CS56—03/02/2016 Code Review Exercise

Instructions: PART ONE (individual)

In lab03, you implemented:

```
  public class Polynomial extends ArrayList<Integer>
```

One of the methods was the toString() method.

Today, you will each be given two excerpts of code from this lab—two implementations of the toString() method by students from previous offerings of CS56.

These students gave their permission for me to share these (anonymously) for this exercise. So the excerpts are identified by letter only: A, B, C, etc.

You need two different lettered copies, e.g. (A,B), (C,F), (D,K), etc.
It does not matter which ones, as long as they are different—they don't have to be consecutive.

You job is to review each of these.

Your job is not to tear apart the code, but to invite the author into a conversation.

- Please try to find at least two instances of good practice the author used (+)
- Please try to find at least two places where the code could be improved. (?)
- Then find as many additional items as you can, of either type.

Please write each item on the worksheet provided, in this format:

+ or ?, then line number, line numbers, or range, then a comment.

For example:

+ 105,108,130	Clear use of comments to indicate purpose of code that follows
? 134	Would "power" be a better variable name than "x"?

etc.

Further instructions:

- One review goes on one side of the worksheet, and the other on the reverse.
- When commenting on places where the code could be improved, you may like to phrase your remarks as a question that invites discussion, rather than as a direct critique.
- When offering comments, please be as specific as you can about how the effect of the good practice you are observing or suggesting.
- Find as many issues as you can.
    - o You should do your best to find at least four, but don't stop there
    - o List as many issues as you can find, of both types (+ and ?)

Instructions—Part Two (group)

Now get into groups of four, based on ONE of the two codes you reviewed.

For example:

- one group should be four people that reviewed A
- another should be four people that reviewed B
- etc.

You are now invited into a conversation with one another, where you produce a combined review.

You will take on four roles:

- **Moderator**—chairs the group and keeps meeting on track
- **Reader**—Reads out lines of code to be reviewed, one at a time. (skip initial javadoc)
- **Recorder**—Records what the group decides to mark down as issues
- **Author**—Advocates on behalf of the code

In fact, none of you is the author of the code—but one of you gets to pretend that she/he is. This will help us practice being "diplomatic" in our sharing. The "pretend author" should advocate for the code "as if" he/she authored it.

What to do:

```
foreach (line of code that isn't initial javadoc) do {
      Reader reads the line of code
      Moderator asks group what issues they have
      All group members read their issues for this line  (moderator last)
      Moderator facilitates a discussion
      Recorder writes down the issues that reflect the group consensus
            (same syntax as for part 1)
      if (this is last line of code)
            do all steps above  but from perspective of method "as a
whole"
}
```

Moderator's roles:

- Keep the discussion moving forward
- Encourage all members to participate
- Moderator may contribute their own ideas, but it is best to do so "last"
- Moderator should encourage forward momentum—try to avoid a situation where group gets "bogged down" on a particular point.
- If group is having trouble reaching consensus on a particular line of code, consider summarizing both (or all three? four?) points of view, and moving on.