CMPS 101 Spring 2018
Program Assignment #4
Vector ADT

May 21, 2018

Goal

The goal of this assignment is to implement the Vector abstract data type in Java, and to test it.

Getting Started

You are given `Vector.java`, a skeleton for the Vector ADT. It includes the signatures (names, argument types, and return types) for several functions. Your job for this assignment is to complete these functions so they work as specified.

Correctness Points

- 10 points - compiles without errors and is a serious attempt at a solution
- 30 points - Implements all the methods in the starter file.
- 10 points - Implements significant additional functionality.

Requirements

Implement `Vector` abstract data type Complete the functions so they work as specified. DO NOT change the signatures for these functions. In addition to the required functions, you will most likely find it useful to include additional fields and methods in your class. Feel free to add as much as you need. The skeleton is only included to provide you with a starting point, and should not limit you in adding additional functionality.

Testing Write test programs to verify that your ADT implementation is working correctly. You should have a test for all of the methods that you implement.

Direction and hints

- You can find the file Vector.java at /afs/cats.ucsc.edu/courses/cmps101-db/Vector.java
- Use good coding style. See `www.soe.ucsc.edu/~sbrandt/105/coding.html` for guidelines.

Submission instructions

- Create directory named `CMPS101S18PA<PROGRAMMING ASSIGNMENT NUMBER>`, e.g. `CMPS101S18PA4` for Programing Assignment 4.
- In the program assignment directory put in the following files
  - `README` - a short file which lists all the files in the directory and describes what they are.
  - `NoteToGrader` - a short note in which you describe your approach.
  - The `<SOURCEFILES>` - the Java files which you wrote in this assignment.
  - The `<TESTFILES>` - the test files which you used to ensure that your program works correctly.
- Go to your directory, and invoke the `script` command. For a tutorial on how to use this command see `http://www-users.cs.umn.edu/~gini/1901-07s/files/script.html`. In your case, you should invoke the following commands:
  - `script pa4submissionfile.txt`
  - `pwd` - this will show us which directory you are in.
  - `ls -l` - this will list the contents of the directory.
  - `cat README`
  - `cat NoteToGrader`
  - `cat <SOURCEFILES>` - this will print the contents of the source files to the screen. We want you to run this command because this is the only way in which we will see the code you wrote. Run this command on every source files you are using, but DO NOT RUN IT ON YOUR BINARY FILES!
  - `<commands to compile/link the program>` - run the commands which build your source code

  - `ls -l` - this will list the contents of the directory again, showing us that there are binaries which resulted from the previous step.
  - `cat <TESTFILES>` - this will print the contents of the test files to the screen. As before, we want you to run this command because this is the only way in which what kind of tests you ran to ensure that the source files are doing what they should be doing. Run this command on every test file you are using, but DO NOT RUN IT ON YOUR BINARY FILES!
  - `<commands to execute required tests>` - run the commands which test your binaries against specific files. Make sure that the results of these tests are printed to screen.
  - `exit` - this will exit the `script` command and produce a plain text called `pa3submissionfile.txt`.

- Take the plain text file which you created in the previous step and paste its contents to a `.pdf` file. If you've never created a `.pdf` file we have a guideline for this on canvas. This `.pdf` file is the only document you will submit on canvas.
- Do not submit source files or binaries!
- Do not run any editing commands between the `script` command and the `exit` command. It will produce a mess in the plain text file.