Tafita Rakotozandry
CS150 Lab
09/19/2020

Lab 5 Report

INTRODUCTION

In this lab, a wrapper node class for the Integer value called Cell was created. Using that, the IntegerList class implements a linked list abstract data structure.Two main functions were created append() and toString(). The overall goal is to measure the run time execution of these functions using a different number of items. The goal of the lab is to introduce recursion and inheritance.

Unit Tests

The class Cell and IntegerList were unit tested. The method toString() and append() were unit tested together for each respective class. It was to reduce the coding time and to minimize the size of the overall lab folder. The method is Empty was also unit tested.

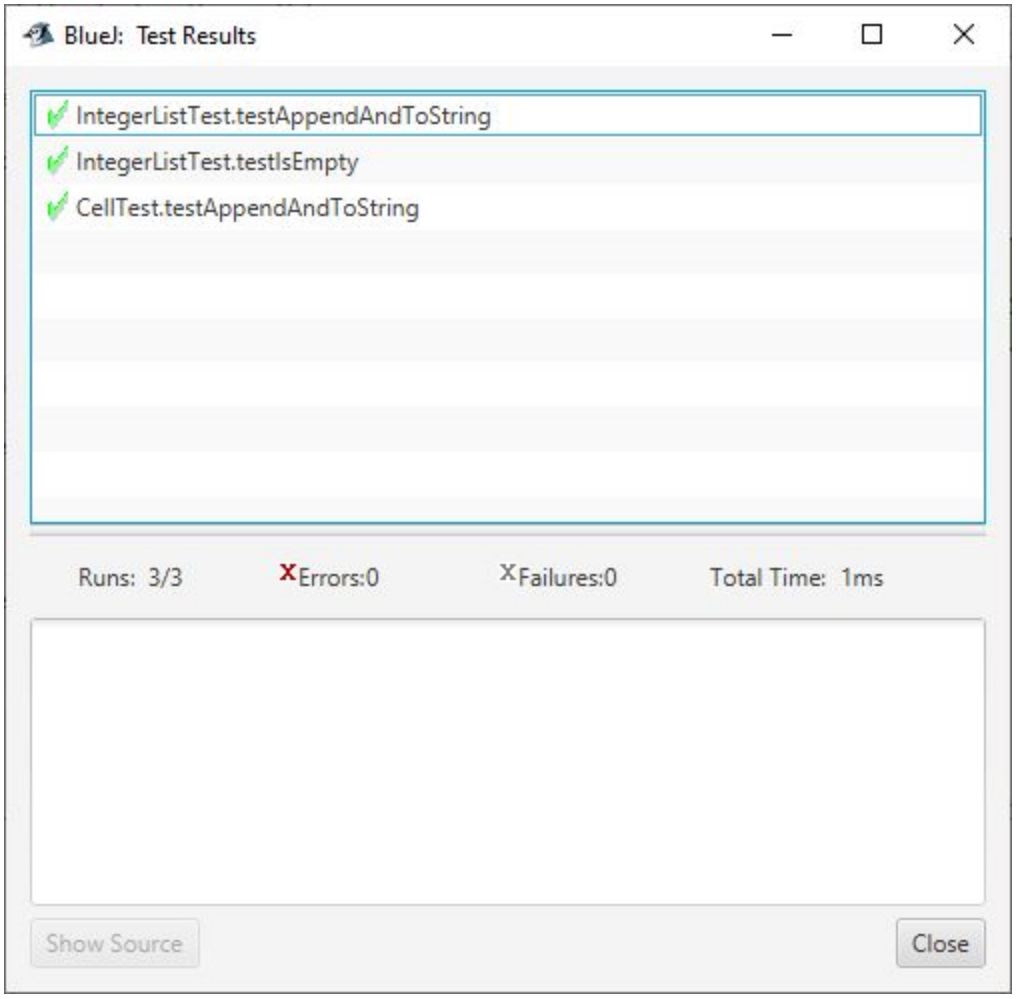
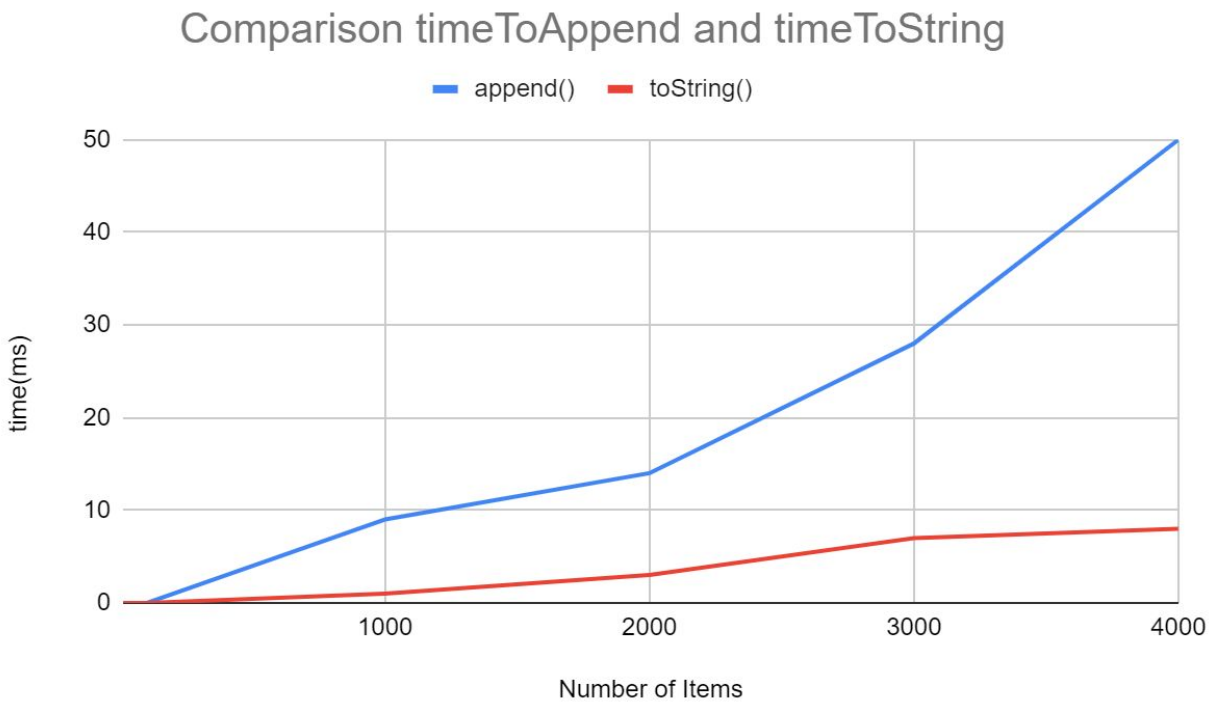


Figure 1: Screenshot Unit Testing Window

Required Output



In the graph above the red line represents the change in time in function of the number of items of the toString() method and the blue one is for append() method. Interpreting the graph, it is clear to say that appending takes more time than calling to String. It makes sense because when realizing appending new value, the root is always the first to be checked if it is null. If it is not, a recursion will take place until the next is null. On the other hand, toString goes through the list through a recursive method once and returns the result.

Trouble Report

One of the constraints that happened when experimenting the program is that there is StackOverflow when the number of items is high. Everytime Java is calling a method, a stack is

created inside the computer. When realizing a recursion, a function calls itself which calls itself until it reaches the bottom condition. The stack is therefore full when the number of items is too high because many functions are called. It is one of the risk of recursion if it is not well handled .

References

Michael Kölling (2015). Unit Testing in BlueJ https://www.bluej.org/tutorial/testing-tutorial.pdf

Weiss, M. A. (1998). Data structures and problem solving using Java. ACM SIGACT News, 29(2), 42-49.