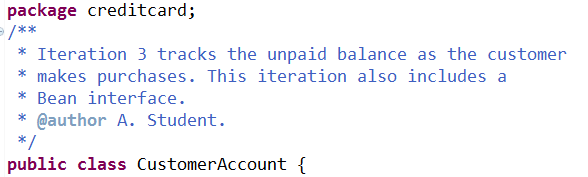
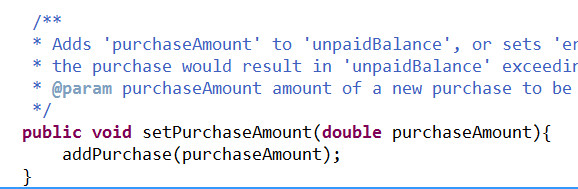
CustomerAccount becomes a JavaBean

## CSCI 4300

**Before you start**: You should have completed the second iteration of the credit card project, which adds purchases or creates an error message, and created a JSP page as a Web view for the **CustomerAccount** class.

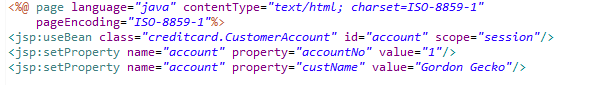
Ready to write some Java code? Let’s go!

1. Launch Eclipse, and open the CreditCard project . Bring up your **CustomerAccount** class in your editor. Modify the class comment to clarify what to change, in case we forget (or don’t clearly understand):To make CustomerAccount into a JavaBean, we need to define the properties exchanged between JSP and Bean:
   1. **Properties read by the JSP page from the Bean**: every piece of data that the page must display. This includes all of the customer data, the credit limit and unpaid balance, and the error message. Your should already have getter methods for all of these properties:
   2. **Properties sent by the JSP page to the Bean**: the amount of a new purchase. We already have a method named addPurchase that does this, but JavaBeans must express all inputs and outputs as getter and setter methods. This is easily accomplished by *wrapping* addPurchase with a setter method:

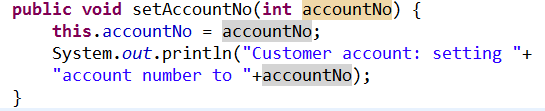
Note that the spec for the new setter method is nearly the same as the addPurchase spec, because the methods do the same task! Remember that method specs should tell what the method accomplishes, not the implementation details of how the method accomplishes its job (in this case, calling another method).

Setter methods must return void, so we have dropped the return type from setPurchaseAmount. I would recommend against simply refactoring addPurchase into setPurchaseAmount, since we use the Boolean return value from addPurchase in the test code.

1. Of course you will want to write test code for this new method, won’t you? Since the logic should be the same as for the addPurchase method, you may just copy the code and carefully make the appropriate changes. Remember that I will whack your grade if you copy your own code without reading it and modifying it as appropriate. Make sure the tests pass before proceeding. *Hint*: there’s no more return type!
2. Now it is time to “marry” the Bean to the JSP. Open the JSP page and replace the import statement with a <jsp:useBean> tag as shown above. This would be a good time to review the description of useBean in your text! The  part tells Tomcat to keep this object around from one request to the next, so that we can accumulate an unpaid balance.[[1]](#footnote-1)
3. Delete all the Java Scriptlets and Expression Scriptlets in your JSP page. These will be replaced with Bean references.
4. Save all the files and rerun the application. If it looks as if Eclipse has not incorporated all your changes, use the Project/Clean command.
5. Uh-oh! The JSP page now has an error. Checking down through the stack trace, you should see that our friend Jasper is upset with us,which means there is a problem translating our JSP. In particular, Bean classes must have a no-arg constructor. So, we must add such a constructor, and then setter methods for the account number (we already have setters for the customer name, address, *etc*. ) We had very good reasons for not writing such a constructor before, but we need it now.[[2]](#footnote-2)
6. Add some code to the testConstructor method in CustomerAccountTest to validate creating a CustomerAccount with the no-arg constructor, then populating it with setter methods. Make sure you see the green bar before proceeding.
7. OK, now add the <jsp:setProperty> tags into the JSP page to populate the Bean:



1. Theoretically, the <jsp:setProperty> tags should automatically cause the Bean setter methods to fire. Since this is getting a bit mysterious, add some debug printouts to verify that these methods are in fact being invoked:



1. Run this puppy and see if the printouts appear on the Console.
2. Now, use <jsp:getProperty> tags to extract the properties from the Bean:



1. Now, add an input form to add a new purchase.

1. Actually the correct scope for this data is application scope, which persists over the entire lifetime of the credit card company. We’ll make this change in a future iteration, when we get to the database part of our course. [↑](#footnote-ref-1)
2. “I contradict myself? Very well then, I contradict myself.” – Walt Whitman [↑](#footnote-ref-2)