

Testing

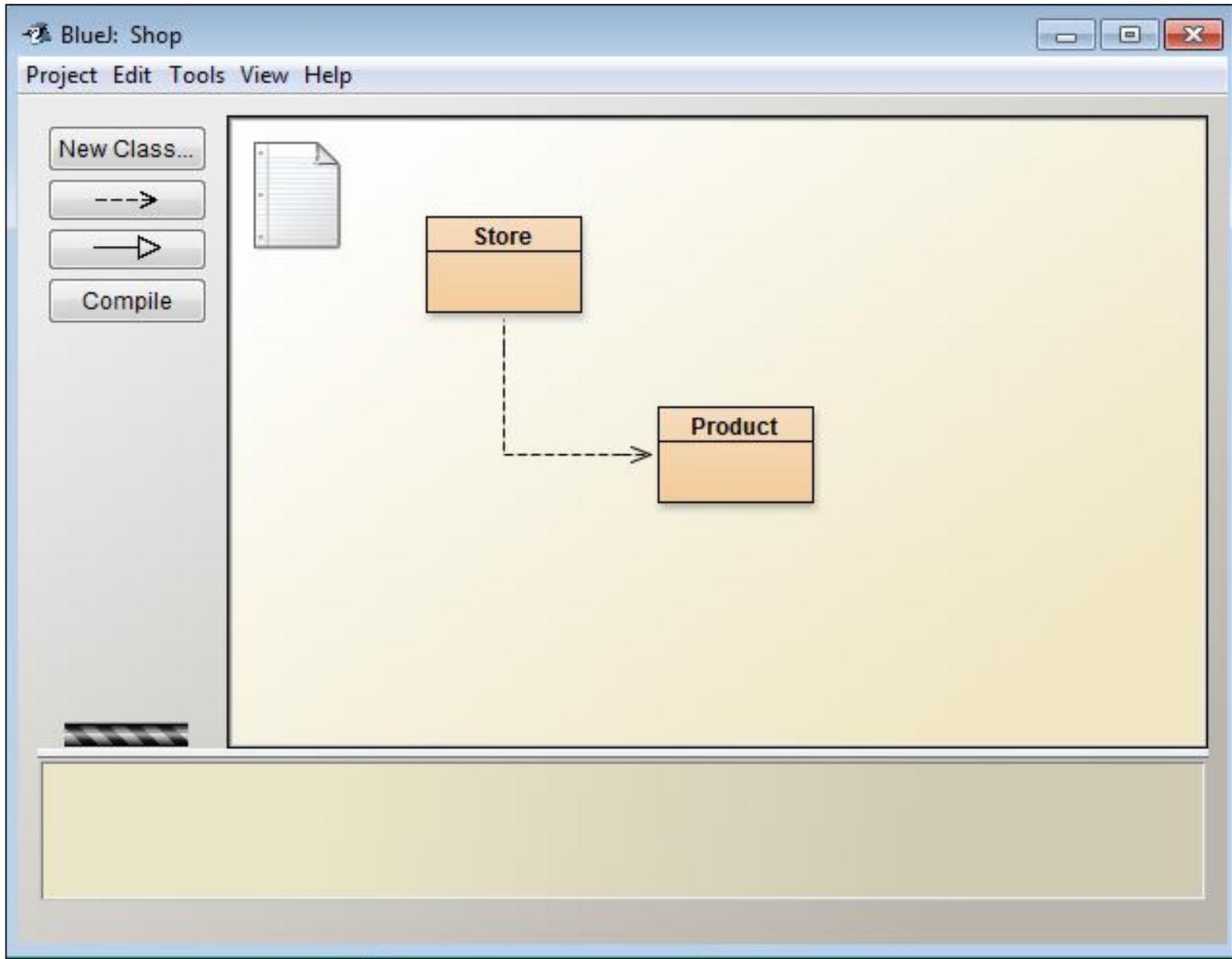
Test Fixture to Object Bench (and vice versa)

Produced by: Dr. Siobhán Drohan



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>



In the usual way, create three Product objects and one Store object

The screenshot shows a Java IDE interface. On the left, there is a sidebar with buttons: "New Class...", a dashed arrow, a solid arrow, and "Compile". The main workspace displays two class diagrams: "Store" and "Product". A dashed arrow points from "Store" to "Product". A dialog box titled "BlueJ: Create Object" is open in the foreground. It contains the following text:

```
// Constructor for objects of class Product  
// @param productName Name of the product  
// @param productCode Code of the product  
// @param unitCost Unit cost of the product  
Product(String iProductName, int iProductCode, double iUnitCost)
```

Below the code, there is a text field for "Name of Instance:" containing "product1". Underneath, the constructor is being invoked with three arguments in dropdown menus:

```
new Product ( "38 Inch TV" , String iProductName  
             1000 , int iProductCode  
             345.99 ) double iUnitCost
```

At the bottom of the dialog are "Ok" and "Cancel" buttons.

Add each Product to the Store
(by calling the **add** method).

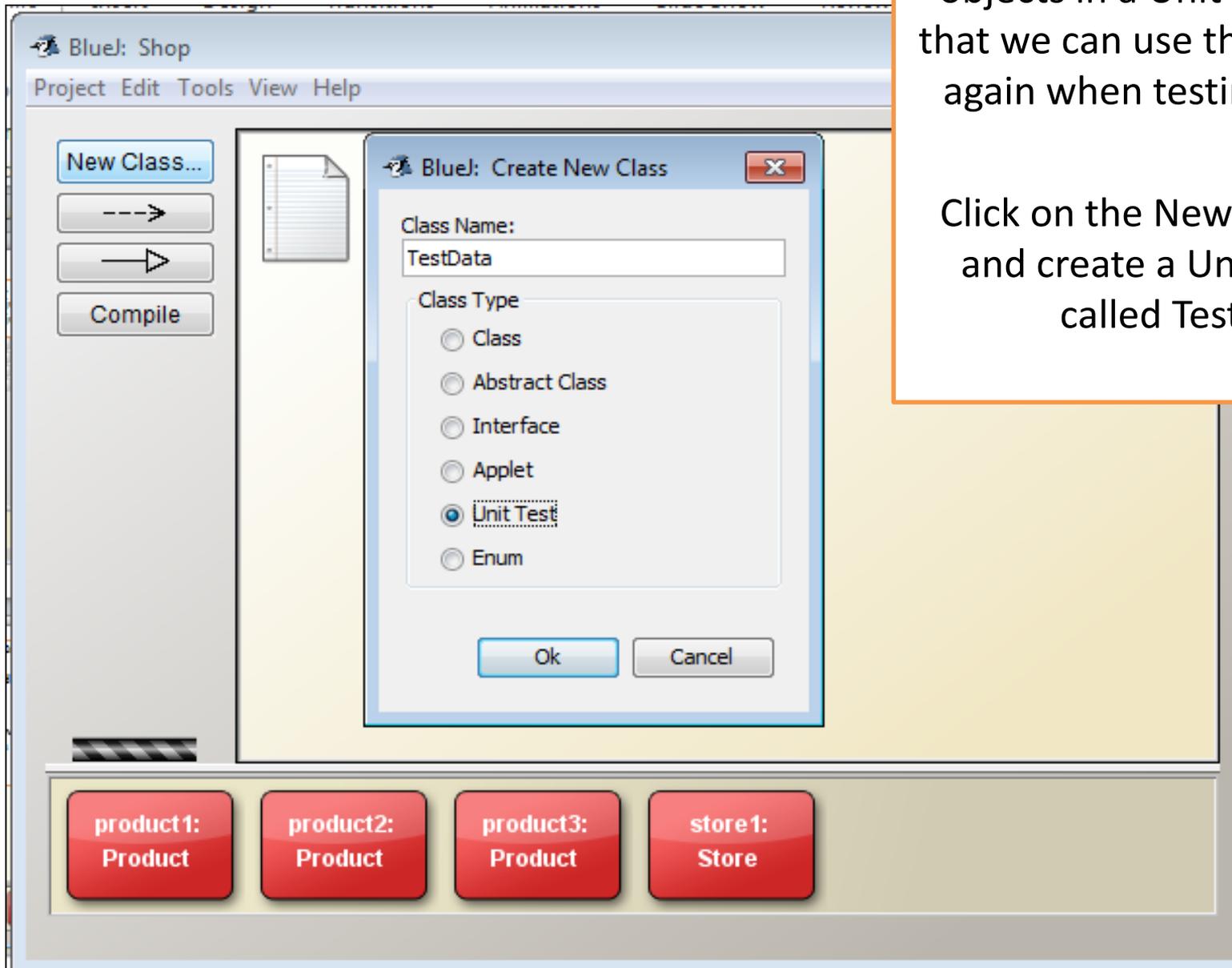
Your Store will now hold three products in its ArrayList.

The image shows a screenshot of an IDE. At the top, a class diagram shows a `Store` class with a dashed arrow pointing to a `Product` class. Below this, a Java code editor displays the following code:

```
store1 : Store  
product1: Product  
product2: Product  
product3: Product  
store1.add(product1)  
store1.add(product2)  
store1.add(product3)  
store1.listProducts()
```

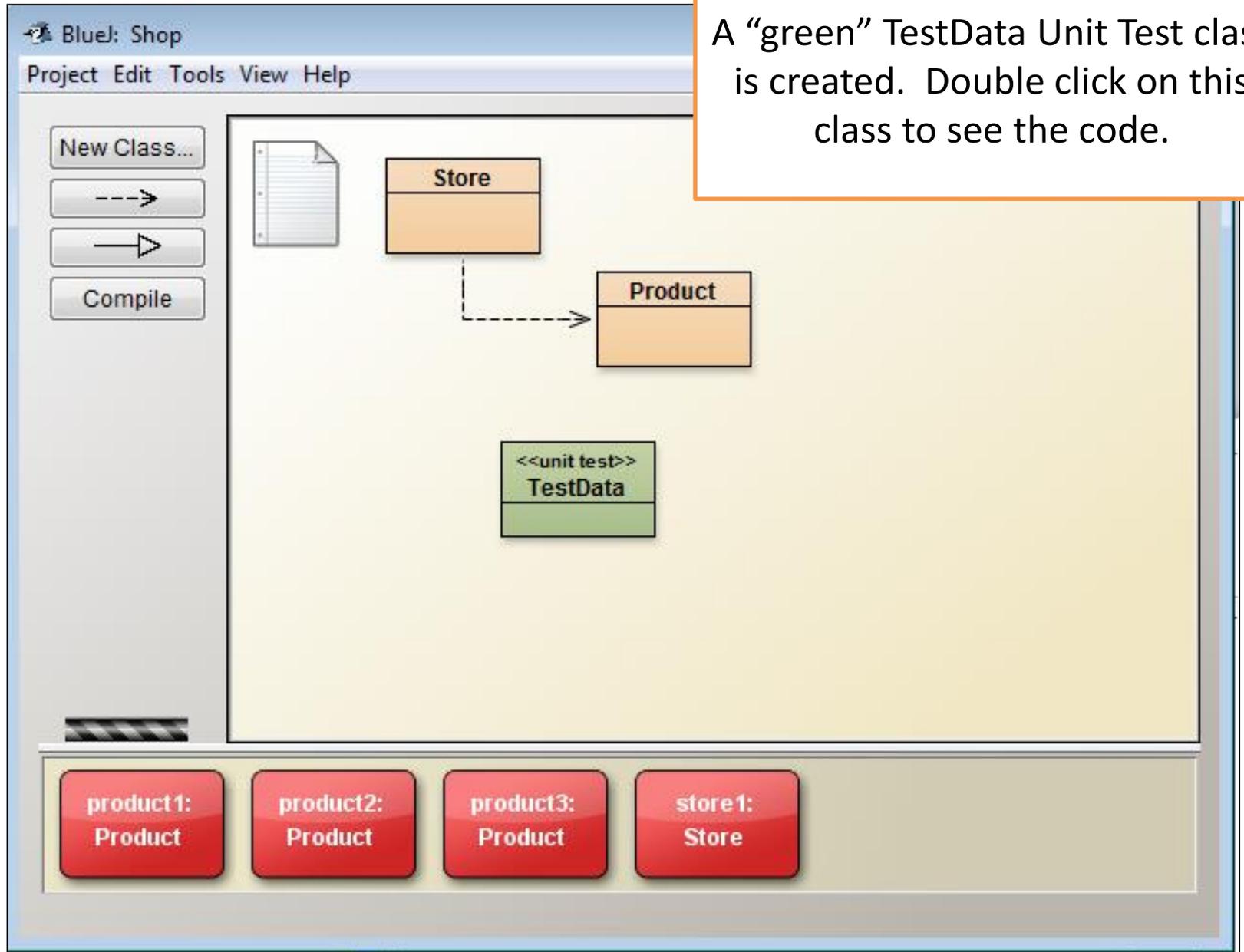
A context menu is open over the code, showing the following options:

- inherited from Object*
- `void add(Product product)`
- `String cheapestProduct()`
- `void listProducts()`
- Inspect*
- Remove*



We will now “save” these objects in a Unit Test class, so that we can use them again and again when testing our code!

Click on the New class button and create a Unit Test class called TestData.



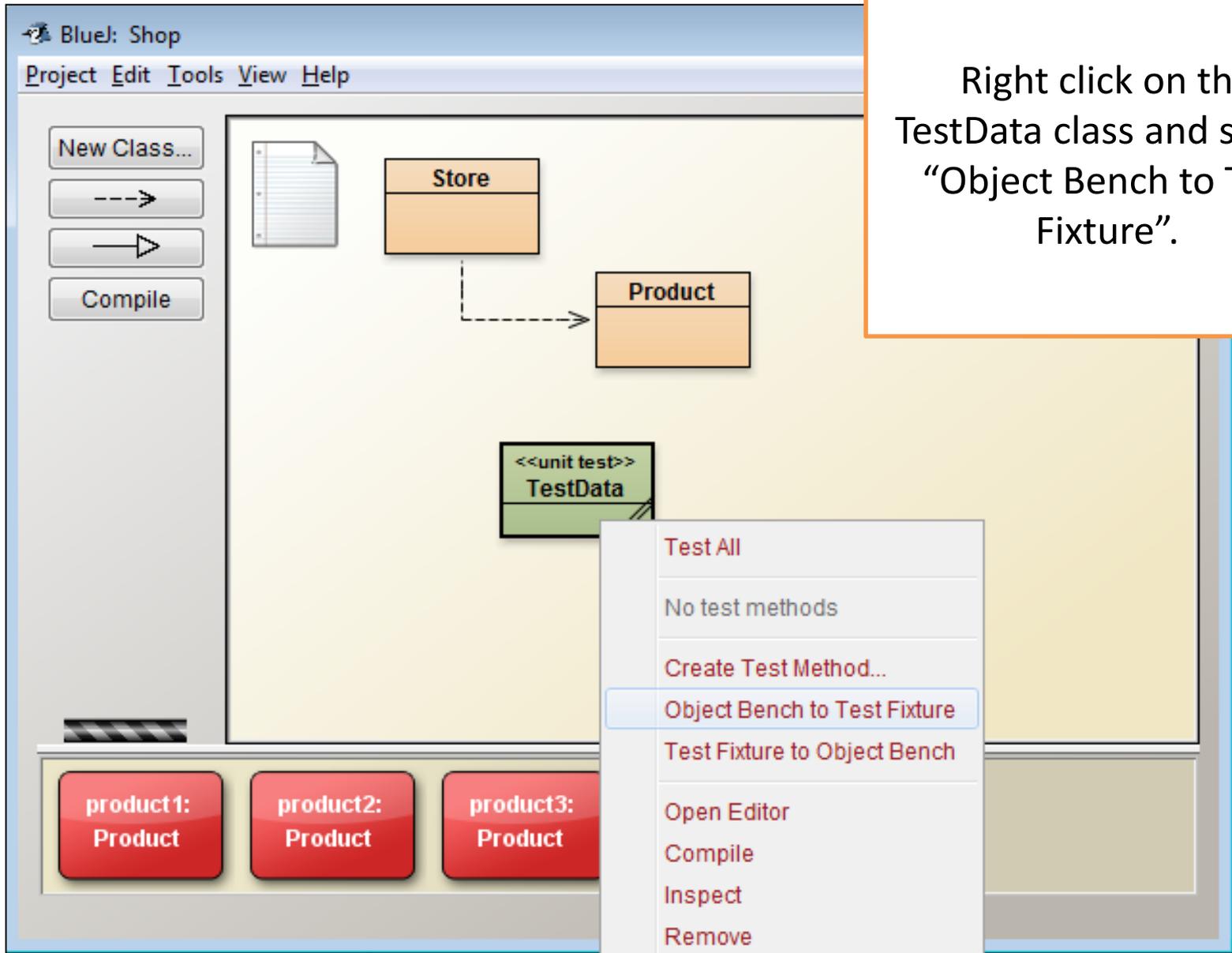
A "green" TestData Unit Test class is created. Double click on this class to see the code.

This is the generated code in the test class.

At the moment we are not interested in the methods created and the terms used e.g.: test fixture, test case, junit, etc. We will cover these later on in the semester or early next semester.

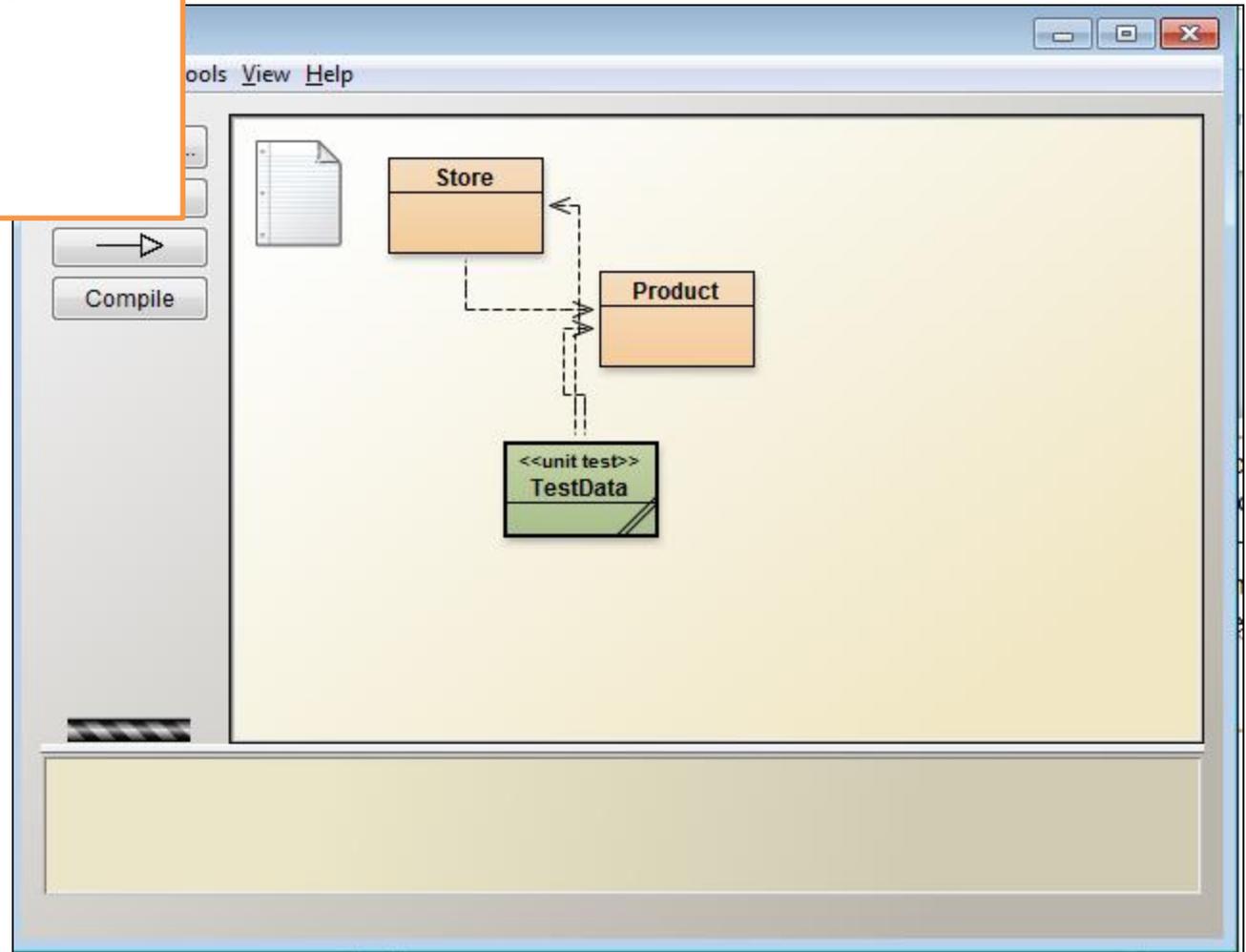
We just want to use this facility to save us time when testing our code.

```
Compile Undo Cut Copy Paste Find... Close
9  * @author (your name)
10 * @version (a version number or a date)
11 */
12 public class TestData
13 {
14     /**
15      * Default constructor for test class TestData
16      */
17     public TestData()
18     {
19     }
20
21     /**
22      * Sets up the test fixture.
23      *
24      * Called before every test case method.
25      */
26     @Before
27     public void setUp()
28     {
29     }
30
31     /**
32      * Tears down the test fixture.
33      *
34      * Called after every test case method.
35      */
36     @After
37     public void tearDown()
38     {
39     }
40 }
```



Right click on the
TestData class and select
“Object Bench to Test
Fixture”.

- Your objects disappear from the Object Bench.
- The objects were “moved” into the TestData class.



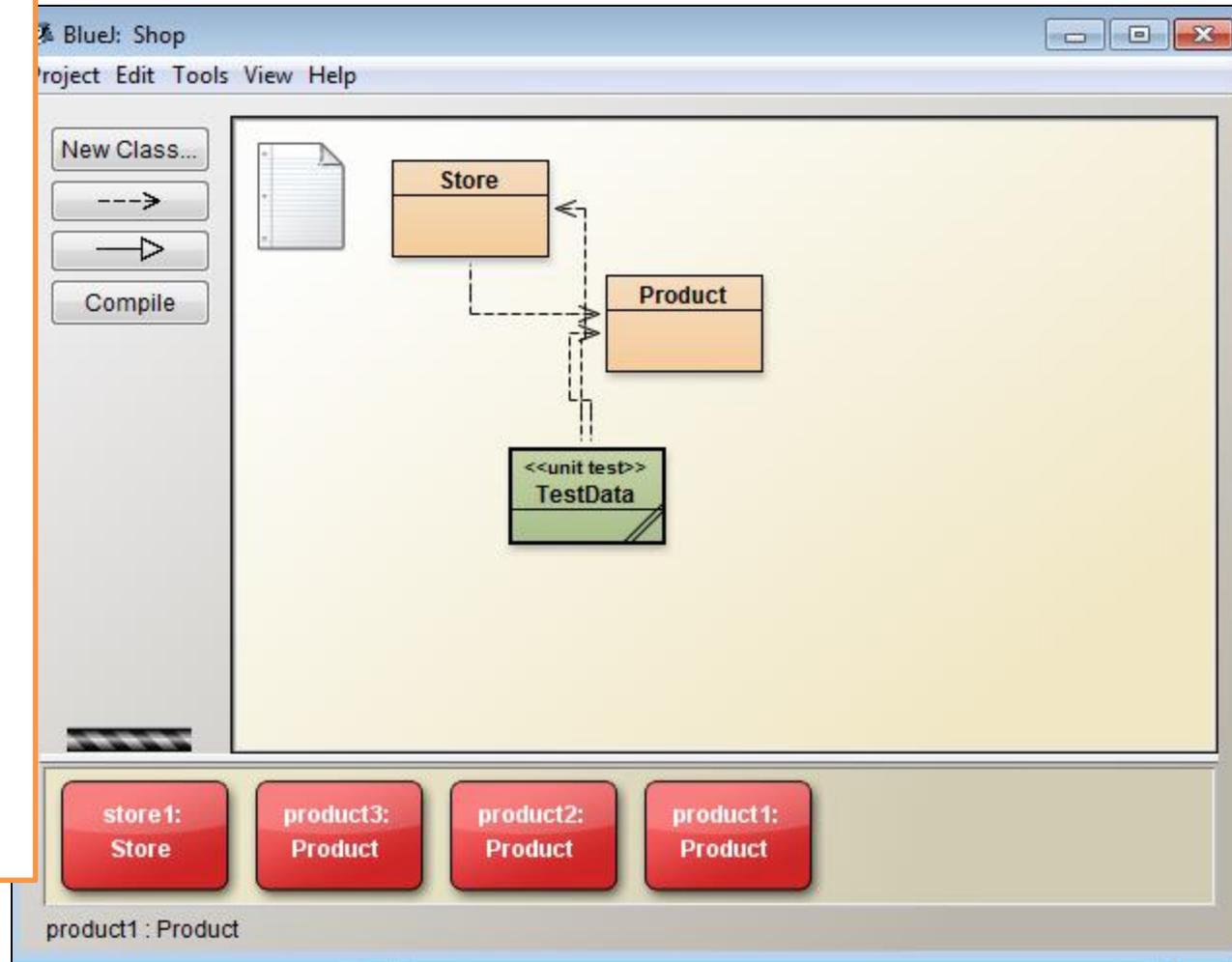
```
TestData
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
12 * @version (a version number or a date)
13 */
14 public class TestData
15 {
16     private Product product1;
17     private Product product2;
18     private Product product3;
19     private Store store1;
20
21     /**
22      * Default constructor for test class TestData
23      */
24     public TestData()
25     {
26     }
27
28     /**
29      * Sets up the test fixture.
30      *
31      * Called before every test case method.
32      */
33     @Before
34     public void setUp()
35     {
36         product1 = new Product("38 Inch TV", 1000, 345.99);
37         product2 = new Product("40 Inch TV", 1001, 399.99);
38         product3 = new Product("42 Inch TV", 1002, 499.99);
39         store1 = new Store("Waterford");
40         store1.add(product1);
41         store1.add(product2);
42         store1.add(product3);
43     }
44 }
```

Double click on the green TestData class and note the changes; the contents from the Object Bench have been moved into the code.

When you make code changes to your Store or Product class, you do not have to manually create objects; instead you can use the ones that you just moved into the green TestData class.

To do this:

- Right click on the green TestData class
- Select “Test Fixture to Object Bench”
- Your objects will be copied from your test fixture to your object bench.



Questions?





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>