Community College of Philadelphia

CSCI211 – Data Structures and Algorithms

Storm List 2

Overview

Using C++, Java, or Python, design a program that creates a **doubly** linked list, where each node contains an object with information about a certain tropical cyclone between the years 1900-2010.

Provided with the homework assignment is a CSV file containing entries for the costliest tropical cyclones to impact the United States mainland during this time period. This is the same csv file that was used in Assignment 3.

Your program must read in the data contained in this CSV file to set the fields/attributes of the objects stored in your linked list's nodes.

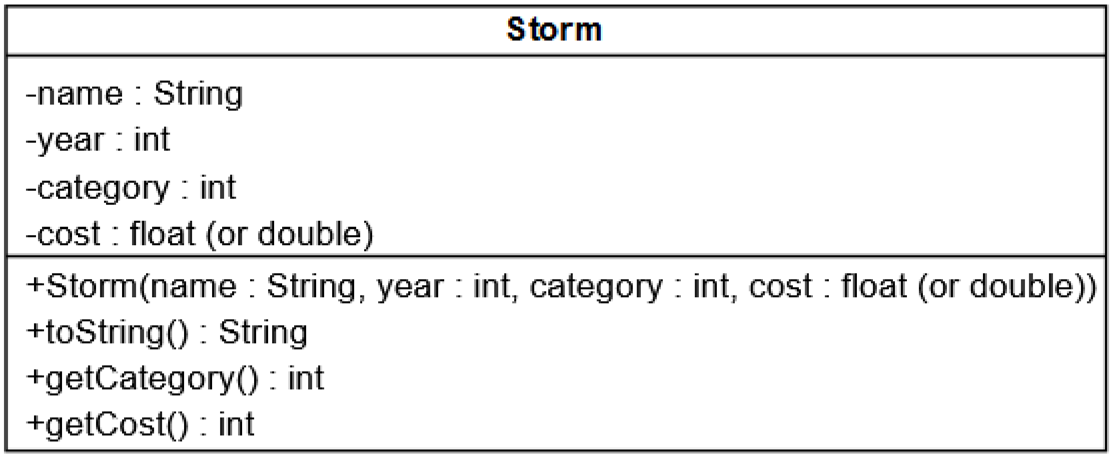
See the following pages of these instructions for details about the necessary classes/objects and an outline of what will happen in your main program.

If you need a refresher on reading files and tokenizing Strings, here are some resources:

- C++
  - Reading/Writing CSV Files | Tokenizing Strings
- Java
  - Reading/Writing CSV Files
- Python
  - Reading/Writing CSV Files

You can use the above resources or other resources for parsing the CSV file, but be sure to leave a comment crediting the source if you do. *How* your program reads the data from the CSV file into your objects isn't being graded, just that the program *is* reading the data from the CSV file into your objects.

<u>Object</u>



```
Storm
-name : String
-year : int
-category : int
-cost : float (or double)
+Storm(name : String, year : int, category : int, cost : float (or double))
+toString() : String
+getCategory() : int
+getCost() : int
```

The class can be implemented as either a struct (C++) or a class and must **only** contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.
    - The **toString** method must return a string in one of the following formats:

Category 1 through 5 (Hurricanes):

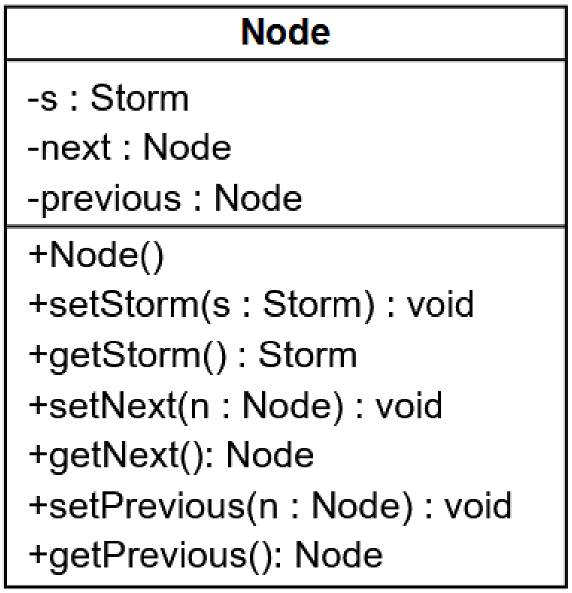
`Hurricane` **`name`** `- Year:` **`year`**`; Category:` **`category`**`; Cost: $`**`cost`**`B`

Category 0 (Tropical Storms):

`Tropical Storm` **`name`** `- Year:` **`year`**`; Cost: $`**`cost`**`B`

Neither the program's output nor the string's text need to be in bold; The bold font used in the examples above are only to emphasize where the data should be in the string that your toString method returns.

Node

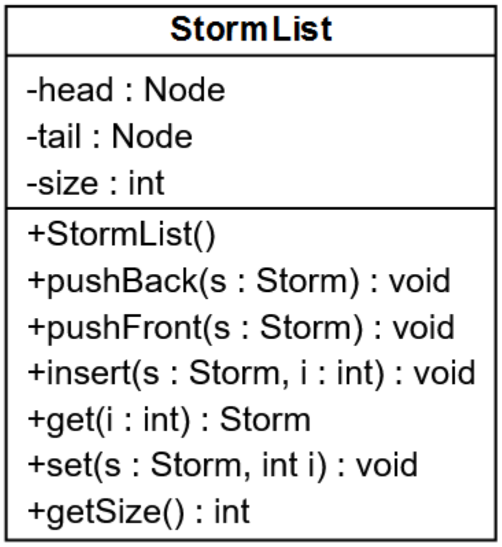


```
Node
-s : Storm
-next : Node
-previous : Node
+Node()
+setStorm(s : Storm) : void
+getStorm() : Storm
+setNext(n : Node) : void
+getNext(): Node
+setPrevious(n : Node) : void
+getPrevious(): Node
```

The class can be implemented as either a struct (C++) or a class and must **only** contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.
  - The constructor must initialize all fields to **null**.

Linked List


StormList
-head : Node
-tail : Node
-size : int
+StormList()
+pushBack(s : Storm) : void
+pushFront(s : Storm) : void
+insert(s : Storm, i : int) : void
+get(i : int) : Storm
+set(s : Storm, int i) : void
+getSize() : int

The class for your linked list must **only** contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram:
  - The constructor must initialize the fields to null, and the length to 0.
  - The two methods that push a new object to the front/back of the list.
  - The method (insert) that accepts a new object will store it in a new node at the provided position/index in the list.
  - The method (get) that retrieves an object from the list using the provided position/index.
  - The method (set) that accepts an object stores it to an existing node at the provided position/index in the list.
  - The method that returns the list's size.

Main Program

In a main function, complete the following:

1. Instantiate a new instance of your linked list.
2. Have the program read the CSV file's data:
   a. Parse the fields from the line/record.
   b. Instantiate an instance of a new object, providing its constructor with the data parsed from the current line/record.
      i. Push the first 15 objects to the back of the list.
      ii. Push the next 14 objects to the front of the list.
      iii. Insert the last object to position 14.
3. Sort the list by category in ascending order.
   a. After sorting, iterate through the list, printing information about each object using its toString method.
4. Sort the list by cost in ascending order.
   a. After sorting, iterate through the list, printing information about each object using its toString method.

You can do all the above in the main method or break it up into separate functions that the main method calls.

Other Requirements

- **Important: You may not use a List, ArrayList, Vector, or similar data structure built-in to your chosen programming language in place of your linked list class. You must implement a linked list and its necessary components/classes yourself.**

- Your classes/structs must closely match the UML class diagrams provided in these instructions. Slight deviations in field, method, parameter names are OK, but **do not** include additional methods or fields in these structs/classes; Access modifiers (public/private) must be as they are seen in the UML class diagrams.
  See here if you need a refresher on UML class diagrams

- **Do not change or modify the CSV file**. I will be using the provided CSV file when I test your program.

- Be sure to use comments to adequately document your source code.

Grading

See Assignment Rubric in Canvas.