String operation based on the user's choice. The method should accept the String object and the user's choice and return the output of the operation.
Options are

- Add the String to itself
- Replace odd positions with #
- Remove duplicate characters in the String
- Change odd characters to upper case

1.10: Create a method that accepts a String and checks if it is a positive string. A string is considered a positive string, if on moving from left to right each character in the String comes after the previous characters in the Alphabetical order. For Example: ANT is a positive String (Since T comes after N and N comes after A). The method should return true if the entered string is positive.

1.11: Create a program to accept date and print the duration in days, months and years with regards to current system date.

  |

Capgemini Internal

1.12: Revise exercise **1.11** to accept two LocalDates and print the duration between dates in days, months and years.

1.13: Create a method to accept product purchase date and warrantee period (in terms of months and years). Print the date on which warrantee of product expires.

1.14: Create a method which accept zone id and print the current date and time with respect to given zone. (Hint: Few zones to test your code. America/New York, Europe/London, Asia/Tokyo, US/Pacific, Africa/Cairo, Australia/Sydney etc.)

**1.15**: Modify Lab assignment 1.6 to perform following functionalities:

   a) Add a method called calculateAge which should accept person's date of birth and

   calculate age of a person.

   b) Add a method called getFullName(String firstName, String lastName) which should

   return full name of a person

   Display person details with age and full name.

Lab 2: Inheritance and Polymorphism

Goals	At the end of this lab session, you will be able to: ➢ Write a Java program that manipulates person details ➢ Working with Inheritance, Polymorphism
Time	315 minutes

2.1: Refer the case study 1 in Page No: 5 and create Account Class as shown below in class diagram. Ensure minimum balance of INR 500 in a bank account is available.
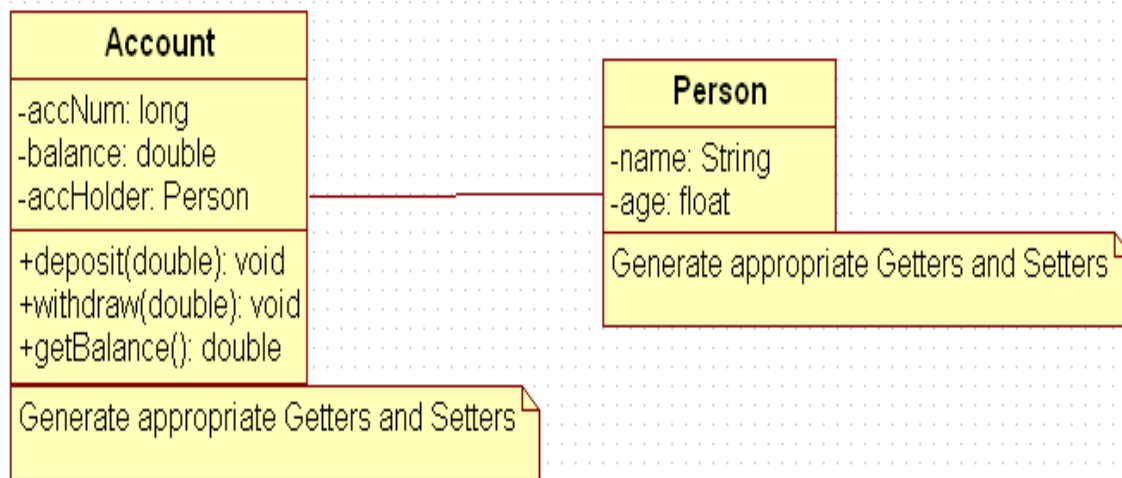
Figure 14: Association of person with account class

a) Create Account for smith with initial balance as INR 2000 and for Kathy with initial balance as 3000.(accNum should be auto generated).

b) Deposit 2000 INR to smith account.

c) Withdraw 2000 INR from Kathy account.

d) Display updated balances in both the account.

e) Generate toString() method.

2.2: Extend the functionality through Inheritance and polymorphism (Maintenance)

Inherit two classes Savings Account and Current Account from account class. Implement the following in the respective classes.

a) Savings Account
   a. Add a variable called minimum Balance and assign final modifier.
   b. Override method called withdraw (This method should check for minimum balance and allow withdraw to happen)

b) Current Account
   a. Add a variable called overdraft Limit
   b. Overridemethod called withdraw (checks whether overdraft limit is reached and returns a boolean value accordingly)

2.3: Refer the case study 2 in page no: 5 and create an application for that requirement by creating packages and classes as given below:

a) com.cg.eis.bean

In this package, create "Employee" class with different attributes such as id, name, salary, designation, insuranceScheme.

b) com.cg.eis.service

This package will contain code for services offered in Employee Insurance System. The service class will have one EmployeeService Interface and its corresponding implementation class.

c) com.cg.eis.pl

This package will contain code for getting input from user, produce expected output to the user and invoke services offered by the system.

The services offered by this application currently are:
   i) Get employee details from user.

ii) Find the insurance scheme for an employee based on salary and
designation.

iii) Display all the details of an employee.

2.4: Use overrides annotation for the overridden methods available in a derived class of an
interface of all the assignments.

2.5: Refer the problem **statement 2.1**. Modify account class as abstract class and declare
withdraw method.

The information contained in this document is proprietary and confidential. For Capgemini only. | **23** / 35

Capgemini Internal

Lab 3: Arrays, Regular Expression, Exception Handling

Goals	At the end of this lab session, you will be able to: ➢ Create and use application specific exceptions
Time	195 minutes

3.1: Modify the Lab assignment **1.6** to validate the full name of an employee. Create and throw a user defined exception if first Name and lastName is blank.

3.2: Validate the age of a person in Lab assignment 2.1 and display proper message by using user defined exception. Age of a person should be above 15.

3.3: Modify the Lab assignment **2.3** to handle exceptions. Create an Exception class named as "EmployeeException"(User defined Exception) in a package named as "com.cg.eis.exception" and throw an exception if salary of an employee is below than 3000. Use Exception Handling mechanism to handle exception properly.

3.4: Write a program to store product names in a string array and sort strings available in an array.

|

Capgemini Internal

Lab 4 : Files IO and Collections

Goals	At the end of this lab session, you will be able to: ➢ Read and write data using streams. ➢ Serialize and Deserialize objects
Time	285 minutes

4.1: Write a program to read content from file, reverse the content and write the reversed content to the file. (Use Reader and Writer APIs).

4.2: Create a file named as "numbers.txt" which should contain numbers from 0 to 10 delimited by comma. Write a program to read data from numbers.txt using Scanner class API and display only even numbers in the console.

4.3: Enhance the lab assignment **3.3** by adding functionality in service class to write employee objects into a File. Also read employee details from file and display the same in console. Analyze the output of the program.

4.4: Modify the above program to store product names in an ArrayList, sort strings available in an array List and display the names using for-each loop.

4.5: Modify the lab assignment **2.3** to accept multiple employee details and store all employee objects in a HashMap. The functionalities need to be implemented are:

   i)   Add employee details to HashMap.
   ii) Accept insurance scheme from user and display employee details based on Insurance scheme
   iii) Delete an employee details from map.

Sample code Snippet of EmployeeServiceImpl class:

```
public class EmployeeServiceImpl {

HashMap<String,Employee> list = new HashMap<String,Employee>();

public void addEmployee(Employee emp)          {
                                        //code to add employee
}
public boolean deleteEmployee(int id)     {
//    code to delete a employee whose id is passed as parameter
}
………………..
}
```

Capgemini Internal

Lab 5: Introduction to Junit

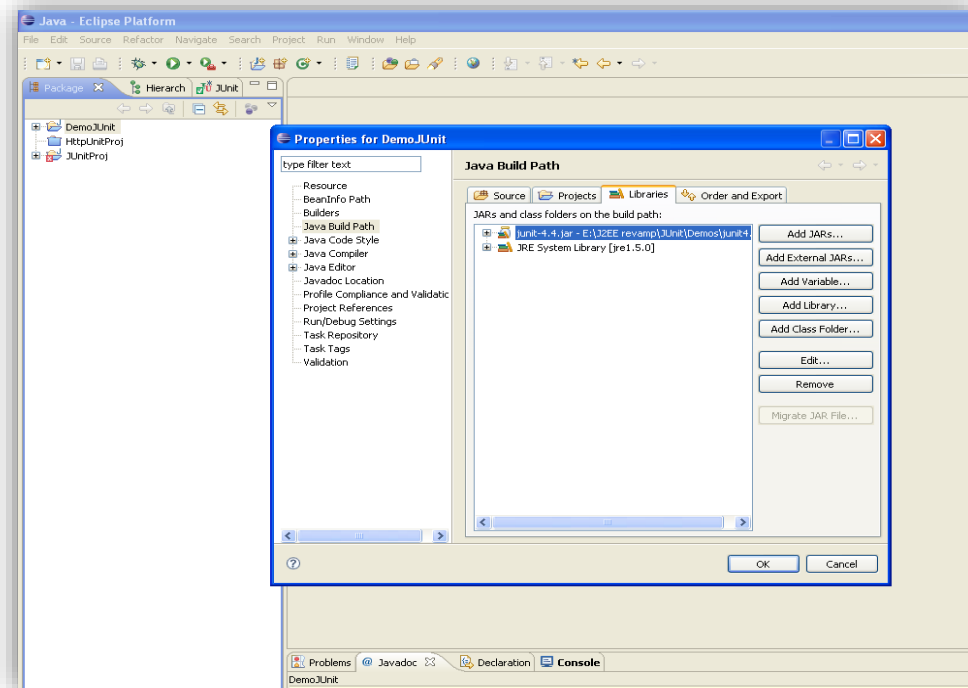| Goals	At the end of this lab session, you will be able to:
	➢ Configuring JUnit in Eclipse
	➢ Using JUnit to write TestCase for standalone Java Applications
Time	330 minutes

5.1: Configuration of JUnit in Eclipse

Step 1: Create a Java project.
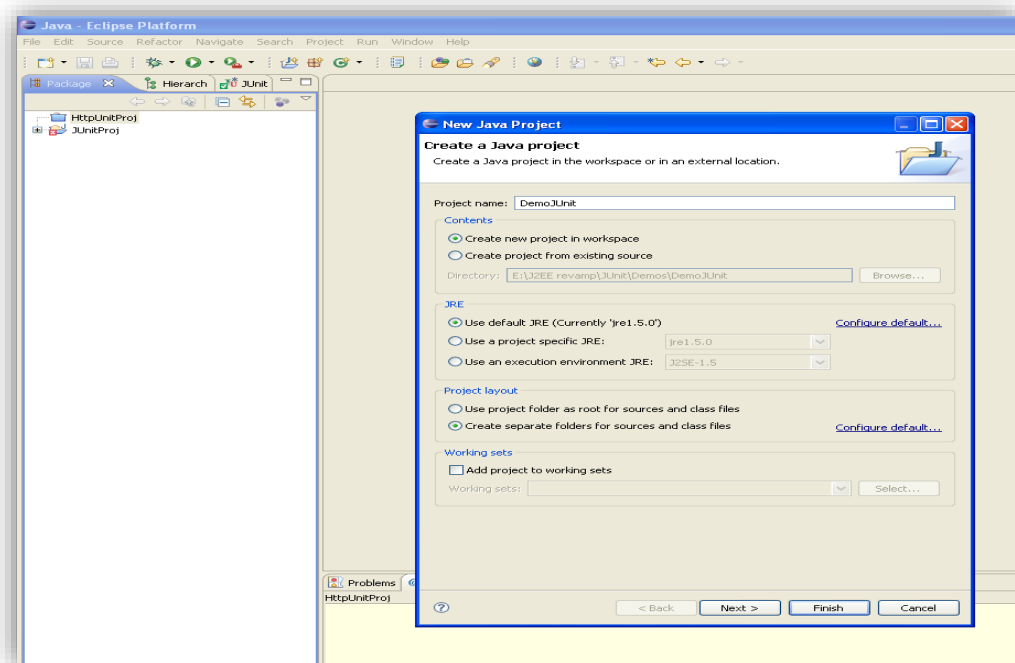
Figure 15: Creating Java Project in Eclipse

Step 2: Add **junit4.4.jar** in the build path of the project.

|

Capgemini Internal

Figure 16: Adding junit4.4.jar in the build path

Step 3: Write the java class as follows:

```
public class  Person
{
        private String firstName;
        private String lastName;

        public Person(String fname,Stringlname)
        {
                if(fname == null &&lname==null){
                        throw new IllegalArgumentException("Both Names
Cannot be NULL");
                }
                this.firstName=fname;
                this.lastName = lname;
        }

public String getFullName()
{
        String first=(this.firstName != null)? this.firstName:"?";
```

|

Capgemini Internal

```
                    String last=(this.lastName != null)? this.lastName:"?";
                    return first + " " + last;
            }

            public String getFirstName(){
                    return this.firstName;
            }

            public String getLastName(){
                    return this.lastName;
            }
            public static void main(String args[])
            {
                    Person p=new Person("a","b");
                    System.out.println(p.getFirstName());
            }
    }
```
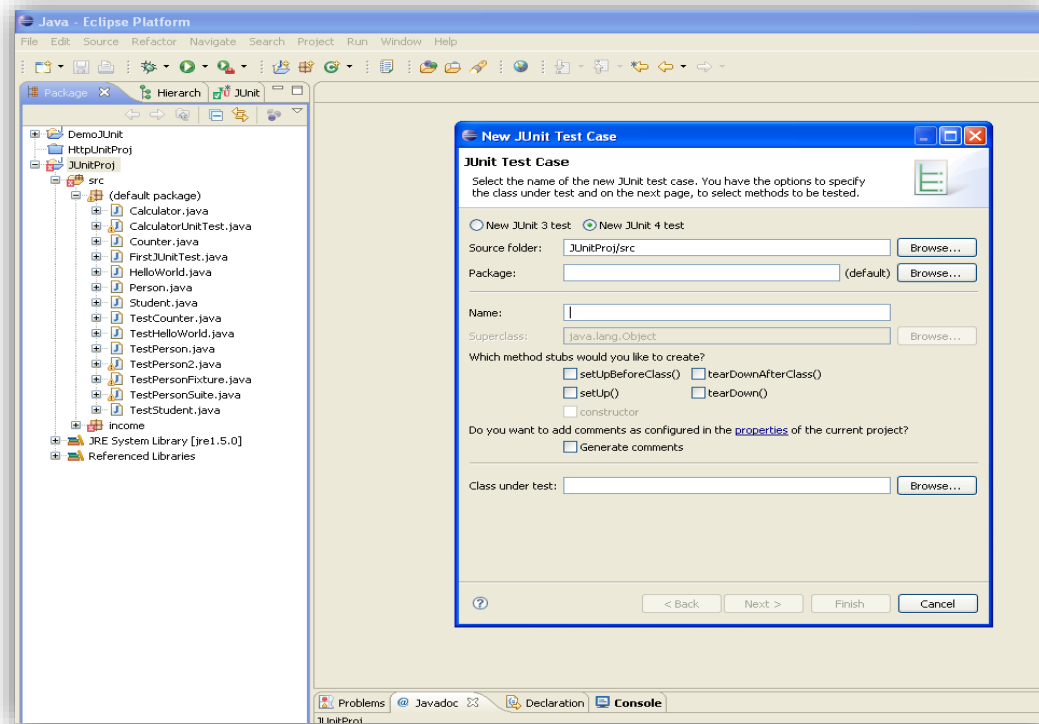
Example 1: Person.java

Step 4: Write the JUnit test class.

- Create a JUnit test case in Eclipse.

|

Capgemini Internal
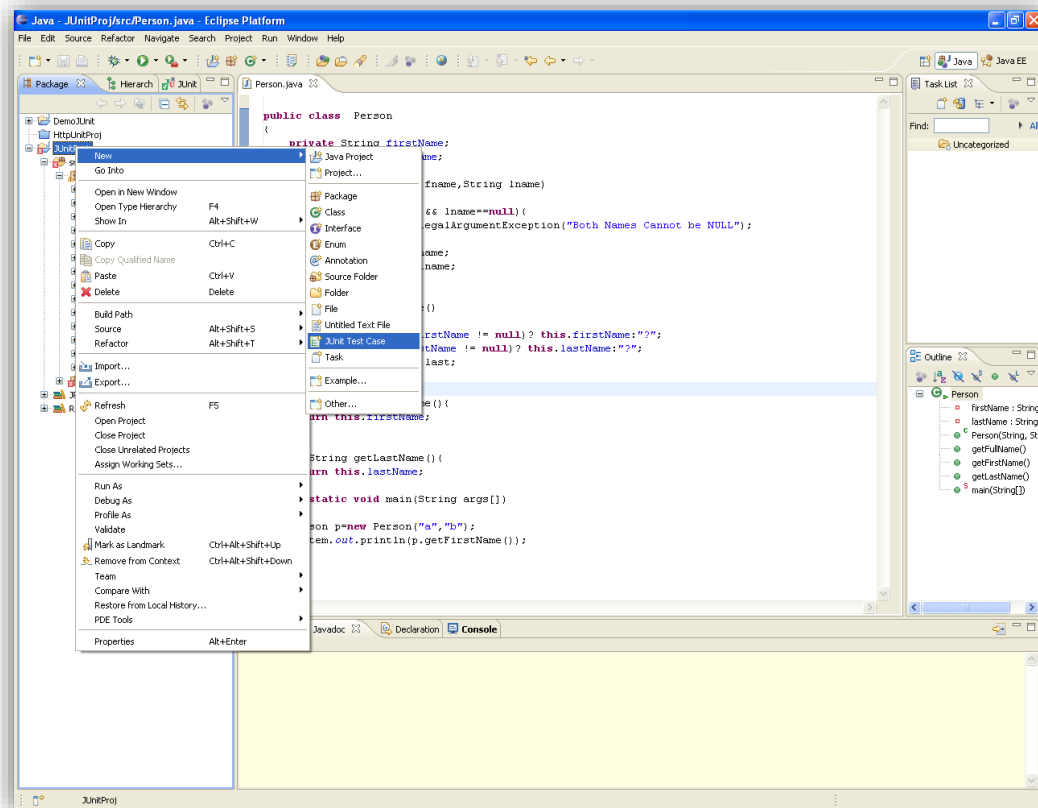
Figure 17: Adding the JUnit test case to the project

- A dialog box opens, where you need to specify the following details:
  - The Junit version that is used
  - The package name and the class name
  - The class under test
  - You can also specify the method stubs that you would like to create

---

The information contained in this document is proprietary and confidential. For Capgemini only. | **30** / 35

Capgemini Internal

Figure 18: Specifying information for the test case

- Write the code as follows:

```
importorg.junit.*;
import org.junit.Test;
import static org.junit.Assert.*;

public class  TestPerson2
{
        @Test
        public void testGetFullName()
        {
                System.out.println("from TestPerson2");
                Person per = new Person("Robert","King");
                assertEquals("Robert King",per.getFullName());
        }

        @Test (expected=IllegalArgumentException.class)
        public void testNullsInName()
                {
```

©2016 Capgemini. All rights reserved.

The information contained in this document is proprietary and confidential. For Capgemini only. | **31** / 35

Capgemini Internal

```
                        System.out.println("from TestPerson2 testing exceptions");
                        Person per1 = new Person(null,null);
                    }
                }
```

Example 2: TestPerson2.java

Step 5: Run the test case.
- Right click the test case class, and select **RunAs→ JUnit Test**.
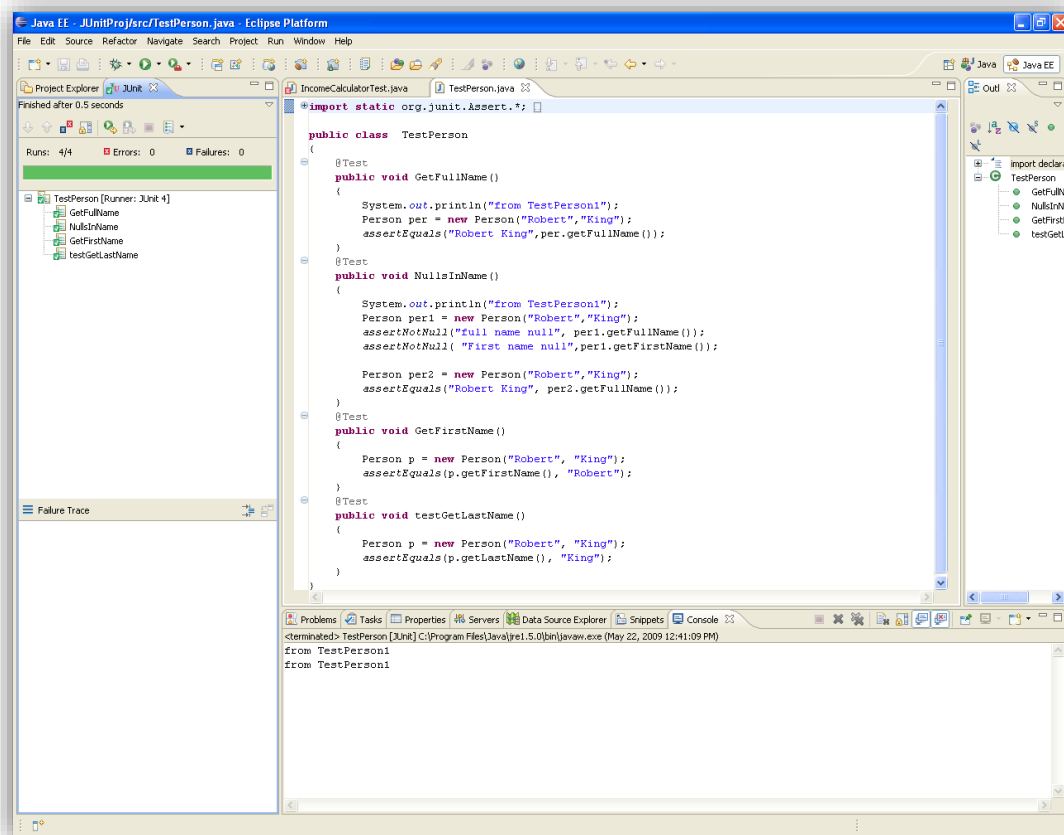- The output will be displayed as shown below:

Figure 19: Output of JUnit text case execution

5.2: Writing JUnit tests

Consider the following Java program. Write tests for testing various methods in the class

Capgemini Internal

Solution:

Step 1: Write the following Java Program **Date.java.**

```java
class Date
    {
 intintDay, intMonth, intYear;
 // Constructor
 Date(int intDay, int intMonth, int intYear)     {
     this.intDay = intDay;
this.intMonth = intMonth;
this.intYear = intYear;
 }
 // setter and getter methods
 voidsetDay(int intDay)
{
     this.intDay = intDay;
}
     intgetDay( )
{
     return  this.intDay;
}

 voidsetMonth(int intMonth)
{
     this.intMonth = intMonth;
}

 intgetMonth( )
{
     return  this.intMonth;
}

 voidsetYear(int intYear)
{
     this.intYear=intYear;
}



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

```
} // Date class
```

Example 3: Date.java

Step 2: Write test class for testing all the methods of the above program and run it using the eclipse IDE.

5.2.1: Consider the Person class created in lab assignment 1.6. This class has some members and corresponding setter and getter methods. Write test case to check the functionality of getter methods and displaydetails method.

5.2.2: Consider the lab assignment **3.3** from Exception Handling Lab. Create a new class ExceptionCheck.java which handles an exception. Write a test case to verify if the exception is being handled correctly.

The information contained in this document is proprietary and confidential. For Capgemini only. | **34** / 35

Capgemini Internal

Appendix A: Table of Figures

Capgemini Internal