Lab 5 – Recursion and Inheritance

due September 20th at 23:59

Objective

Today's lab will help you get familiar with

- Recursion
- Inheritance
- Abstract classes

Abstract Classes Review:

Abstract (which Java supports with abstract keyword) means that the class or method or field or whatever cannot be instantiated (that is, created) where it is defined. Some other object must instantiate the item in question. If you make a class abstract, you can't instantiate an object from it. Therefore, abstract classes do not have a constructor..

Assignment:

1. Create a class Cell. The class has the following:
    a. val: a private variable - Integer
    b. next: a private variable - Cell
    c. append(Integer x): checks to see if next is null. If it is, create a new Cell and assign it to next, and set the value of val to x. Otherwise call append(x) on next.
    d. toString(): returns a String constructed by pre-pending val to the String returned by applying toString() to next (if next is not null)
2. Create an abstract class IntegerListADT. The class will have the following methods (no constructors) declared with an empty body:

    a. append(Integer x): inserts x at the end of the list
    b. toString(): returns a String that is stored in this list
    c. isEmpty(): returns true if the list is empty.
3. Create a class IntegerList that inherits from IntegerListADT. The class has the following:
    a. root: a private variable - Cell
    b. append(Integer x): checks to see if root is null. If it is, create a new Cell and assign it to root. Call append(x) on root.
    c. toString(): checks to see if root is null. If it is, return the empty String. Otherwise, return the String from calling toString() on root. d. isEmpty(): returns true if root is null. Otherwise return false.
4. Create an ExperimentController class. The class will have the following methods:
    a. timeAppend(int numberOfItems, int seed): • Create an instance of IntegerList
        i. For the specified numberOfItems, insert random integers between 0 and 200 by using Random and the append() method.
        ii. The method will return the time taken to add all the items to the container using append().
    b. timeToString(int numberOfItems, int seed): • Create an instance of IntegerList.
        i. For the specified numberOfItems, insert random integers between 0 and 200 by using Random and the append() method.
        ii. The method will return the time taken to call toString() after all the items have been inserted.
5. Unit test the Cell and IntegerList classes.
6. Run your program through ExperimentController so that you test your program for various sizes of input. For each amount of data you should run multiple trials with different seeds. You then can create graphs where the y axis signifies the amount of time, and the x axis is the number of elements. More specifically:

a. Compare the average run time of append() for different amounts of data.
b. Compare the average run time of toString() for different amount of data.

Submission:

In addition to your code you must submit lab notes(in PDF format). Save your lab notes in the project folder before you compress it and upload it. Details about the lab notes can be found on the CS150 lab Moodle page.

Grading:

1. Unit testing (for Cell and IntegerList classes) – 1 pt
2. Cell class – 2 pts
3. IntegerList class – 2 pts
4. IntegerListADT class – 1 pt
5. ExperimentController Class – 1 pt
6. Style/commenting – 1 pt
7. Report – 2 pts