# Problem 7: Getting a Real Taste of Java -- Balls and Boxes

Until now, we have only been introducing tools. In this problem, we will delve into a real programming exercise. If you are not familiar with Java, we recommend walking through Sun's Java Tutorial.

The intention of this problem is to give you a sense of what Java programming entails and to demonstrate the use of Eclipse and the JUnit testing tool. If you have never programmed in Java. you may find this problem somewhat challenging. Do not be discouraged, if you spend effort getting this problem right now, you will likely have less trouble later in the course. This is why we encourage you to work in pairs. The intention is that students who have Java experience can help those who have little or no experience to get up to speed. Nevertheless, if you get stuck, please do not hesitate to seek help from one of the TAs or LAs.

As you work on this problem, you should also record your answers to the various question in a file called problem7.txt in the project's doc/ directory.

**a. Warm-Up: Creating a Ball**

As a warm up exercise, take a look at Ball.java. A Ball is a simple object that has a capacity.

- What is wrong with Ball.java? Please fix the problems with Ball.java and document your work in problem7.txt.

If you pay attention to the warnings in Eclipse, you should be able to find at least one of the bugs without referring to the JUnit results.

**b. Using Pre-Defined Data Structures**

Next, we want to create an class called BallContainer. As before, skeleton code is provided (see BallContainer.java). A BallContainer is a container for Balls. BallContainer must support the following methods and your task is to fill in the code that will implement all these methods correctly:

1. add(Ball)
2. remove(Ball)
3. getCapacity()
4. size()
5. clear()
6. contains(Ball)

One of the nice things about Java is that it has many libraries and pre-defined data structures that you can simply use without have to write your own from scratch. One of the intentions of this problem is to expose you to the use of pre-defined Java data-structures. In BallContainer, we use a java.util.LinkedList to keep track of the balls. If

you open BallContainer.java in Eclipse, you will notice that there is a warning message associated with the line:

contents = new LinkedList();
It should say something about an "Unsafe type operation". If you are not familiar with Java 1.5, you may be puzzled by this warning. Since Java 1.5, the Java Collections
If you pay attention to the warnings in Eclipse, you should be able to find at least one of the bugs without referring to the JUnit results.
framework is strongly typed so instead of defining a generic LinkedList, you should define a typed LinkedList. Please modify this line to remove this warning for the constructor statement.

Before you proceed to implement the required method, please take a moment to read through the documentation for LinkedList. Some of the methods that you are required to implement simply require you to call the appropriate predefined methods for LinkedList.

Hint: Place your cursor on LinkedList, and press SHIFT-F2. A web browser window will appear, showing documentation for the LinkedList class.

Most of the methods that you are required to implement are quite simple. Before you start coding, please take time to think about the following questions (you need to turn in your answers):

- There are two obvious approaches for implementing getCapacity():

    1. Every time getCapacity() is called, go through all the Balls in the LinkedList and add up the capacities. (Hint: If you choose this approach, you will probably want to use an Iterator to extract Balls from the LinkedList. You can see an example of how Iterator is used in BoxTest.java.)

    2. Keep track of the total capacity of the Balls in BallContainer whenever Balls are added and removed. This obviates the need to perform any computations.

- Which approach do you think is the better one? Why?

**c. Implementing Algorithms**

By the time you are done with BallContainer, you should be somewhat comfortable with Java, so in this problem, we want you to do a little more design and thinking and a little less coding. Your final task in this problem is to create a class called Box. A Box is also a container for Balls. The key difference between a Box and a BallContainer is that a Box has only finite capacity. Once a box is full, we cannot put in more Balls. The size (capacity) of a Box is defined when the constructor is called:

public Box(double capacity);

Since a Box is really a BallContainer with some extra properties, it makes sense to say that in fact a Box is a type of BallContainer, which is why the Box is defined to extend BallContainer. This is called Inheritance. Don't worry too much about this; you will learn more about this later in the course. For now, it suffices to know that what this means is

that Box automatically has all the methods and properties of BallContainer. When you look in Box.java, you will realize that Box has only two methods defined in Box.java:

1. add(Ball)
2. getBallsFromSmallest()

Depending on your implementation of getCapacity() in Problem 7b, you may also needto implement a different remove(Ball) method for Box as well. If so, please do so. There is no need to change your implementation of BallContainer for this problem. You can also make other changes to Ball, BallContainer or Box, but you should explicitly describe your changes in problem7.txt and explain why the changes were made.

As before, your task is to implement these two methods. Before you start working on getBallsFromSmallest(), you may wish to check out the documentation on Iterator (Place your cursor on Iterator, and press SHIFT-F2.) Also, take some time to answer the following questions (which you need to turn in):

1. There are many ways to implement getBallsFromSmallest(). Brainstorm with your buddy and come up with at least two ways. Briefly describe them.
2. Which of the above ways do you think is the best? Why?

There is no single correct answer. The whole point of this exercise is help you fight that urge to code up the first thing that comes to mind, but instead spend a little more time thinking before you start coding. Remember: More thinking, less coding.