

# LAB 1

## Familiarization of Rational Rose Environment And UML for small Java Application Development

### OBJECTIVE AND BACKGROUND

The purpose of this first UML lab is to familiarize programmers with Rational Rose UML environment and Java application development. We will learn how to work with Rational Rose Enterprise software, Case diagram, and Class Diagram. We will also learn how to create a Use Case, two Java classes called HelloUML and Test, generate Java code, edit the Java code, and compile the Java program. At the end of this lab, you should be able to run the program Test to display HelloUML message under the Window's command line.

### WEB SITE REFERENCES

- Professor Lin's Web site: <http://www.etcs.ipfw.edu/~lin>
- Rational, the Software Development Company [www.rational.com](http://www.rational.com)

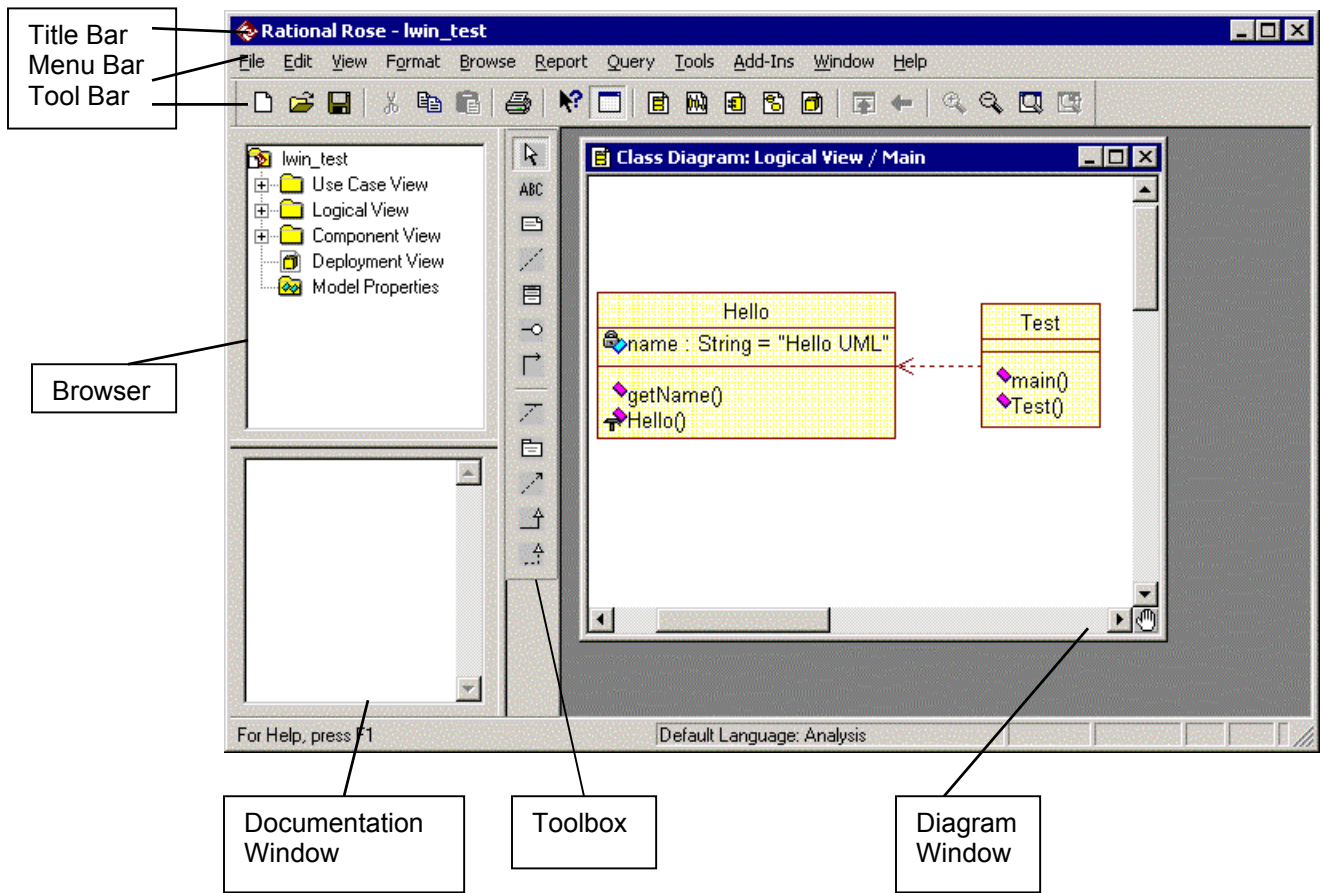
### EQUIPMENT AND SOFTWARE

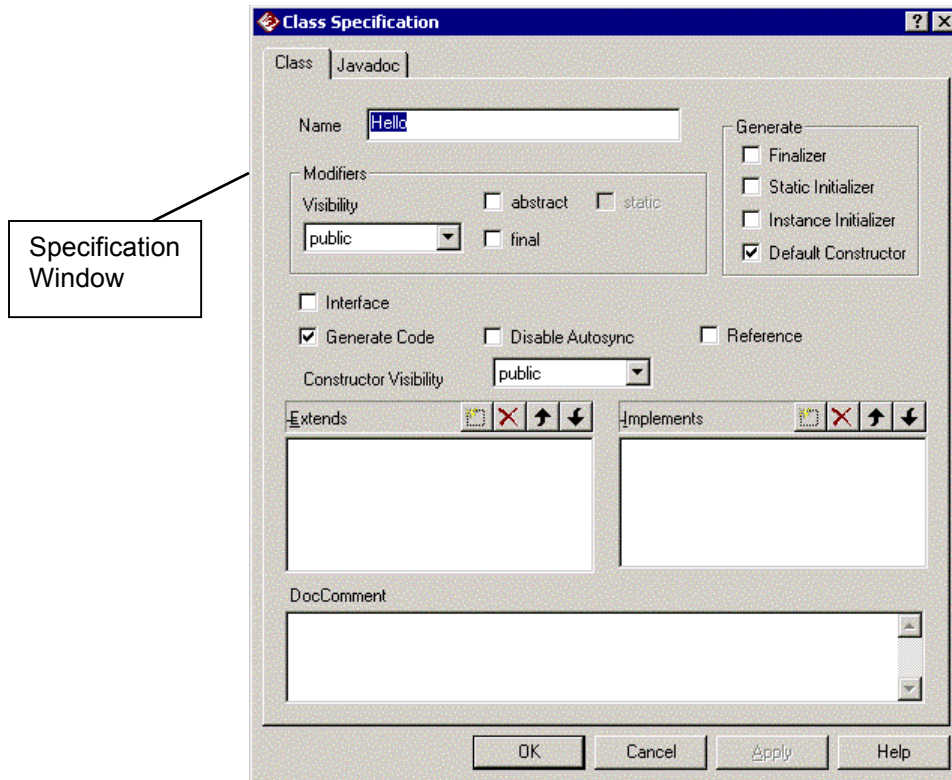
- PC (256 Mbytes, Windows 95/98/2000 OS)  
Rational Rose Enterprise Software
- SUN Java Development Tool kit J2SE (standard edition) or J2EE (enterprise edition)
- Notepad editor

### PROCEDURE

#### a. A Tour of Rational Rose Environment

- Application Window: a title bar, tool bar, menu bar, and a work area for displaying toolbox, browser, document window, diagram window, and specification window
- Browser: a window with navigation tool for viewing diagrams and many other model elements
- Documentation Window: for preparing text information to describe model elements or relationships
- Diagram Windows
- Overview Window: can be showed through put the mouse pointer over hand icon
- Other windows such as Log, Edit, can be access through **View** on menu bar



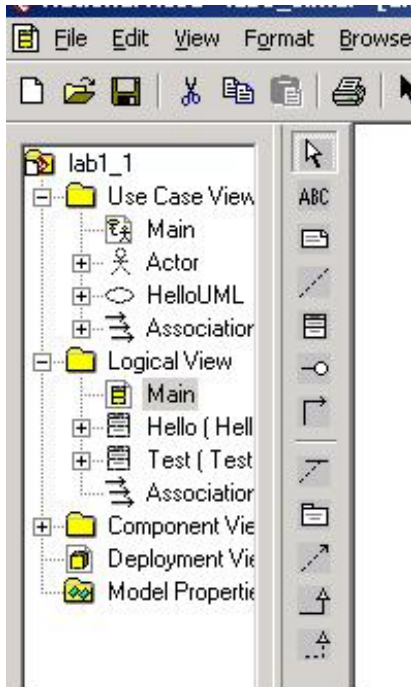


## b) Creating a simple application (Hello.java)

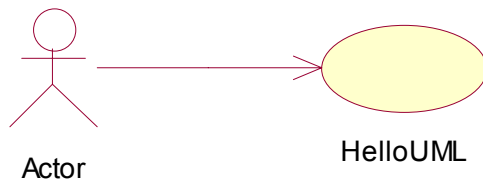
**Step 1: Run Rational Rose Enterprise**

**Step 2: Create a Use Case Diagram**

1. Click on Use **Case View > Main**

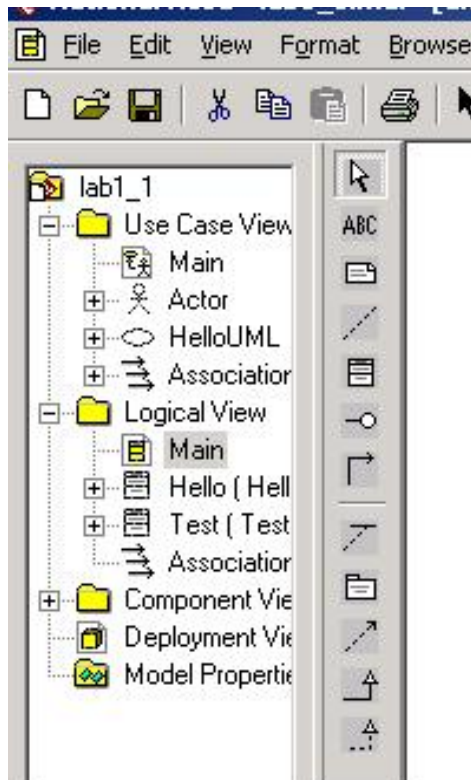


2. From the menu bar select **Tool > Create > Use Case**
3. Select the **Actor** from the Toolbox and place it into the Use Case Diagram
4. Right click on the Actor to see the pop-up menu, and then select **Open Specification** menu
5. Select the Use Case from the Toolbox and place it into the User Case Diagram
6. Right click on the User case to change the name to **HelloUML**

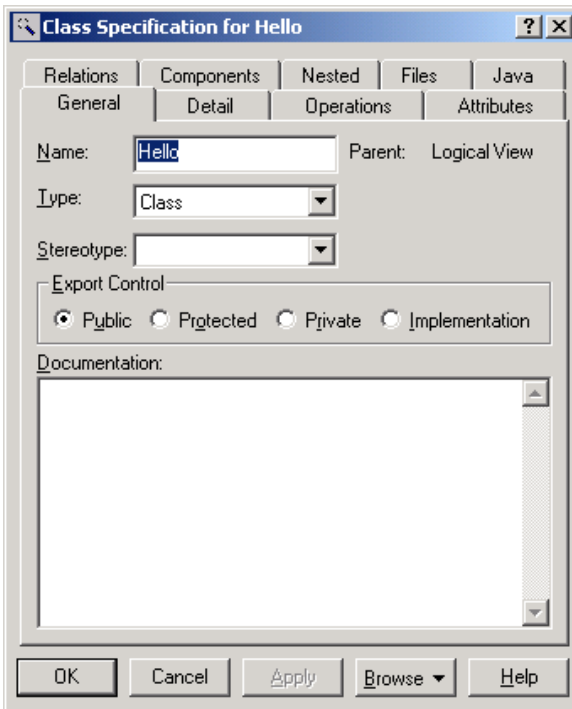


### Step 3: Create Hello Class Diagram (classes, attributes, and operations)

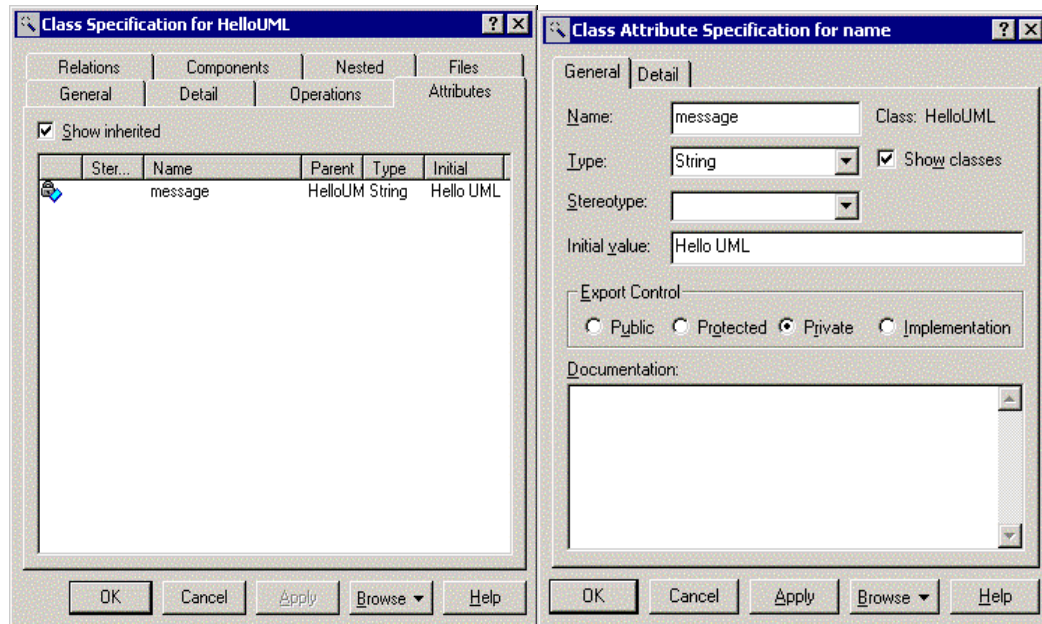
1. Click on Use **Logical View > Main**



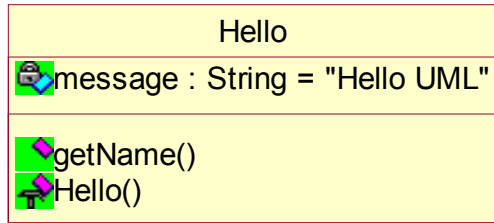
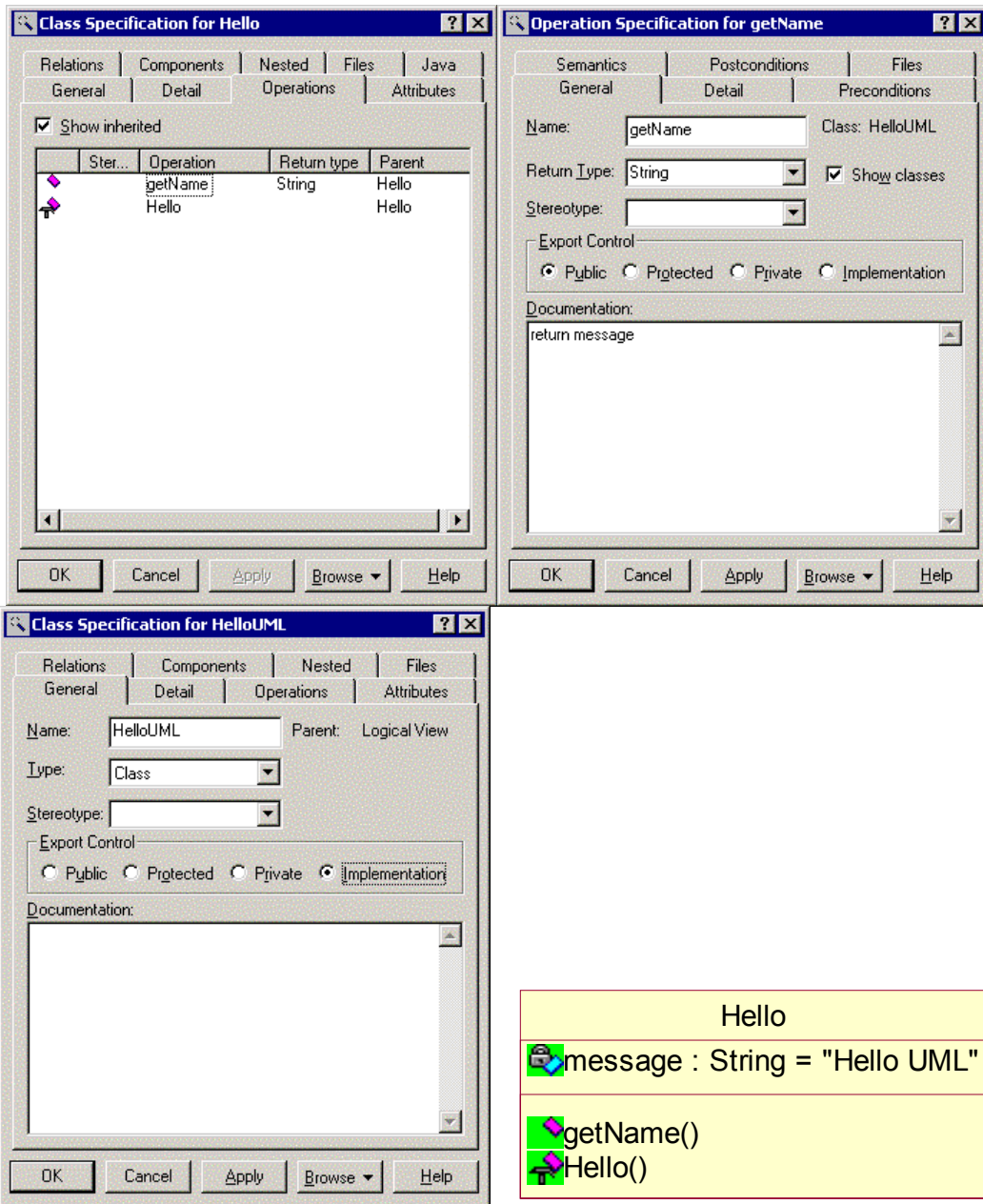
2. Create a new class called "Hello"
  - **Tools > Create > Class**
  - Select the Class icon (rectangle box symbol) from the Toolbox
  - Place a class to the workspace of the Class Diagram
  - Type in a desired class name: **"Hello"**
  - Right click the new class diagram and select **"Open Specification"**
  - View the class name in the name box: **Hello**
  - Select **"class"** in the Type combo box



3. Add **Attributes** to the Hello class
  - Right click the Hello class to select **New Attributes** from a dialog box
  - Accept the default **name**, and hit enter
  - Right click the Hello class again to select **Open Specification**
  - The window called Class Specification for Hello is then displayed; change the attribute name to **message**; select Type as: **String**; specify Initial Value with a pair double quotes as: **"Hello UML"**; set Export Control to: **private**

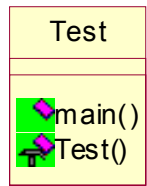
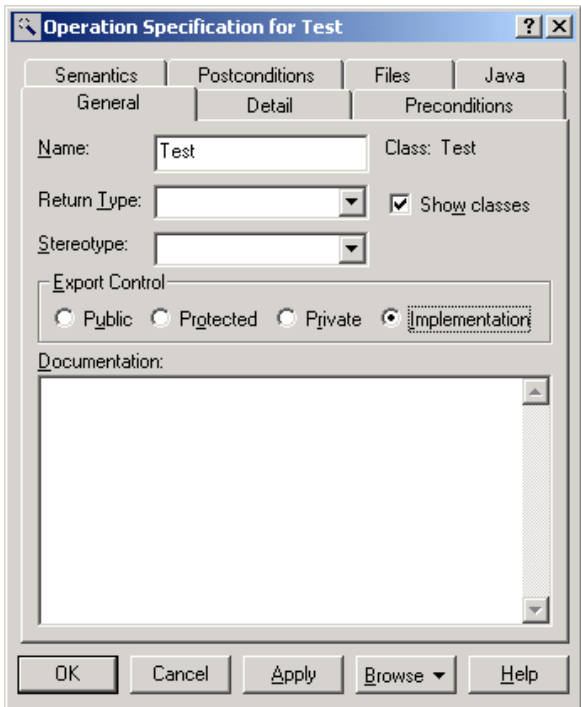
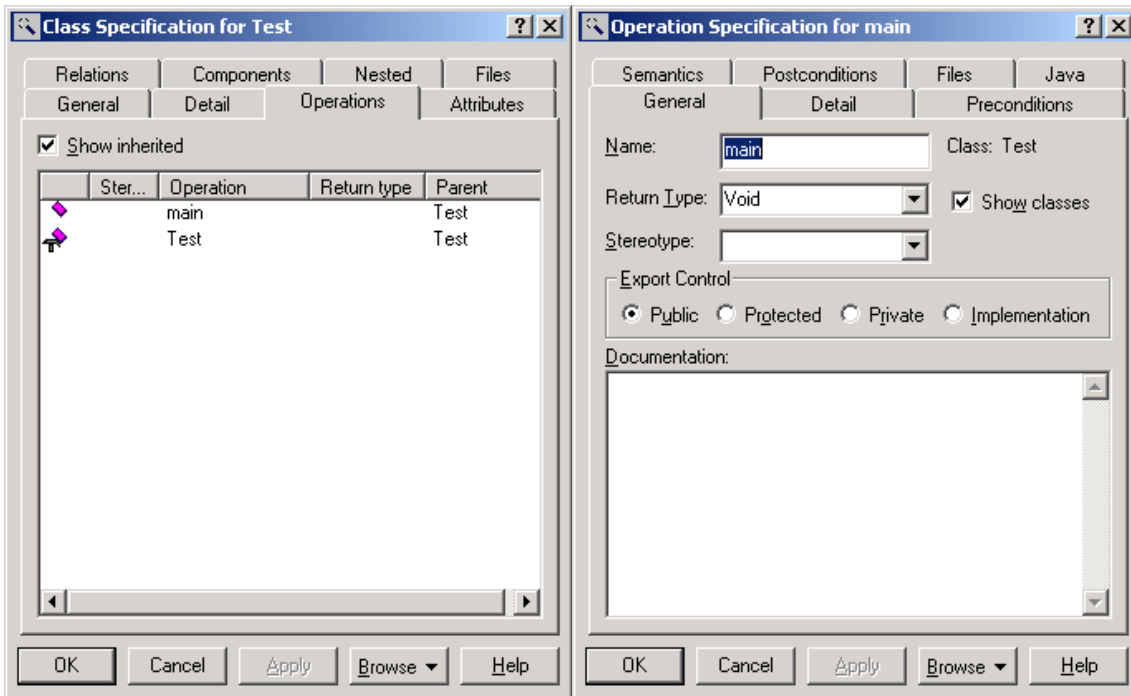


4. Add **Operations** to the Hello class
  - Right click the Hello class to get an dialog box
  - Select **New Operations**, and give a new method name as **getName**, then hit Enter key to enter another method name as **Hello**
  - Double click the **getName** method to display Operation Specification menu for getName dialog box
  - Add return type as: **String**; add some information on this operation method to Document text box
  
5. Change Operation Specification for Hello method
  - Right click on the class diagram Hello to select Operation Specification for showing Class Specification for Hello
  - Select **Implementation** radio button in the Export Control
  - Verify the Hello class diagram is the same as shown below.



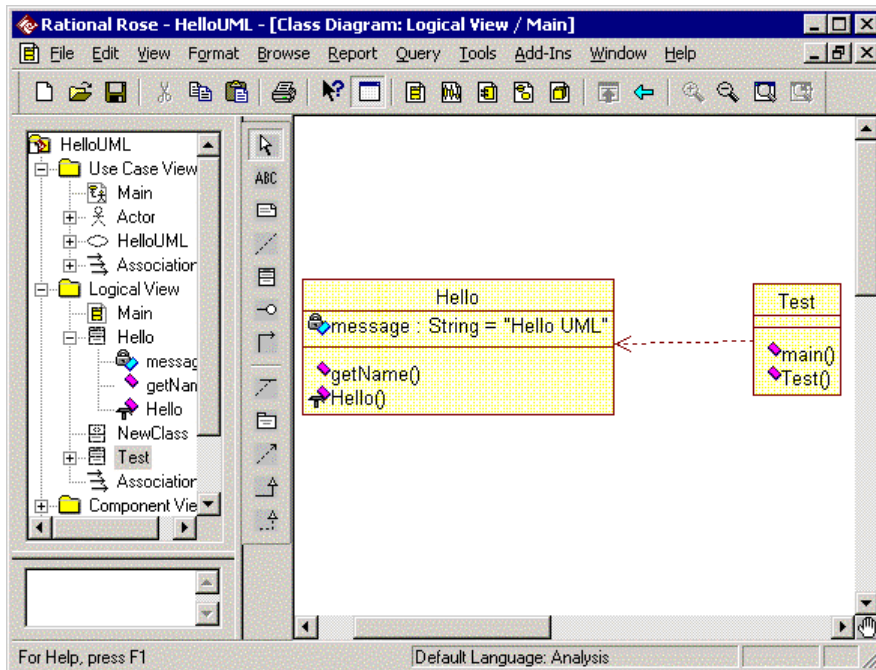
#### Step 4. Create the Test Class Diagram

1. Select Class icon from the Toolbox
2. Place a class into the workspace Class Diagram, and give the class name: **Test**
3. Right click on the Test class, and select New Operations
4. Add two methods: **main** and **Test**



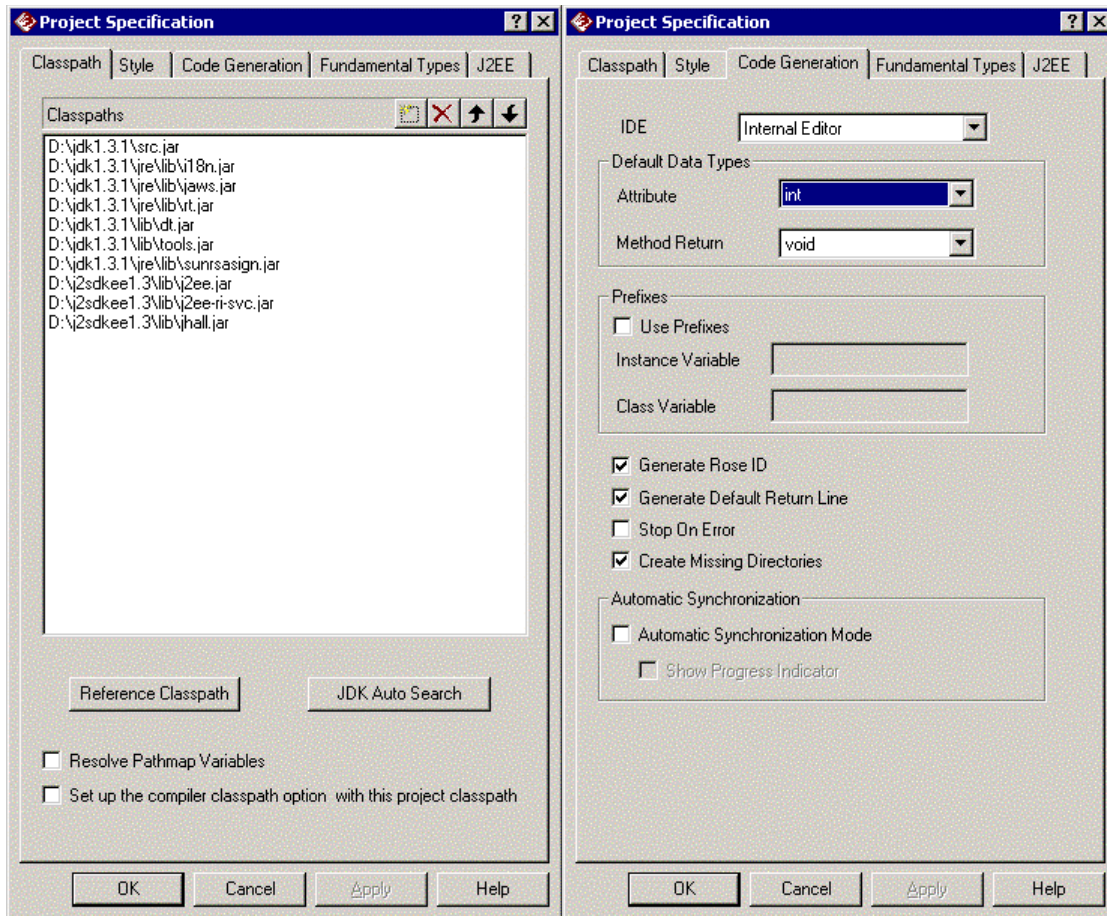
### Step 5. Object Instantiation

1. Select the dash line icon (Dependency or instaintiates)
2. Draw the line starting from the Test to Hello to make an object instantiation

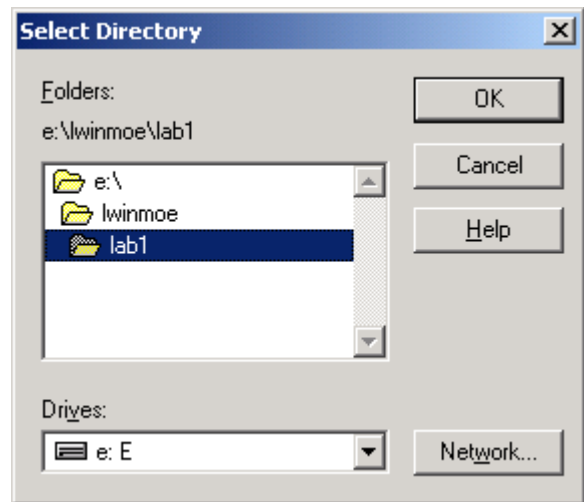
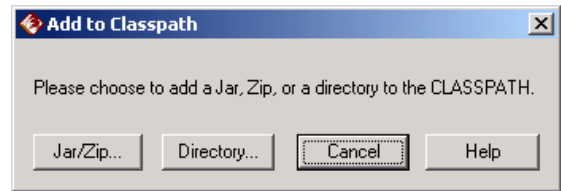
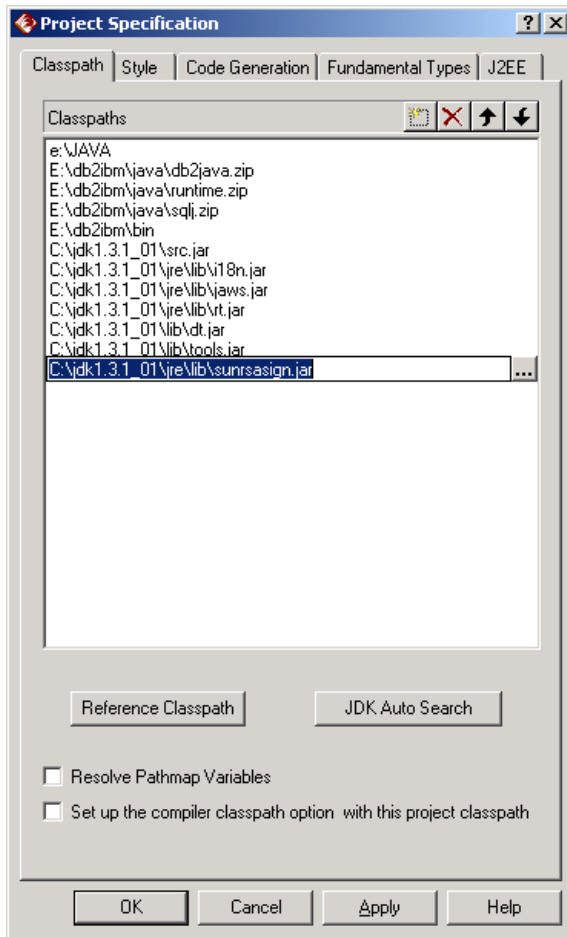


## Step 6. Code Generation

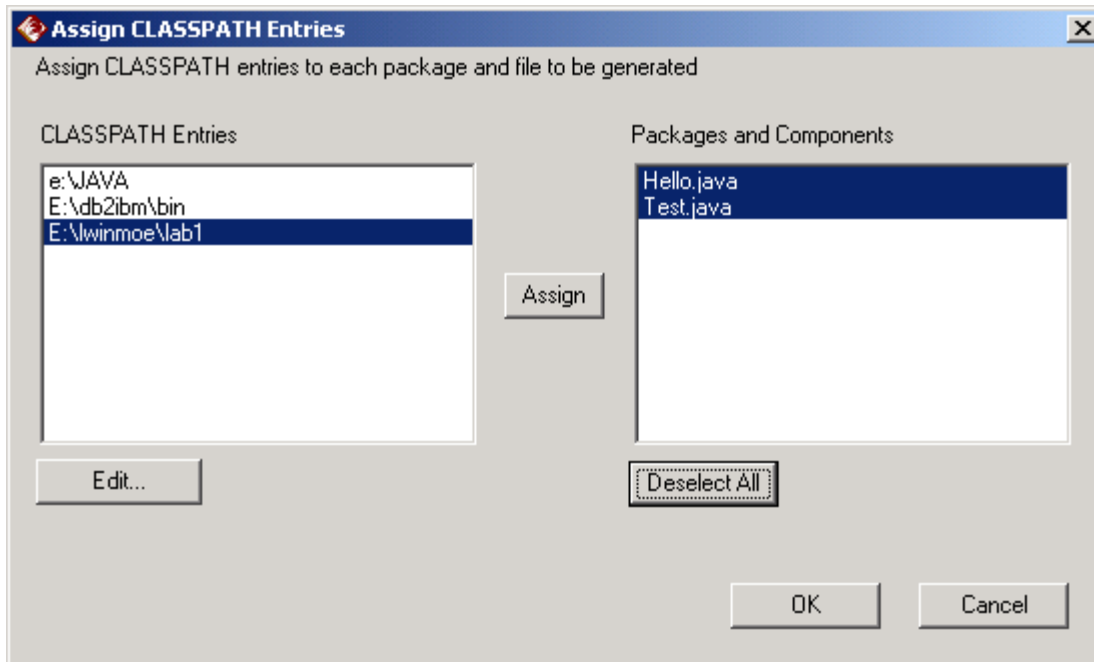
1. From the menu bar, select **Tools > Java/J2EE > Project Specification**.



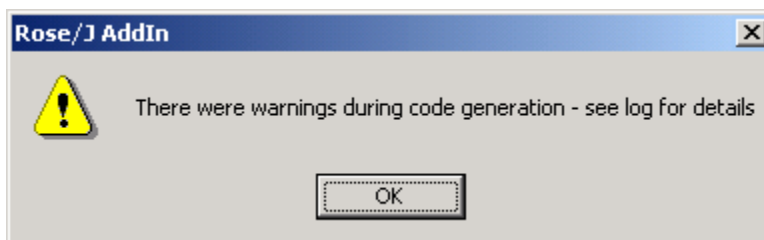
2. Double-click on the last one of the Classpaths as shown in the following diagram.
3. Click on the button with three dots to see a dialog box.
4. Choose **Directory**.



5. Select the class diagrams, **Test** and **Hello** by left clicking on **Test** first and hold down **Ctrl** key and left click on **Hello** class.
6. From the menu bar, select **Tools> Java/J2EE> Generate Code**. Choose your working directory and click “**OK**”.



The following warning message from the dialog box and **Log** windows is OK for the time being:



**20:47:43| WARNING: Class Logical View::Test - no return type has been specified for operation main - default will be used**

The following codes in **Hello.java** and **Test.java** are generated

**//Source file: C:\\My Documents\\Hello.java**

```
class Hello
{
    private String message = "Hello UML";

    /**
     * @roseuid 3C4762C801C8
     */
}
```

```

public Hello()
{
}

/**
 * @return String
 * @roseuid 3C4762A1029E
 */
public String getName()
{
    return null;
}
}

```

//Source file: C:\\My Documents\\Test.java

```

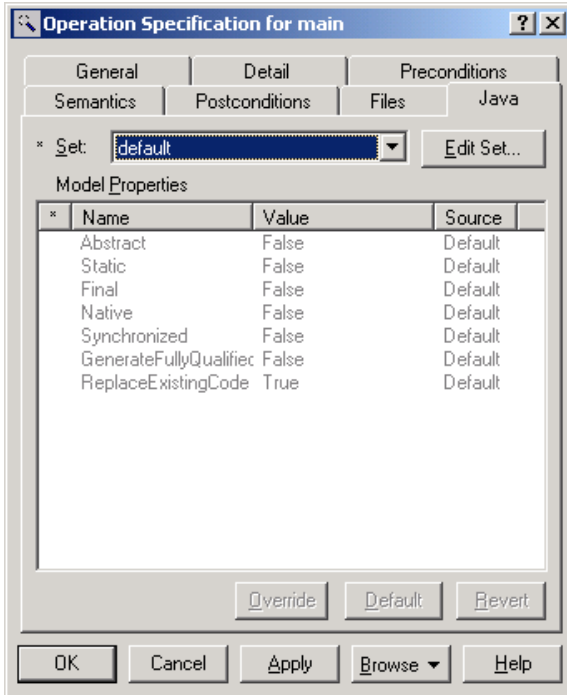
public class Test
{

    /**
     * @roseuid 3C47635B0350
     */
    public Test()
    {
    }

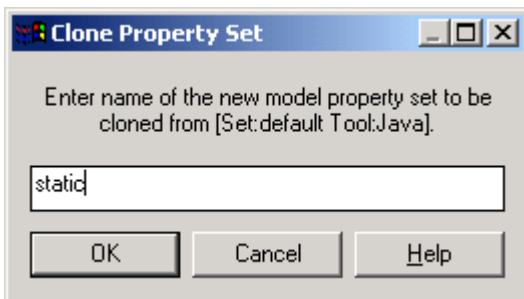
    /**
     * @roseuid 3C476355017B
     */
    public void main()
    {
    }
}

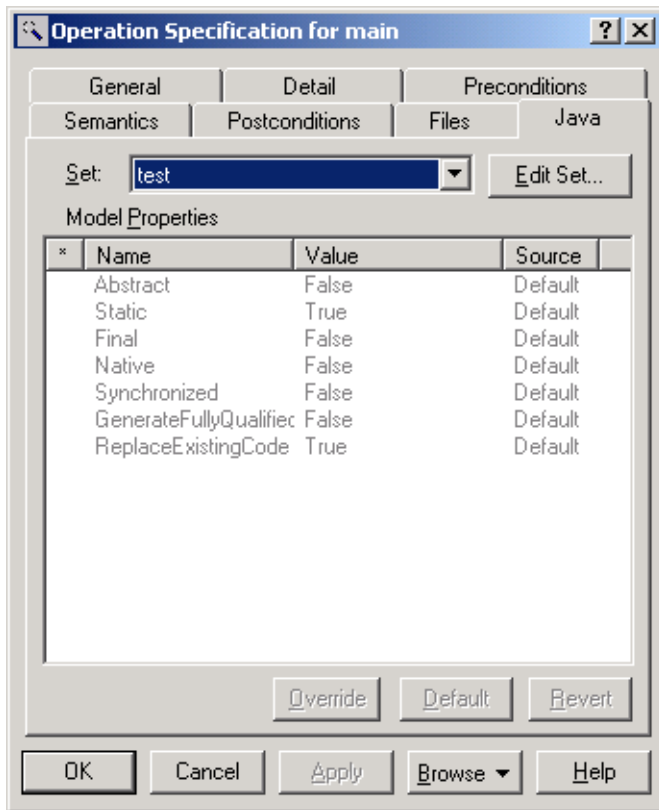
```

7. **main()** is not static method in the generated codes. To change the method to **static** type: Right click on the class, Test, the dialog box will appear. Click on **Operations** tab. Double click on **main** method and another dialog box will appear. Click on "**java**" tab and the following dialog box will appear. Click on "**Edit Set**" button.

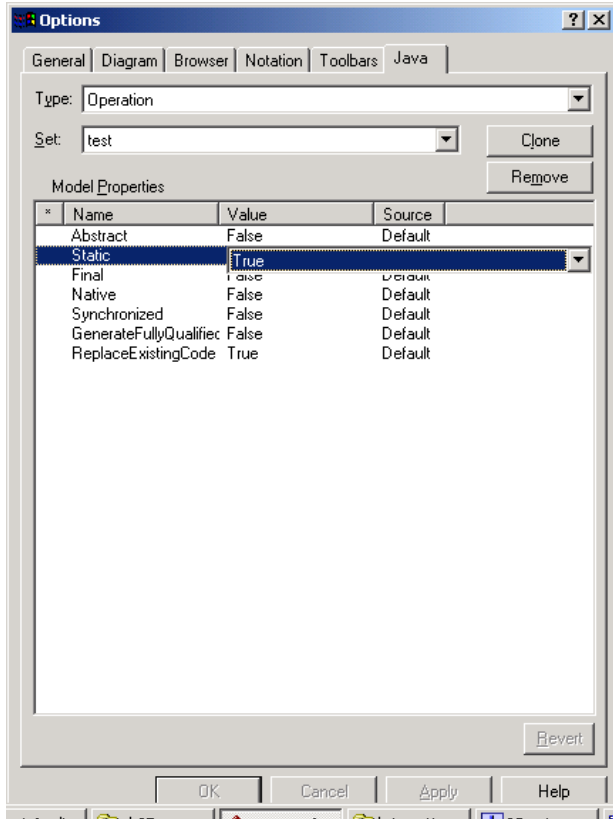


8. Create a new set “static” by cloning. Type in the name “static” in the following dialog box.





8. Change "static" to be **true**.



9. Select two classes **Test** and **Hello**. Code generate again with the command from menu bar **Tools> Java/J2EE> Generate Code** and no error message should appear this time.

### Step 7. Java Code Editing

1. Open source java files in **Notepad** and edit those files as shown below. Or use any other Development Environment such as **JBuilder** or **JCreator**. Change or add according to the highlighted words in the java source files.

**//Source file: E:\winmoe\lab1\Hello.java**

```
public class Hello
{
    private String message = "HelloUML";

    /**
     * @roseuid 3C47663F006F
     */
    public Hello()
    {
```

```

    }

    /**
     * @return String
     * @roseuid 3C47663A025C
     */
    public String getName()
    {
        return message;
    }
}

```

**//Source file: E:\lwinmoe\lab1\Test.java**

```

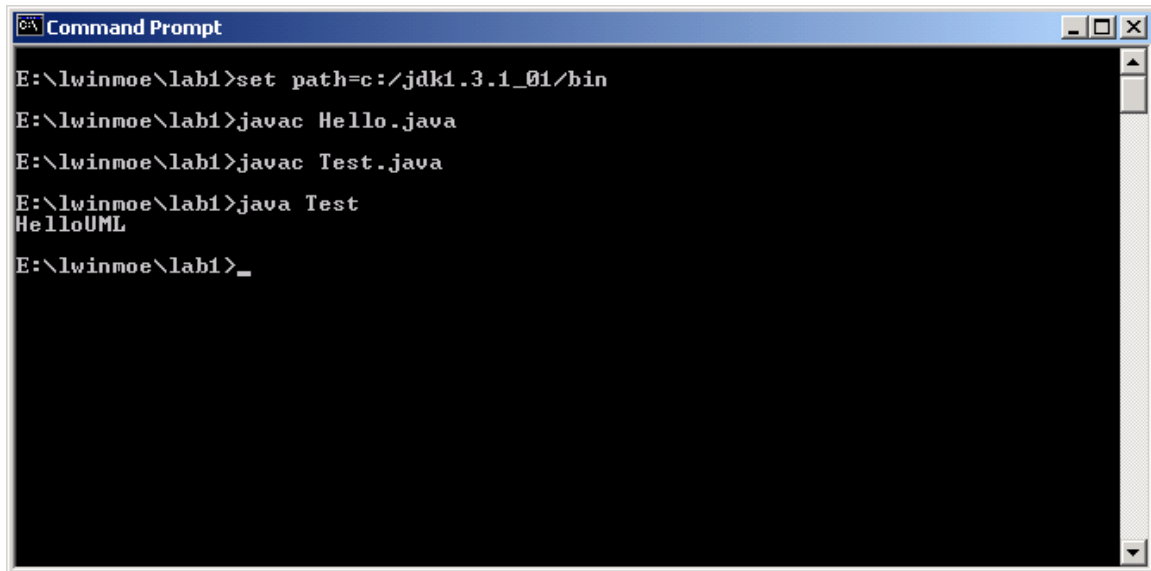
public class Test
{
    /**
     * @roseuid 3C4766960237
     */
    public Test()
    {
    }

    /**
     * @return Void
     * @roseuid 3C4766930387
     */
    public static void main(String[] args)
    {
        Hello h = new Hello(); // instantiate a Hello object called h
        System.out.println(h.getName()); // call getName() method from Hello object
        // h
    }
}

```

### Step 8. Compiling and Running Java Program from Command Line

1. To compile **Hello.java** and **Test.java** from command line, open a command shell.
2. Make sure the path is set to **c:\jdk1.3.1\_01\bin** or the directory where Java Development Kit (JDK) was installed.
3. Run **javac** program with the source file **Hello.java** with the parameter name. Do the same for **Test.java**.
4. Run your Test program with the command  
**>java Test**



```
Command Prompt
E:\lwinmoe\lab1>set path=c:/jdk1.3.1_01/bin
E:\lwinmoe\lab1>javac Hello.java
E:\lwinmoe\lab1>javac Test.java
E:\lwinmoe\lab1>java Test
HelloUML
E:\lwinmoe\lab1>_
```