# Spec Breakdown

1. Rewrite Assignment One in Java
   1. Circular doubly linked list with sentinel
   2. Classes
      1. **MyPolygons**
      2. **Node**
      3. **Polygon**
      4. **Point**
   3. Do NOT return nodes at any point
   4. Extend **MyPolygons** to include sorting functionality in a class called **SortedPolygons**
   5. Create an interface called **ComparePolygons**
      1. This contains an abstract method **compareTo*()***
      2. Implement the interface on **Polygon**.
2. Write an *abstract* **PlanarShape** class the purpose of which is to inherited by any more specific shapes, e.g. triangles, square and so on. These are all instances of a **PlanarShape.**
   1. It will have abstract methods:
      1. toString()
      2. area()
      3. originDistance()
   2. **PlanarShapes** must be comparable therefore we must use the standard Comparable<T> interface, using the specification **implements Comparable<PlanarShape>.**
3. **Polygon** is now a descendent of **PlanarShapes** and because **PlanarShapes** is an abstract class the methods for **Polygon** will have to be implemented.
4. The linked list:
   1. Classes will have to be redesigned to use generics and iterators.
   2. The circular doubly-linked linked list will be called **LinkedList.**
   3. Contain methods:
      1. prepend()
      2. append()
      3. takeHead()
   4. There will be no *current* item as this functionality will be carried out by an iterator.
   5. **LinkedList** must be iterable so must provide an iterator.
5. Second list will be called **SortedList**.
   1. This will be an extension of the **LinkedList**
   2. It will need a method insertInOrder()
      1. Using insertion sort algorithm
   3. Must use the **comparable** interface implementation of the **compareTo<PlanarShape>** method.
   4. **SortedList** will be instantiated to hold only **PlanarShapes.**
6. Output:
   1. Format is the same as for assignment one EXCEPT it should be produced using iterators to visit the objects in the list.
7. Investigate Factory Method to produce polygon objects and return them using a **PlanarShape** reference.