Overview: You recently completed (we hope!) your own implementation of the Polynomial methods, using an array for the storage of a polynomial. One problem with arrays is that they don’t natively support a lot of operations, which means that you had to do a lot of the dirty work yourself (e.g. shifting of data, if you kept the terms of the polynomials in order by exponent).

An alternative to doing the implementation from scratch is to adapt another class. We learned about adaptation not too long ago. As a Polynomial object is really just a list of Term objects, we can employ one of Java’s supplied list classes to represent a Polynomial’s collection of Term objects. This won’t be a perfect example of adaptation, because we need operations such as scalarMultiply() that have no corresponding operation in a general-purpose list class. Still, it should be much easier than what you had to do in Program #7, once you become comfortable with the operations of the class that you choose to adapt.

Java has multiple general–purpose list classes. Two of them, ArrayList and Vector, are very similar. A third, LinkedList, stores lists of data using a different internal representation that we’ll talk about in detail soon. All three implement Java’s List interface.

Assignment: Start by making two copies of your Prog7.java file; name them Prog8A.java and Prog8B.java (and adjust the class names within, too!). Also bring along Quantity.java and Term.java, which, if you did a good job implementing them for the last assignment, shouldn’t need much (if any) updating to be reused here. You may update any or all of these files to add more tests, correct logic errors, etc.

Next, create two new Polynomial classes and files (creatively named PolynomialA.java and PolynomialB.java). As the names suggest, Prog8A.java is to use PolynomialA and Prog8B.java is to use PolynomialB. Both sets of files are to use the same Quantity.java and Term.java files.

PolynomialA.java will implement Program #7’s Quantity interface plus its Polynomial methods by adapting either Java’s ArrayList class or Java’s Vector class (your choice). Thus, in PolynomialA you will be replacing the array object with either an ArrayList object or a Vector object, and rewriting the implementations of the methods that rely on that representation.

PolynomialB.java will do a similar adaptation, using Java’s LinkedList class instead (no choice here). We haven’t talked about linked lists yet, but our coverage of list operations should enable you to use the LinkedList class for this task easily.

Data: As with Program #7, for this program there will be no sample data. After you submit your programs, we will run our versions of the Prog8A and Prog8B classes on your new PolynomialA and PolynomialB classes. The non–documentation portion of your grade on this assignment will be determined by how well your code passes our testing. As usual, make a point of doing a really good job testing your classes (which, if you did a good job of it on the last assignment, should be easy because you can reuse (and augment!) them for this assignment. Reviewing your tests from Program #7 should help you think of new tests to try.

Output: Because the output is dependent upon the construction of the Prog8A and Prog8B classes, there is no specific output expected. The correct behavior of your Polynomial methods again will be the major factor in your assignment grade.

(Continued ...)
Turn In: Use the ‘turnin’ utility to electronically submit your Term.java, Quantity.java, PolynomialA.java, Prog8A.java, PolynomialB.java and Prog8B.java files to the cs227p08 directory at any time before the stated due date and time. Of course, you can turn them in late if you still have late days to use, or don’t mind losing 20% per day if your late days are exhausted.

Hints, Reminders, and Other Requirements:

- If you didn’t complete the Term.java and Quantity.java files in Program #7, you’ll need to get them functional, as they are reused here. If you had completed them, you should be able to reuse them here w/o modification . . . if you implemented them correctly!

- Be sure to note, in your external documentation for PolynomialA.java, from which class you chose to perform your adaptation: ArrayList or Vector.

  If you happen to already know how to use ArrayList, this is a good opportunity to try Vector, and vice-versa. If both are new to you, we recommend adapting ArrayList.

- Don’t forget to revise your documentation as necessary for this assignment. If you did a good job documenting your Program #7 code, you shouldn’t have to make many changes to document this one. (But be sure to make those changes – your SL will be looking at your documentation closely.)

- You’re still welcome to share testing code (that is, your Prog8A/B classes) with your classmates, as you were on Program #7.

- If you find that you have to extensively modify the files that this assignment shares with the last assignment (that is, Term.java, Quantity.java, and just about all of Prog7.java), take some time to reflect on your Program #7 design decisions. What could you have done differently that would have allowed you to avoid changing those files for reuse in this assignment? Remember, one of the advantages of object-oriented languages is their support for code reuse. Whenever possible, you should try to design your code to be easily reusable.

- If you’re an avid reader of the Java API, you may have noticed that there are methods (such as toArray() covered by the List interface) that turn lists of elements into arrays of elements. Using such methods in combination with your existing array-based Polynomial class from Program #7 would make this assignment really easy . . . which is why we’re forbidding their use in this assignment. This shouldn’t be a shock, given that we’ve just talked about lists in class and the point of this assignment is to give you some practice with them. Please follow the ‘spirit’ of the exercise.

---

1 And why wouldn’t you be? There’s action, suspense, and romance in every class description! ©