(8 pts.) 1. Fill in the following blanks with the appropriate Java keywords:

   a) The following code wants to throw an exception:
   ```java
   void foo() {
      __throw_______ new FeatureUnimplementedException();
   }
   ```

   b) The following code wants to execute some code that might throw an exception, handle a certain exception if it occurs, but in any case, do some cleanup:
   ```java
   public void converseWithStarship() {
      OpenCommunicationToStarshipEnterprise();
      __try______ {
         sendMessageToCaptainPicard();
         Response r = receiveResponseFromStarship();
         if (r.shipIsUnderAttack())
            beamSelfToShip();
      } __catch______ (CommunicationException c) {
         reportCommunicationErrorToEngineer();
      } __finally______ {
         closeCommunicationToStarshipEnterprise();
      }
   }
   ```

(7 pts.) 2. The following are the steps that are used in Test-Driven Development, but in mixed-up order.  Number them from 1 to 7 in the correct order:

   4 Write enough code that the test compiles (test will likely fail).
   3 Compile (compile will likely fail).
   1 Decide the next small step to accomplish.
   6 Refactor.
   7 If you need more testing of the small accomplishment, go back to step 2.
   2 Write a test that'll test that accomplishment is successful.
   5 Write just enough code to make your test pass.

(5 pts.) 3. Give a one-sentence definition of refactoring: