JUnit Testing `PythonList`

Recall from the last lab, a list in Python is a fundamental, random access data structure. One main difference between Python lists and arrays / list-based structures in Java is *indexing*. For example, the index of the 'last' element in a Python list is indexed with `-1` and decreasing as we move toward the 'first' element (classic index 0). A few more examples follow.

For list `L = [8, -3, 4, 7, 100, -11]`, `L[-1] == -11` and `L[-6] == 8`.

Note that indices are not unrestricted with Python lists. In particular, given a list L, acceptable indices are in the interval

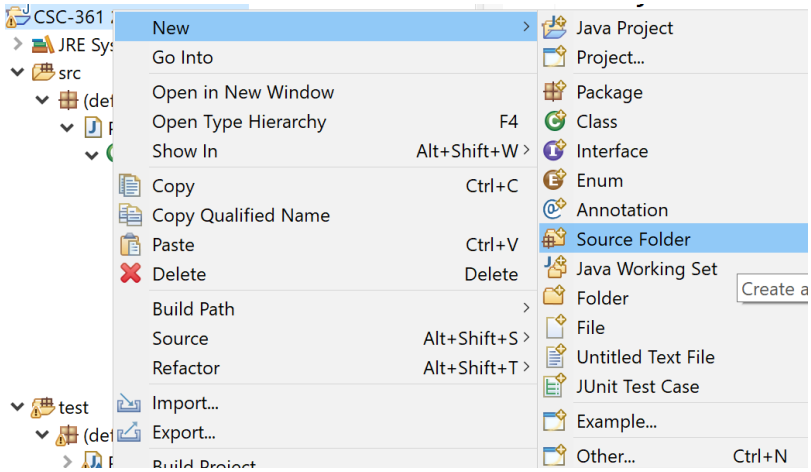
$$-L.size() \leq index < -L.size()$$

and that there are no valid indices for empty lists.

This second part of the lab asks you to thoroughly unit test your `PythonList` implementation by constructing `junit` tests.

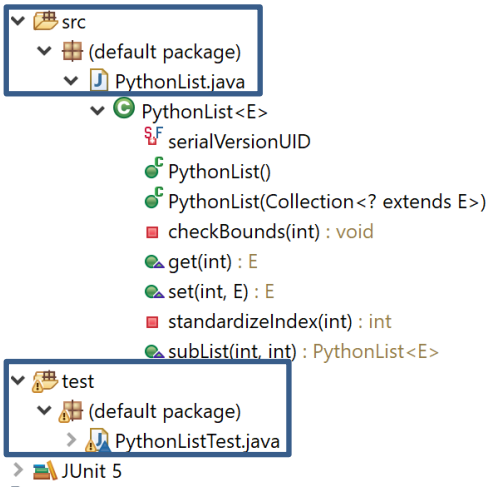
`junit` Testing `PythonList`

You are to implement a set of unit tests for each method of `PythonList`. Place these tests in a single class, `PythonListTest` that will be located in a test folder in Eclipse; the `test` folder will be at the same level in the directory as `src`. See the figure below for how to add a folder and where the folder should be placed.

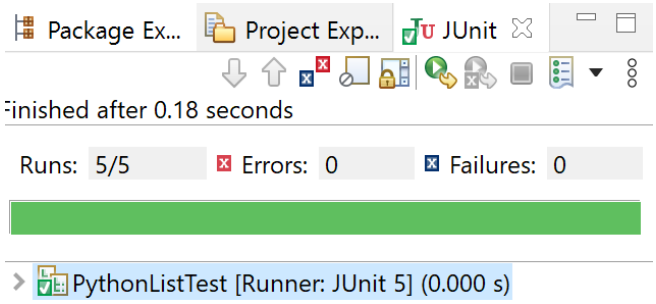


No output should be produced by your tests; we are seeking only a 'green' output indication in Eclipse.

You are to implement a set of unit tests in a single class: `PythonListTest`. See the course slides on how to create this file in Eclipse (`New > JUnitTestCase > New Junit Jupiter Test` and select `Test` folder). Always place testing classes in a 'mirrored' location in the `test` folder; if your project has packages, create those mirrored packages in the `test` folder. When your implementation is complete, the Package Explorer view of the project should like similar to the following snapshot.



You are to test each non-constructor method individually; use your tests from Lab 1 as a starting point for your tests in Lab 2. When you execute your `junit` tests, ***no output*** should be produced; we are seeking only a 'green' output indication in Eclipse.



Submitting Source Code

Your code should be well documented, including docstring comments of methods, blocks of code, and header comments in ***each*** file.

Testing code needs fewer comments as they should be self-descriptive; however, it is recommended that each individual test or family of tests be numbered and have a brief comment.

Header Comments

Your program must use the following standard comment at the top of *each source code file*. Copy and paste this comment and modify it accordingly.

```
/**
 * Write a succinct, meaningful description of the class here. You should avoid wordiness
 * and redundancy. If necessary, additional paragraphs should be preceded by <p>,
 * the html tag for a new paragraph.
 *
 * <p>Bugs: (a list of bugs and / or other problems)
 *
 * @author <your name>
 * @date   <date of completion>
 */
```

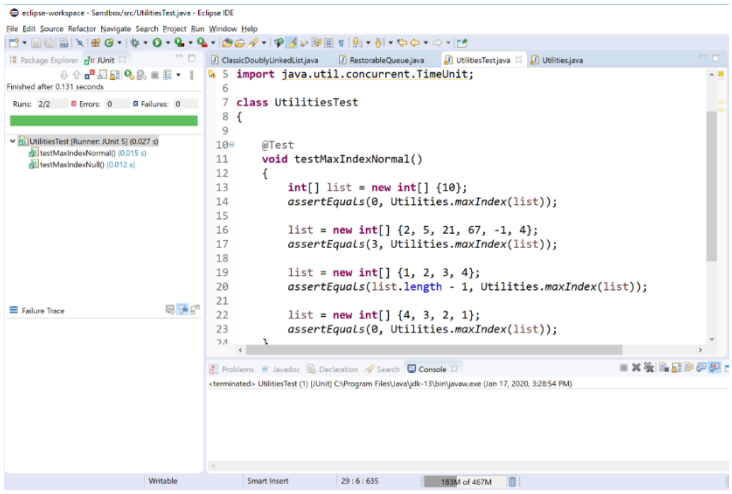
Inline Comments

Comment your code with a *reasonable amount of comments* throughout the program. Each method should have a comment that includes information about input, output, overall operation of the function, as well as any limitations that might raise exceptions; Javadoc comments are ideal. Each *block* of code (3-4 or more lines in sequence) in a function should be commented.

It is ***prohibited*** to use *long* comments to the right of lines of source code; attempt 80 to 100 character-wide text in source code files.

Submitting; Proof of Program Execution

Execute your code and take a screenshot of the associated output console. Place these screenshots into a word processing document (Word, OpenOffice, GoogleDocs, etc.). If multiple screenshots are necessary, label each clearly. Please make sure to crop and enlarge the screenshots so that the picture and / or text is clear (and doesn't strain my old eyes). For example, ***the screenshot below is not appropriately sized*** although it contains ideal information (output console, code, etc.). Create a PDF of this document and call it `evidence.pdf`.



Source Code Files

You are to submit your entire project folder (including any files provided to you).

Final Submission File

In a folder named `lab`, place (1) the project code folder and (2) `evidence.pdf`. Zip folder `lab` and label that zip file as `lab.zip`. This zip file is to be submitted via Moodle.