Practices for Lesson 3

Practices Overview

In these practices, you examine and modify existing Java programs and also run them to test the program.

Practice 3-1: Viewing and Adding Code to an Existing Java Program

Overview

In this practice, you are given a completed Java program. You open it, examine the lines of code, modify it, compile it, and then test it by executing the program.

Assumptions

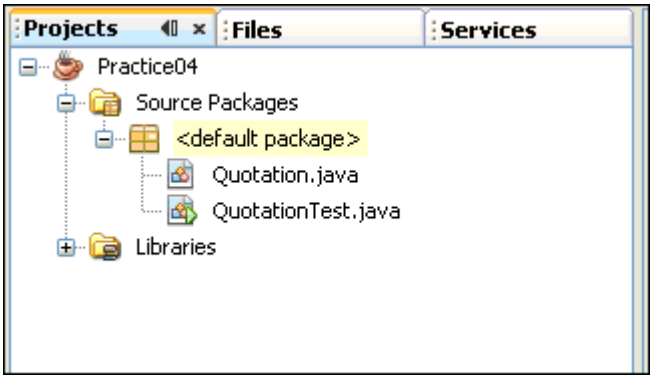
- Quotation.java and QuotationTest.java appear in the folder for this practice: `Lesson03`

Tasks

1. Create a new project from existing Java source, just as you did in Practice 1-2. The high-level steps are shown in the table below. If you need further detail, refer to Practice 1-2, steps 3 and 4.

Step	Window/Page Description	Choices or Values
a.	Menu	File > New Project
b.	New Project wizard | Choose Project step	Category: Java Project: Java Project with Existing Sources **Next**
c.	New Project with Existing Sources wizard | Name and Location step	Project Name: Practice04 **Next**
d.	New Project with Existing Sources wizard | Existing Sources step	**Add Folder**: `Lesson03` **Finish**
e.	Project Properties window | Source category	**Source/Binary Format**: JDK 7 **OK**

Note: The Projects window should now look like this when the **<default package>** node is expanded:



2. Double-click the `Quotation.java` file in the Projects window to open it for editing.

3. Identify the field and the method contained within this class, using the table below:

Member	Variable or Name
Field variable:	
Method name:	

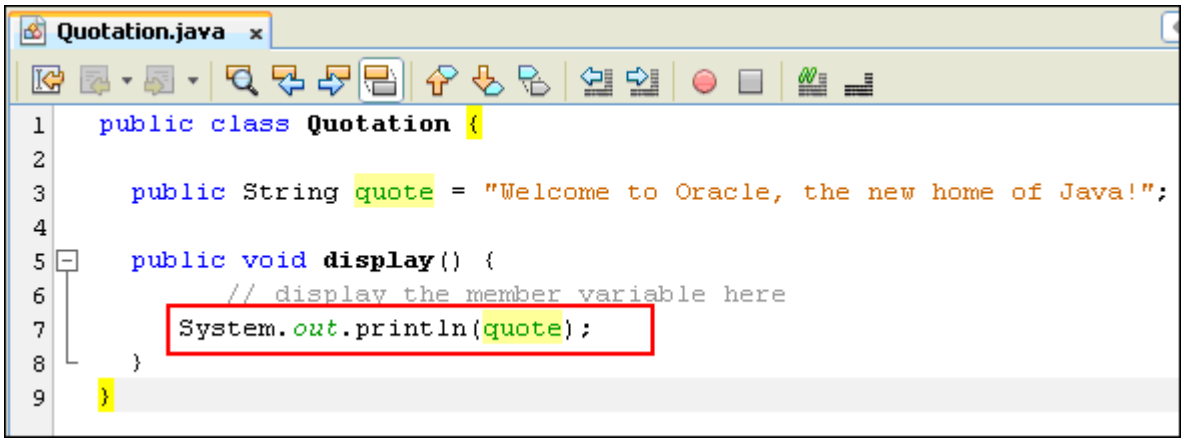
Solution: Field variable – `quote`; Method name – `display`.

4. In the `display` method, write the code to display the `quote` field. *Hint:* Use the `System.out.println` method shown in the Student Guide for this lesson. Be sure to finish the line of code with a semicolon.

   Note: You will notice, as you type the code, that NetBeans' code assist feature provides feedback and help whenever you pause in your typing. For instance, if you stop at some point at which the code, as is, would not compile successfully, it displays a red exclamation mark in the left margin. If you pause after typing the dot (".") following `System` or `out`, it gives you context sensitive help in the form of a list of methods and fields that would be valid for the particular class to the left of the dot. You can select from the list instead of typing.

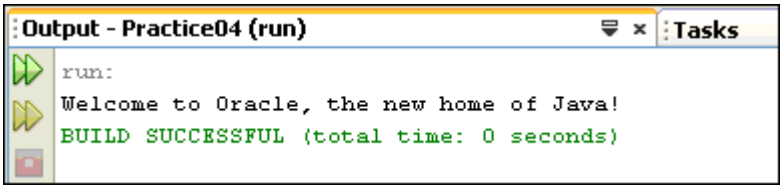
Solution:

```
System.out.println(quote);
```



5. Click the Save button to save and compile Quotation.java.
6. Open the QuotationTest.java file in the editor and examine its `main` method. It creates an instance of the Quotation class and then calls its `display` method.
7. Run the QuotationTest class by right-clicking QuotationTest.java in the Projects window and selecting **Run File**. The output from the display method appears in the Output window.

   Note: You were able to skip the Compile step because when you select Run File, NetBeans first compiles not only the class you selected to run, but also any referenced classes within that class (`Quotation.java`).



8. Edit the `Quotation.java` file now to change the default value of the `quote` field.

9. Run QuotationTest again to verify the output.
10. In the Editor pane, close `Quotation.java` and `QuotationTest.java`.

Practice 3-2: Creating and Compiling a Java Class

Overview

In this practice, you create a Java class and compile it. You also create another Java class to test the previous class.

Assumptions

None

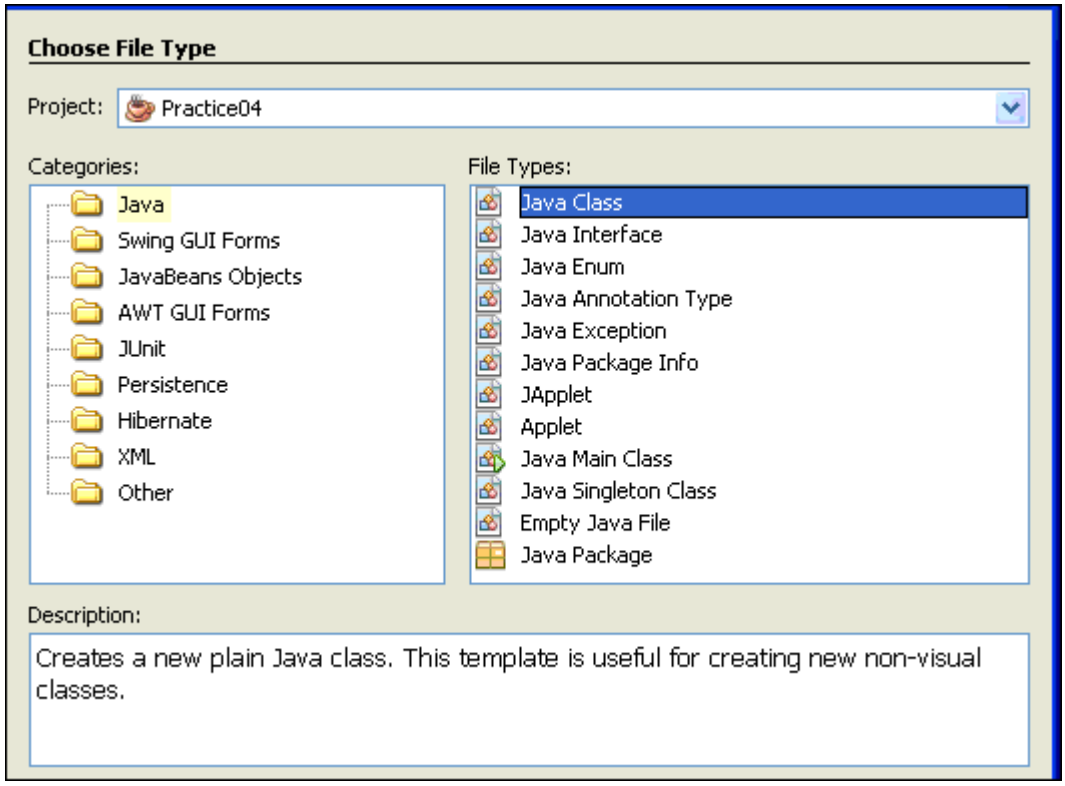
Tasks

1. Create a new Java class in the Practice03 project using the NetBeans wizard. The high-level steps for this task are shown in the table below. If you need more assistance, you can use the detailed steps that follow the table.

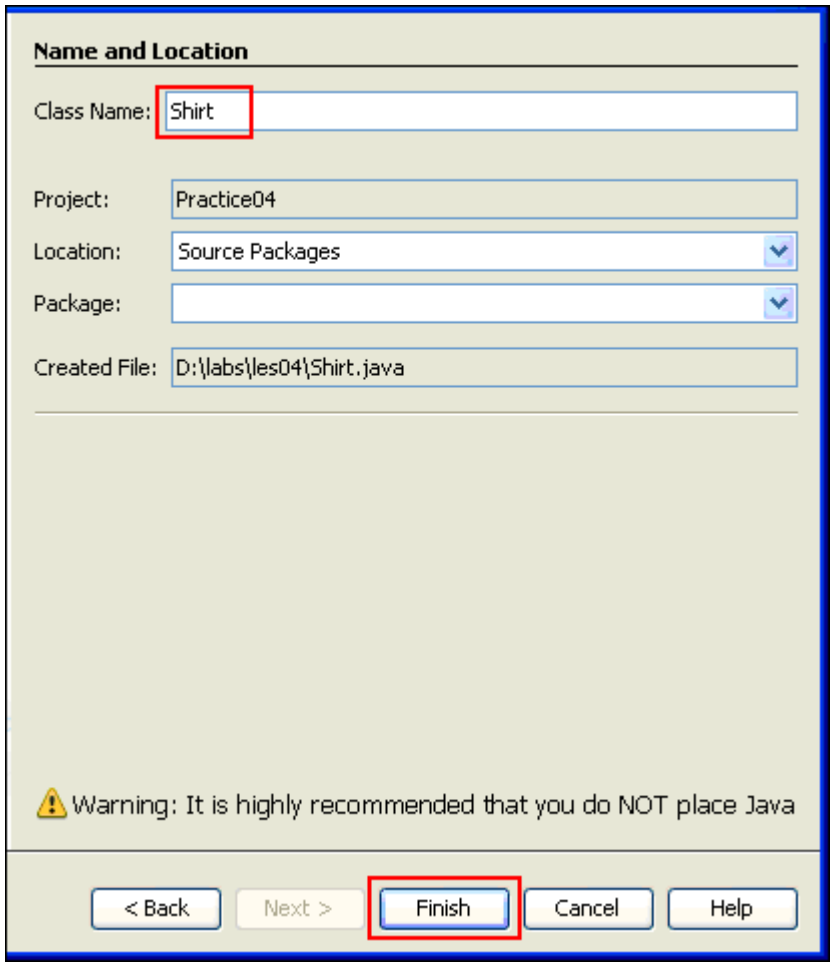
Step	Window/Page Description	Choices or Values
a.	Menu	**File > New File**
b.	New File window | Choose File Type step	Category: Java File Types: Java Class **Next**
c.	New Java Class window | Name and Location step	Class Name: `Shirt` **Finish**

   a. From the main menu, select **File > New File**.

   b. The New File wizard opens and you are on step 1 “Choose File Type”. Select **Java** in the Category column. Select **Java Class** in the File Types column. Click **Next**.



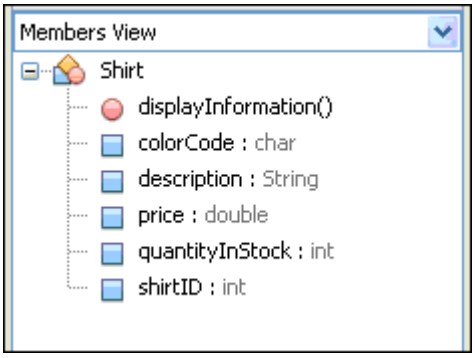
c. In the New Java Class window, you are on step 2 “Name and Location”. Enter “Shirt” as the Class Name. Click **Finish**.



The Java source file for the new class now appears in the editor ready for you to fill in the details.

2. Enter the Java code syntax for the Shirt class shown in this lesson of the Student Guide.
3. Click the Save button to save and compile the Shirt class. Any red error icons in the left margin should disappear after saving if there were no compilation errors. If necessary, fix any errors that appear in the Output window and save again.

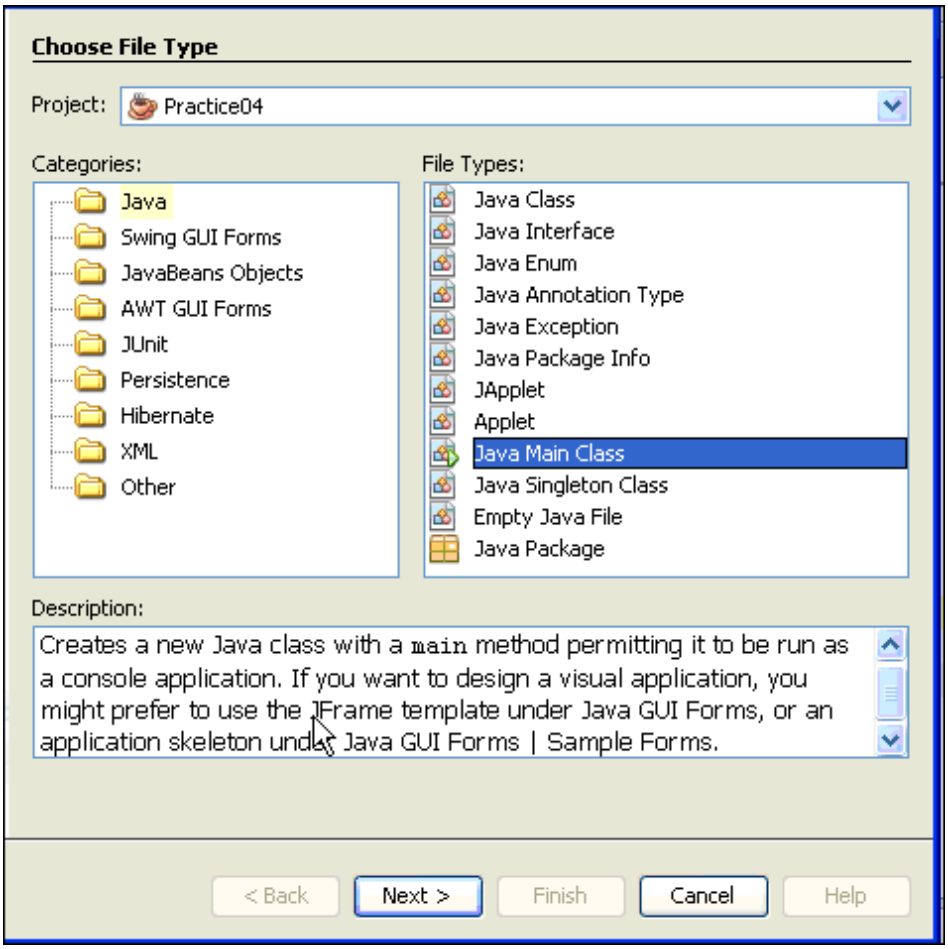
   Note: The Navigator pane (lower left corner of NetBeans) for the `Shirt` class now shows the Members view of the class. Notice the color coding that distinguishes between fields and methods. Both of these are considered “Members” of the class.



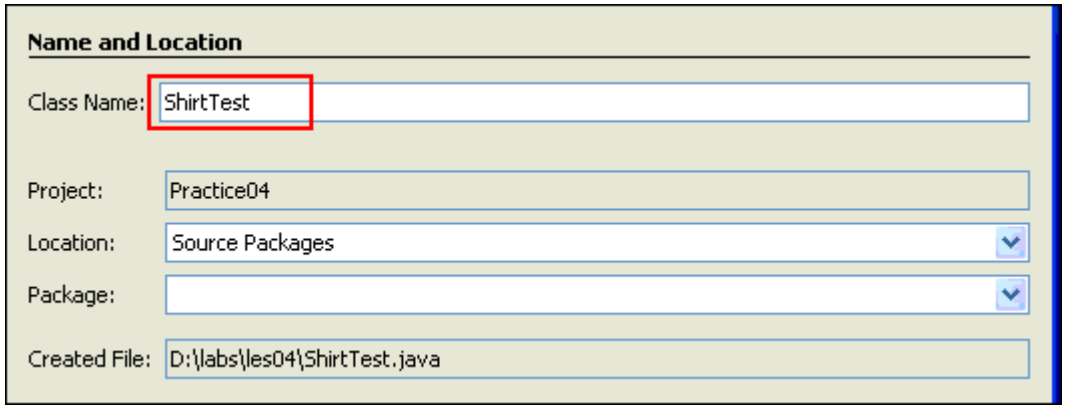
4. Follow the instructions from Step 1 to create another new class. This will be a Test class, so it will need a `main` method. To accommodate that change, the table below shows the substitutions in the Step 1 instructions you should make as you go through the New Class wizard. For more detail, see the screenshots following the table.

Step	Window/Page Description	Choices or Values
a.	New File window | Choose File Type step	File Types: **Java Main Class**
b.	New File window | Name and Location step	Name: **ShirtTest**

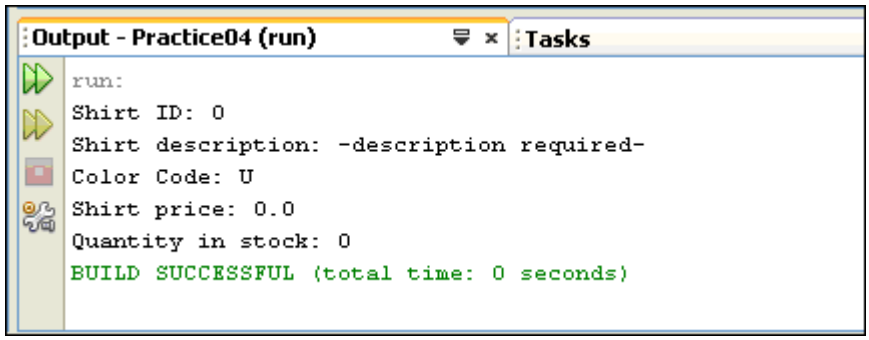
a. In the Choose File Type step, select **Java Main Class** instead of Java Class.



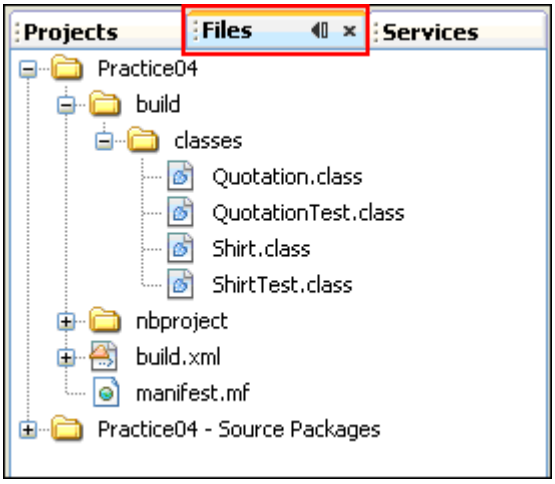
b. In the Name and Location step, enter **ShirtTest** as the name.



5. Replace the **To Do:** comment in the `main` method with the two lines of code that appear in the `main` method for the ShirtTest class shown in this lesson of the Student Guide.
6. Save and compile the code by clicking Save.
7. Run the ShirtTest class by right-clicking ShirtTest.java in the Projects window. Look for the output of the `displayInformation` method in the Output window.



8. Find the class files that were generated by NetBeans when you ran the program. Click the Files tab to open the Files window and find Shirt.class and ShirtTest.class as shown below.



9. Open (or return focus to) the `Shirt.java` file. Modify the values of `ShirtID` and `price`.
10. Run the ShirtTest class again. Verify that the modified values are shown in the Output window.

Practice 3-3: Exploring the Debugger

Overview

Virtually every Java IDE provides a debugger. They tend to offer the same core features and work very similarly. In this practice, you debug the ShirtTest program using the NetBeans debugger. You set breakpoints, examine field values, and modify them as you step through each line of code.

Assumptions

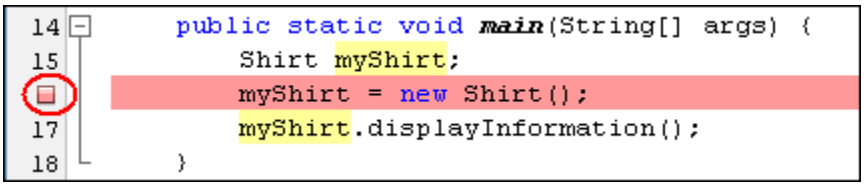
None

Tasks

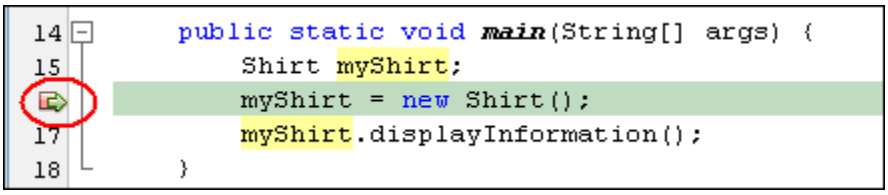
1. Set a breakpoint in the ShirtTest class. Click in the left margin of the editor, next to the following line of code:

   ```
   myShirt = new Shirt();
   ```

   A pink square appears in the margin, indicating a breakpoint.



2. Run the debugger by right-clicking on the ShirtTest file in the Projects window and selecting **Debug File**.
3. The debugger starts the program and stops at the breakpoint. In the Editor panel you should now see a different icon that points with a green arrow to the line of code.

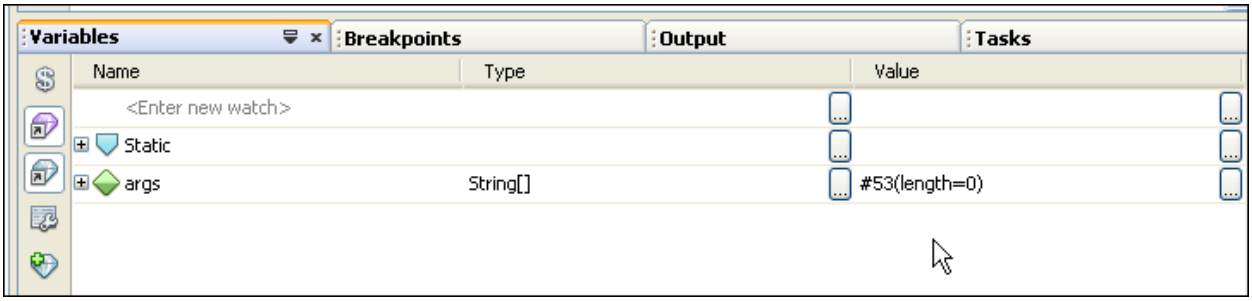


   This line of code has not yet been executed.
4. Several other changes have occurred in the NetBeans window.
   - A new toolbar appears, containing buttons that you use when debugging.

     - Move your cursor over each of the toolbar buttons to read the toolbar tip explaining what each button does. The buttons are described below.
       - The first button, **Finish Debugger Session**, stops the debugging session.
       - The second button, **Pause**, pauses the execution of the debugger.
       - The third button **Continue** the execution, either to the next breakpoint or to the end of the program.
       - The fourth button, **Step Over**, moves the program forward to the next line of code in the current class (in this case, the ShirtTest class).
       - The fifth button, **Step Over Expression**, allows you to step over an entire expression to the next line of code in the current class.

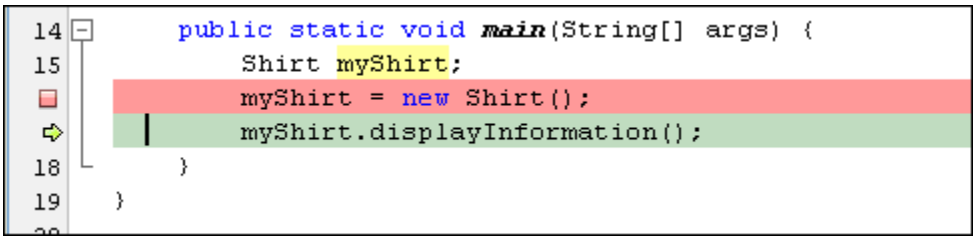
- The sixth button, **Step Into**, allows you to step into another class referenced in this current line of code.
- The seventh button, **Step Out**, allows you to step back out of a class that you stepped into.
- The last button, **Run to Cursor**, takes execution to the line of code where the cursor appears.

- The panel at the bottom of the window changes to show debugging output and variables and other useful information during a debug session.



- In the Variables panel, you see all variables that are visible to the current class. Remember that the execution was stopped *before* the Shirt class object has been instantiated. Consequently, you do not see the `myShirt` variable in this panel.

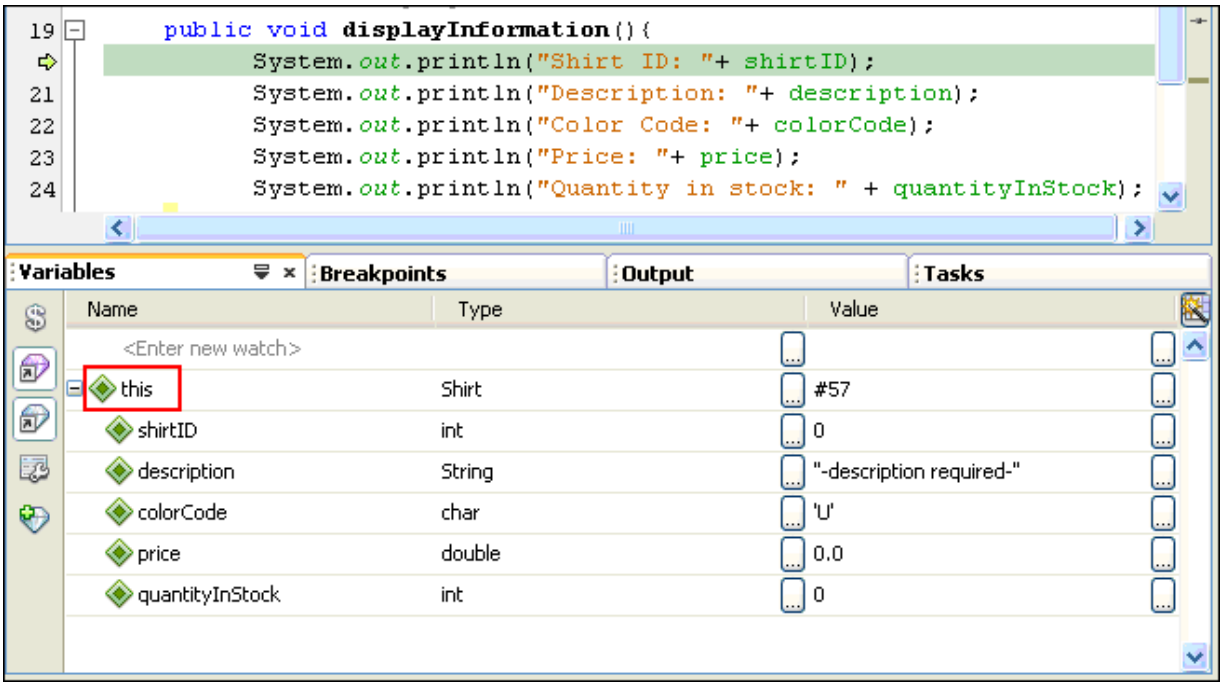
5. Click the Step Over button to move to the next line of code. 
6. The arrow now points to the line of code that calls the `displayInformation` method on the `myShirt` object. In the Values window, you now see the `myShirt` variable. Expand it to see all of the fields of this `Shirt` object.



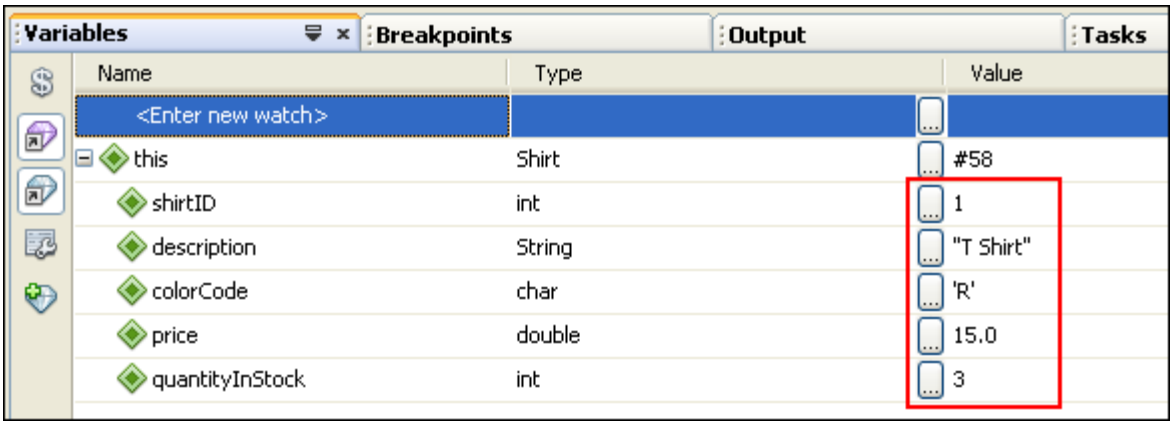
At this point, the `displayInformation` method has not yet been executed. You could change the values of the object's fields right now, using the Variables window if you wanted to. However, instead, you "step into" the `myShirt` object and change the values during the execution of the `displayInformation` method.

7. Click the Step Into button to step into the `displayInformation` method. 

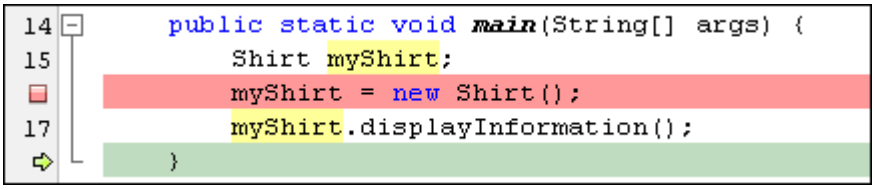
8. The arrow icon is pointing to the first executable line of code within the `displayInformation` of the `Shirt` class. In the Variables window, expand **this** to see the fields of this object.



9. In the Value column double-click each field's value and edit it to change the value. Ensure that you use the correct value for the data type expected and enclose any character data types with the type of quote mark indicated. After editing the final field, click the tab button so that the text you typed into the edit buffer is accepted.

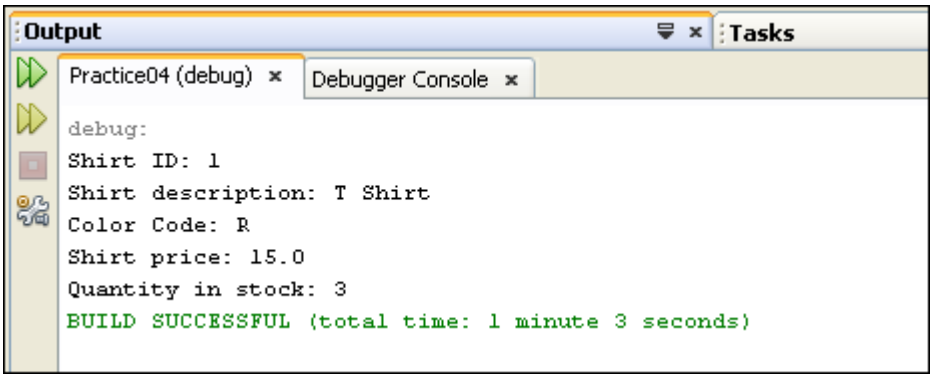


10. Click the Step Out button to return to the next line of code in the ShirtTest class. The `displayInformation` method will have completed.



11. Notice that the `myShirt` object field variables reflect the changes you made while in the method.
12. Click the Continue button now to finish execution and end the debug session.

13. Click the Output tab to view the output.



You have now experienced some of the most commonly used features of a typical IDE Debugger. You may wish to use the debugger in remaining labs to help you diagnose and fix problems you may experience in your programs.

14. Close the Practice04 project in NetBeans. In the Projects window, right-click Practice04 and select **Close** from the context menu.