Lab 3 - Recursion and Inheritance

due September 15th 2019 at 23:55

Objective

Today's lab will help you get familiar with

- Recursion
- Inheritance
- Abstract classes

Abstract Classes:

Abstract classes. 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:

For this lab:

1. Create a class **Cell.** The class has the following:
   a. *val:* a private variable - **String**
   b. *next:* a private variable - **Cell**

   c. *append(String x):* checks to see if *next* is null. If it is, create a new Cell and assign it to *next*. 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 **StringListADT**. The class will have similar functionality to the List ADT that you covered in lecture and is in the book (Section 2.1). The class will have the following methods (no constructors) declared with an empty body:
   a. *append(String 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 **StringList** that inherits from **StringListADT.** The class has the following:
   a. *root:* a private variable - **Cell**
   b. *append(String x):* checks to see if *root* is null. If it is, create a new **Cell** and assign it to *root.* If it is not-null, 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. (From Lab 2) Create a private static String array (twoLetterWords) initialized to contain the following words, one per element: aa ab ad ae ag ah ai al am an ar as at aw ax ay ba be bi bo by da de do ed ef eh el em en er es et ew ex fa fe gi go ha he hi hm ho id if in is it jo ka ki la li lo ma me mi mm mo mu my na ne no nu od oe of oh oi ok om on op or os ow ox oy pa pe pi po qi re sh si so ta te ti to uh um un up us ut we wo xi xu ya ye yo za.

   The class will have the following methods:
   a. timeAppend(int numberOfItems, int seed):

        i. create an instance of a StringList
        ii. for the specified numberOfItems, insert random words from twoLetterWords to the container using the addAppend() method.
        iii. The method will return the time taken to add all the items to the container using addAppend().
    b. timeToString(int numberOfItems, int seed):
        i. create an instance of a StringList
        ii. for the specified numberOfItems, insert random words from twoLetterWords to the container using the append() method.
        iii. The method will return the time taken to call *toString()* after all the items have been inserted.
5. Unit test the Cell and StringList 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 a report (in PDF format). Save your report in the project folder before you compress it and upload it. Details about the report can be found here.

Grading:

1. unit testing (for Cell and StringList classes) - 1 pt
2. Cell class - 2 pt
3. StringList class - 2 pts
4. StringListADT class
5. ExperimentController Class - 1 pt
6. Style/commenting - 1 pt
7. Report - 2 pts