3 Type the following code into the afterFactory( ) method:

```
importPackage( Packages.javax.swing );
countOfMinis = reportContext.getPersistentGlobalVariable(
   "cmKey").intValue();
frame = new JFrame( "Count of Minis = " + countOfMinis );
frame.setBounds( 310, 220, 300, 20 );
frame.show( );
```

4 Select Preview to see the results. If there were no errors in the code, you see a report similar to the one in Figure 9-18.

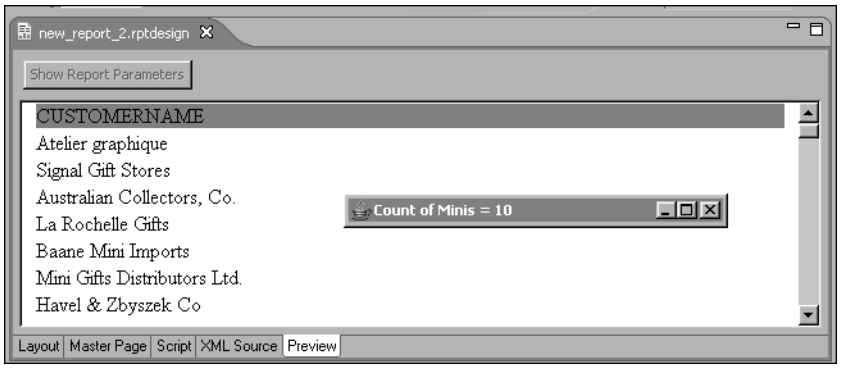

Figure 9-18 Result of changing the afterFactory( ) method

If you do not see the Count of Minis window, look for it behind the Eclipse window. If the Count of Minis window does not appear, the most likely reason is a scripting error caused by an error in one of your code entries. If you suspect that a scripting error has occurred, scroll to the bottom of the report, where all scripting error messages appear. In most situations, there is a brief error message next to a plus sign (+). The plus sign indicates that there is a more detailed error message that you can view. To expand the brief error message, choose the plus sign. Scroll down to see the more detailed error message.

Calling Java from JavaScript

Rhino provides excellent integration with Java classes, allowing BIRT scripts to work seamlessly with business logic written in Java. Wrapping Java in JavaScript allows the developer to write powerful scripts quickly by leveraging both internal and external libraries of existing Java code. You can use static methods, non-static methods, and static constants of a Java class.

Understanding the Packages object

The Packages object is the JavaScript gateway to the Java classes. It is a top-level Rhino object that contains properties for every top-level Java package, such as java and com. Packages also contains a property for every package that it finds