Project clarifications

* Groups of 2-3, you do not need to use the same groupings as in the past
  + This project is not recommended for 1 person
* Allow the @Pointcut to specify a parameter list and return type and search based on these in addition to the name
  + NOTE: up to you how you wish to implement this
  + No need to handle wildcards, only specific parameter lists and return types
  + No need to handle inheritance related issues for the parameters (i.e. compatible type), no need to handle primitive vs wrapper class, no need to handle covariant return types
    - If the params and return do not exactly match what you specify it doesn’t match it
* Allow for processing of an @Around advice in addition to the required @Before and @After
  + Add an Object processAround(Method method, Object[] args) to AspectManager.java
  + NOTE: one of the design patterns that will be presented will give a potential answer
  + Around advices are handled after the @Before and prior to @After
  + Around Advices are processed as follows

Around1

Around2

Original method

Around2

Around1

* Allow for the handling of stateful Aspects, i.e. Aspects that remember values from previous invocation of a SPECFIC proxy instance. Example, an Aspect that remembers how many times a method call was called on a specific object.
  + HINT: to remember a specific object instance you can use the hashcode() of the object as a key to a map
    - You will need to modify the existing code so you can pass the current instance for later use