Overall requirements:

* + The user should be able to create and delete shapes.
  + The user should be able to create a set which may contain 0 or more shapes.
  + Each shape may be in 0 or 1 sets. It may be helpful to think of a newly created shape to be in the “no set” set.
  + A “name” of a shape or a set of shapes must be 30 or fewer characters and may only contain letters. They are CASE-SENSITIVE. Each shape must have a unique name, each set must have a unique name, but a shape and a set may have the same name.
  + Legal colors are BLACK, RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET, WHITE. These are case-insensitive.
  + Any number describing a shape (radius, side, length, width) must be a Double, and positive.
* Show commands:
  + show shapes
    - Description: Shows the NAME and DESCRIBE of all shapes.
    - Example output 1:  
      There are no shapes in the system.
    - Example output 2:  
      Listing all shapes in the system:  
      [MyCircle] This is a RED circle of radius 2.5  
      [someshape] this is a BLACK square with side 3.4  
      …
  + show sets
    - Description: Shows the NAME count of items in each set
    - Example output 1:  
      There are no sets in the system.
    - Example output 2:  
      [FirstSet] 5 items.  
      [somesset] 0 items.  
      …
  + show <setName>
    - Description: Shows the NAME and DESCRIBE of all shapes in the set
    - Example output 1:  
      Set <setName> does not exist.
    - Example output 2:  
      Set <setName> is empty.
    - Example output 3:  
      Set <setName> has the following items:   
      [MyCircle] This is a RED circle of radius 2.5  
      [someshape] this is a BLACK square with side 3.4  
      …
* Create commands:
  + create set <setName>
    - Description: Create an empty set with <setName> or error if name is invalid or exists
    - Example output 1:  
      Invalid set name.
    - Example output 2:  
      Set <setName> already exists.
    - Example output 3:  
      Set <setName> successfully created.
  + create rectangle <shapeName> length <l> width <w> color <c>
    - Creates a rectangle with the specific name, length, width, and color
    - Example output 1:  
      Invalid shape name.
    - Example output 2:  
      Invalid shape parameters.
    - Example output 3:  
      Shape <shapeName> already exists.
    - Example output 4:  
      Rectangle <shapeName> successfully created.
  + create circle <shapeName> radius <r> color <c>
    - Creates a circle with the specific name, radius, and color
    - Example output 1:  
      Invalid shape name.
    - Example output 2:  
      Invalid shape parameters.
    - Example output 3:  
      Shape <shapeName> already exists.
    - Example output 4:  
      Circle <shapeName> successfully created.
  + create square <shapeName> side <s> color <c>
    - Creates a square with the specific name, radius, and color
    - Example output 1:  
      Invalid shape name.
    - Example output 2:  
      Invalid shape parameters.
    - Example output 3:  
      Shape <shapeName> already exists.
    - Example output 4:  
      Square <shapeName> successfully created.
* Delete commands:
  + delete set <setName>
    - Deletes the set but does not delete the underlying shapes. Set must exist
    - Example output 1:  
      Set <setName> does not exist.
    - Example output 2:  
      Set <setName> has been deleted.
  + delete shape <shapeName>
    - Deletes the shape and removes it from all sets. Shape name must exist.
    - Example output 1:  
      Shape <shapeName> does not exist.
    - Example output 2:  
      Shape <shapeName> has been deleted.
* Add and Remove commands:
  + add <shapeName> to <setName>
    - Add the shape to the set. Both must exist. Warning message if shape is already in set.
    - Example output 1:  
      Shape <shapeName> does not exist.
    - Example output 2:  
      Set <setName> does not exist.
    - Example output 3:  
      Shape <shapeName> is already in set <setName>
    - Example output 4:  
      Shape <shapeName> added to set <setName>
  + remove <shapeName> from <setName>
    - Remove the shape from the set. Both must exist. Warning message if shape is not in set.
    - Example output 1:  
      Shape <shapeName> does not exist.
    - Example output 2:  
      Set <setName> does not exist.
    - Example output 3:  
      Shape <shapeName> is not in set <setName>
    - Example output 4:  
      Shapes <shapeName> removed from set <setName>
* Compare commands
  + compare shape [perimeter|area] <shapeName1> <shapeName2>
    - Compares the perimeter or area of two shapes. Both must exist.
    - Example output 1:  
      Shape [<shape1Name>|<shape2Name>] does not exist.
    - Example output 2:  
      The [perimeter|area] of <biggerShape> is <amt> larger than <smallerShape>.
  + compare set [perimeter|area] <setName1> <setName2>
    - Compares the sum of the perimeters or areas of the shapes in two sets. Both set must exist.
    - Example output 1:  
      Shape [<setName>|<set2Name>] does not exist.
    - Example output 2:  
      The [perimeter|area] of <biggerSet> is <amt> larger than <smallerSet>.
* \*\*\* Extra credit commands
  + clear set <setName>
    - Removes all shapes from a set, leaving an empty set. Set must exist.
    - TWO EXTRA CREDIT POINTS
    - Example output 1:  
      Set <setName> does not exist.
    - Example output 2:  
      Set <setName> is now empty.
  + union <setName1> with <setName2>
    - Add all shapes from setName2 to setName1 if they are not already there.
    - THREE EXTRA CREDIT POINTS
    - Example output 1:  
      Shape [<set1Name>|<set2Name>] does not exist.
    - Example output 1:  
      <count> items from <set2Name> added to <set1Name>.

Discussion of Architecture:

* You will need a database schema. This will be provided to you.
* You will need a mechanism for storing Shapes and Sets of shapes in memory.
* You will need to create a strategy for saving items to, and loading items from, the database. We will discuss this on March 3.
* You will need a command parsing infrastructure. This should skew towards simplicity, the UI doesn’t have to be perfect (e.g. “syntax error” if the command isn’t perfect – more details = more complete project).