Assignment 1

Due date

- Git url: https://classroom.github.com/a/TFTQcOcl

Due date

- git setup is due by June 8th.
- Due by 11.59 PM EST on June 14th.

Submit your code as per the provided instructions.

Updates

Assignment Goal

A simple Java program and corresponding test code, along with use of Git.

Team Work

- No team work is allowed. Work individually. You cannot discuss the assignment with ANYONE other than the instructor and TA.

Programming Language

You are required to program using Java.

Compilation Method

- Compilation: Your code should compile on bingsuns or remote.cs.binghamton.edu with the following command:
- `ant -buildfile src/build.xml all`

  or

  ```
  You should be able to demo your code via Eclipse (screen sharing via
  google hangout or Skype).
  ```

Policy on sharing of code

- EVERY line of code that you submit in this assignment should be written by you. Do NOT show your code to any other student. Do not copy any code from any online source. Code for File I/O or String operations, if found online, should be clearly cited, and you cannot use more than 5 lines of such online code.
- Code downloaded in its entirety from another person's online repository of code (GitHub, BitBucket, etc.) and submitted as student's own work, even if citied, is considered plagiarism.
- Code snippets, for File I/O, if used from an online source should be cited by mentioning it in the README.txt and also in the documentation of every source file in which that code appears.
- Post to the listserv if you have any questions about the requirements. Do NOT post your code to the listserv asking for help with debugging.

Project Description

Assignment Goal: Develop a program, using Java, to design a new data structure and develop test code for it.

- Design a data structure named "MyArrayList" with the following properties:
  - The data structure should have an integer array as a private data member. Start with some initial int array size, say 50. If the input size is greater than 50, then create a new array of size 75, copy the contents of the original array into this new array, and then continue).
  - void insertSorted(int newValue); This method should insert a new value but keep the data structure sorted in ascending order. If a value is a duplicate, you can store it before or after an existing value.
  - void removeValue(int value); This method should remove all occurences of a value, if it exists, and then move the remaining values so that the array list has all values in ascending order.
  - int indexOf(int value); This method should return the index of the first occurrence a value. It should return -1 if the value does not exist in the array list.
  - int size(); This method should return the total number of values that are stored in the array list.
  - int sum(); This method should return the sume of all values that are stored in the array list.
  - For all the above methods, provide your own implementation.
  - An empty constructor, which sets the privata data members to default values.
  - A toString(...) method that prints all the values of the array in a pretty manner.
  - In addition, you can use private helper functions to consolidate operations such as array resizing and sorted insertion/removal.
- In the file input file, you will be provided input as shown below. All input numbers will be in the range 0-10000. Insert all of these numbers into your MyArrayList.
- `12`
- `1`
- `0`
- `1234`
- `9999`
- `123`

- Use Java, to write a program to make the assignments, so that the output file looks like the following:
-
- `The sum of all the values in the array list is: 12345`
- Replace 12345 in the output file with the actual sum.
- The MyArrayListTest.java file should have a method named "testMe(MyArrayList myArrayList, Results results)" that should call at least 10 different testMethods (all in the MyArrayListTest.java) each of which should call a single or a combination of methods on the myArrayList instance and test that it has been implemented correctly (including tests for boundary cases). At the end of each internal test method, it should write to results.storeNewResult(...), the name of the test, whether it passed the test or failed. If it failed, the error should be printed. If it passed, it should print that "test XYZ passsed", where XYZ is replaced with a meaningful name for the test.
- Class participation points will be given to the first 10 students who post interesting MyArrayListTest.java files to me (which I will then send to the entire class).
- The Driver.java should have the main(...) function. It should create the MyArrayList instance and a Results instance, and call the testMe method on a MyArrayListTest instance. Next, it should print the String stored in the Results instance.
- It is ok to use an in-built sorting algorithm, but you write your own code for copying the array during resizing.

Sample Input Files sent by students in this course

Please note that I have not verfied these input files.

Compiling and Running Java code

- Your README.txt file should have the following information:
  - instructions on how to compile the code
  - instructions on how to run the code
  - justification for the choice of data structures (in terms of time and/or space complexity).
  - Academic Honesty statement

Code Template

  - firstName_lastName_assign_1.tar.gz.
  - git.
- You should have the following directory structure (replace john_doe with your name).
- `john_doe_assign_1/`
- `john_doe_assign_1/README.txt`
- `john_doe_assign_1/myArrayList`
- `john_doe_assign_1/myArrayList/src`
- `john_doe_assign_1/myArrayList/src/build.xml`
- `john_doe_assign_1/myArrayList/src/BUILD`
- `john_doe_assign_1/myArrayList/src/BUILD/classes`
- `john_doe_assign_1/myArrayList/src/myArrayList`

- john_doe_assign_1/myArrayList/src/myArrayList/driver
- john_doe_assign_1/myArrayList/src/myArrayList/driver/Driver.java
- john_doe_assign_1/myArrayList/src/myArrayList/store
- john_doe_assign_1/myArrayList/src/myArrayList/util
- john_doe_assign_1/myArrayList/src/myArrayList/util/FileProcessor.java
- john_doe_assign_1/myArrayList/src/myArrayList/util/Logger.java
- john_doe_assign_1/myArrayList/src/myArrayList/util/FileDisplayInterface.java
- john_doe_assign_1/myArrayList/src/myArrayList/util/Results.java
- john_doe_assign_1/myArrayList/src/myArrayList/util/StdoutDisplayInterface.java
- john_doe_assign_1/myArrayList/src/myArrayList/test
- john_doe_assign_1/myArrayList/src/myArrayList/test/MyArrayListTest.java
- john_doe_assign_1/myArrayList/src/myArrayList/MyArrayList.java
- Some details on each class
    -
    -
    - FileProcessor.java: this class should have a method String readLine(...), which returns one line at a time from a file.
    -
    - Logger.java: You can ignore it for assignment-1. We will start using it in subsequent assignments.
    -
    - FileDisplayInterface.java: This interface should have a method void writeToFile(String s);
    -
    - Results.java: This class should have a data structure as private data member that should store Strings.
    -
    - StdoutDisplayInterface.java: This interface should have a method void writeToStdout(String s);
    -
    - Results ~~FileProcessor~~ should implement FileDisplayInterface and StdoutDisplayInterface
    -
-
-

Code Organization

- Your directory structure should be EXACTLY as given in the code template.

Sample Input Files from Students

. Please note that I have not verfied these input files. If you find any error, or have any comments, please send them to the instructor via email.

Submission

- Read this file for general guidelines on how to prepare a README for your submission.

- Make sure all class files, object files (.o files), executables, and backup files are deleted before creating a zip or tarball. To create a tarball, you need to "tar" and then "gzip" your top level directory. Create a tarball of the directory firstName_lastName_assign_1. We should be able to compile and execute your code using the commands listed above.
- Instructions to create a tarball
- you can run the command "ant clean; ant tarzip" to let ANT create the tarball in BUILD/dist/ folder.
- Alternatively, you can do the following:
    - Make sure you are one level above the directory firstName_LastName_assign_1.
    - tar -cvf firstName_lastName_assign_1.tar firstName_lastName_assign_1/
    - gzip firstName_lastName_assign_1.tar
- Upload your assignment to myCourses, assignment-1.

General Requirements

- Start early and avoid panic during the last couple of days.
- Separate out code appropriately into methods, one for each purpose.
- You should document your code. The comments should not exceed 72 coloums in width. Use javadoc style comments if you are coding in Java. Include javadoc style documentation. It is acceptable for this assignment to just have the return type described for each method's documentation.
- Do not use "import XYZ.*" in your code. Instead, import each required type individually.
- All objects, in Java, that may be needed for debugging purposes should have the "toString()" method defined. By default, just place a toString() in every class.
- Every class that has data members, should have corresponding accessors and mutators (unless the data member(s) is/are for use just within the method.).

Design Requirements

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed. **There is NO difference in penalty for assignments that are submitted 1 second late or 23 hours late** .

Grading Guidelines

Grading guidelines have been posted here.

```
Following are the commands and the instructions to run ANT on your
project.

Make sure you place your build.xml inside src folder along with
myArraylist folder.

-----------------------------------------------------------------------
## To clean:
```

```
ant -buildfile src/build.xml clean

Description: it cleans up all the .class files that were generated when you
compiled your code.

-----------------------------------------------------------------------
## To compile:

ant -buildfile src/build.xml all

Description: Compiles your code and generates .class files inside the BUILD
folder.

-----------------------------------------------------------------------

## To run by specifying arguments from command line
## We will use this to run your code

ant -buildfile src/build.xml run -Darg0=input.txt -Darg1=output.txt

Description: So, you code must accept input.txt as the first argument to
fetch the input and output.txt as the second to write the results. You
have to place input.txt in the following path :
john_doe_assign_1/myArraylist. You don't have to place output.txt to
run the command. Your code should be able to generate output file if
there isn't one existing already. If there is one already, it should
overwrite the file everytime you run the code.

-----------------------------------------------------------------------

###IMPORTANT GUIDELINES###

1. Make sure you clean your project before you submit it. There
shouldn't be any executables or .class files.

2. You are NOT supposed to include input.txt and output.txt files
along with your submission. We would use a different input file to run
and test your code. Your code must run for any input file.

3. Any violation in the specifications given would lead to you losing
points. Make sure you follow the directory structure given in the
requirements page.

Please drop me an email if you aren't able to proceed with the ANT
commands or are unclear with anything regarding the assignment. Also,
make sure you keep updating the code to bitbucket repository on a
timely basis.
```

mgovinda at binghamton dot edu

Back to Programming Design Patterns