FPP Standardized Programming Exam
June, 2017

This 90-minute programming test measures the success of your FPP course by testing your new skill level in two core areas of the FPP curriculum: OO programming (specifically, polymorphism) and data structures. You will need to demonstrate a basic level of competency in these areas in order to move on to MPP.

Your test will be evaluated with marks "Pass" or "Fail." A "Pass" means that you have completed this portion of evaluation only; your professor will evaluate your work over the past month to determine your final grade in your FPP course, taking into account your work on exams and assignments. A "Fail" means you will need to repeat FPP, with your professor's approval.

There are two programming problems to solve on this test. You will use the Java classes that have been provided for you in an Eclipse workspace. You will complete the necessary coding in these classes, following the instructions provided below. In order to pass, you must get a score of at least 70% *on each of the two problems*.

Problem 1. **[Data Structures]** In your `prob1` package, you will find a class `DoublyLinkedDeleteFirst`. For this problem, you must implement the following methods

```
String deleteFirst()
boolean isEmpty()
```

This `deleteFirst()` method removes the node in position 1 of the list and returns the string contained in that node. The `isEmpty()` method returns `true` if there are no items in the list, `false` otherwise.

If `deleteFirst`() is called on a list that contains 0 or 1 element, an `IllegalStateException` must be thrown.

A `toString` method has been provided so you can test your code.

Example. Suppose your list has these values:

`["Bob","Bill","Tom"]`

After executing `deleteFirst`, the list should contain these elements (in this order):

`["Bob", "Tom"]`

Requirements for Problem 1:

(1) Your code must run correctly for lists containing any number of elements, including an empty list.
(2) No data may be placed in the header node.
(3) `IllegalStateException` must be thrown when `deleteFirst` is called on your list when it has only 0 or 1 element.
(4) You must use Java's `IllegalStateException` class (you must not create your own exception class for this).
(5) You may not introduce any new instance variables or instance methods, and you may not modify the other methods in `DoublyLinkedDeleteFirst`.
(6) The `Node` class contained in `DoublyLinkedDeleteFirst` must not be modified (in particular, no constructor other than the default constructor should be used to construct instances of `Node`).
(7) During execution, each `Node` in your `DoublyLinkedDeleteFirst` must have correct values for the `next` and `previous` `Node`s.
(8) There should be no compiler or runtime errors. In particular, no `NullPointerException`s should be thrown during execution.