This assignment has you writing some AspectJ code to improve your ability to learn and write pointcuts. You should find a lot of help on completing these assignments by going to the library Web pages and looking at the *AspectJ in Action* and *AspectJ Cookbook* books that are online.

You are given the project that we used in class for demonstration. It is in the file ChessBoardObserver.zip. You can import this Eclipse project directly into Eclipse (with the AspectJ plugin) by selection **Import>Existing projects into workspace…** and selecting the zip file.

You will notice that there is a folder called **SampleAspects**. This contains some aspects that you could move to the **aspects** package under your **src** file and try them out.

For this part of the assignment, you will create aspects that exhibit the pointcuts described in the descriptions below. You should test them to make sure that they work. You will put all of your aspect files in the **Solutions** folder when you submit your work.

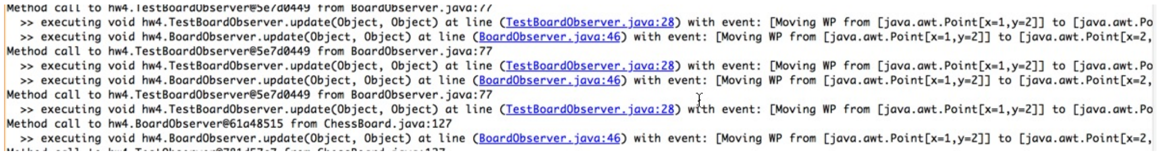
1. Create an aspect named ShowExceptionAspect. This will catch the HW4Exception in the ChessBoard.move() method and print a message like "Exception thrown at execution(void hw4.ChessBoard.move(Point, Point)) with message [Error moving from java.awt.Point[x=4,y=5] to java.awt.Point[x=4,y=8]]". There is a JUnit test that causes this exception to be thrown, so you should see the output from your aspect by running the tests. You might find some help at http://www.hubberspot.com/2012/12/how-to-implement-after-throwing-advice_12.html.
2. Create an aspect named FixArgumentsAspect. This will execute on entry to the ChessBoard.move() method. It will look at the values of the coordinates and if the x or y coordinates of either point are less than 0 or greater than 7, the aspect's advice will set them to 0. This should cause the badCoordinate() test to fail. See *AspectJ Cookbook* section 4.2 for help.
3. This is similar to the previous aspect. Create an aspect called PointValidatorAspect. This will check the x and y coordinates before all calls to the Point constructor. If the values are not in the range 0-7, then the advice will throw an IllegalArgumentException (an instance of a RuntimeException). This will also cause the badCoordinate() test to fail. See *AspectJCookbook* section 7.1 for help.
4. This item will require two pointcuts, one for each message that you will print out. Create an aspect called ObserverReporterAspect. Whenever a call is made to any Observer method (there is only one, update()) it will print out a line like this:

   Method call to hw4.TestObserver@cddb2e7 from BoardObserver.java:77

   When the method is executing, it will print out a line like this:

>> executing void hw4.TestBoardObserver.update(Object, Object) at line (TestBoardObserver.java:28) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Point[x=2,y=1]]]

There is a toString() method on the MoveEvent that prints out a significant part of this line. See *AspectJ Cookbook,* sections 4.3-4.4 for help. A sample portion of the output would look like this:



```
Method call to hw4.TestBoardObserver@5e7d0449 from BoardObserver.java:77
  >> executing void hw4.TestBoardObserver.update(Object, Object) at line (TestBoardObserver.java:28) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Po
  >> executing void hw4.BoardObserver.update(Object, Object) at line (BoardObserver.java:46) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Point[x=2,
Method call to hw4.TestBoardObserver@5e7d0449 from BoardObserver.java:77
  >> executing void hw4.TestBoardObserver.update(Object, Object) at line (TestBoardObserver.java:28) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Po
  >> executing void hw4.BoardObserver.update(Object, Object) at line (BoardObserver.java:46) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Point[x=2,
Method call to hw4.TestBoardObserver@5e7d0449 from BoardObserver.java:77
  >> executing void hw4.TestBoardObserver.update(Object, Object) at line (TestBoardObserver.java:28) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Po
Method call to hw4.BoardObserver@61a48515 from ChessBoard.java:127
  >> executing void hw4.BoardObserver.update(Object, Object) at line (BoardObserver.java:46) with event: [Moving WP from [java.awt.Point[x=1,y=2]] to [java.awt.Point[x=2,
```