Middle East Technical University | Department of Computer Engineering

CENG 351

Data Management and File Structures

Fall '2016-2017

Programming Assignment 2 - B+ Trees

Due Date: January 8, 2016, 23:59

1 Story

Merry Christmas! With the advancement of technology, Santa Claus has decided to become a "Cyber" Santa Claus.
(I think he saw the Amazon Drone and got a bit jealous.)

Before investing on drones, he decided to build a database for the gifts, and he needs your help. You are going to create a primary B+ tree, with options of adding gifts and searching.
(No gift is being removed from Santa's inventory, never!)

2 TL; DR

- Implement missing methods in stated classes
- Use given attributes / methods
- Create your own attributes / methods that do not clash with provided ones.
- Parse input file / command line inputs
- Write output to command line
- Have fun with GUI. If you have time, dig the code and try to improve.
  (I'll give you a cookie if you improve the GUI.)

3 Requirements

3.1 GUI Classes and Main Class

GUI classes are already implemented for your convenience. You are not expected to modify anything in:

1. CengChristmasList.java
2. CengGift.java

3. CengNodeType.java

– GUIs

4. CengGUI.java
5. GUIInternalNode.java
6. GUILeafNode.java
7. GUILevel.java
8. GUITreeNode.java
9. WrapLayout.java

You can modify these files as you wish, but note that you will NOT upload these files, which means any changes in these files will NOT affect grading.

However, there are few methods/variables that is required by GUI to work. They're already marked in your source code.

Please note that you are allowed to add methods, attributes as you like. Note that using `instanceof` is not a good choice, generally.

The implementation is partly done, you are expected to use given methods/attributes, and implement missing methods in those classes:

3.2 CengTreeParser.java

```
public class CengTreeParser
{
  public static ArrayList<CengGift> parseGiftsFromFile(String filename)
  {
    ArrayList<CengGift> giftList = new ArrayList<CengGift>();

    // You need to parse the input file in order to use GUI tables.
    // TODO: Parse the input file, and convert them into CengGifts
  }

  public static void startParsingCommandLine() throws IOException
  {
    // TODO: Start listening and parsing command line -System.in-.
    // There are 4 commands:
    // 1) quit : End the app, gracefully. Print nothing, call nothing, just ↩
       break off your command line loop.
    // 2) add : Parse and create the gift, and call ↩
       CengChristmasList.addGift(newlyCreatedGift).
    // 3) search : Parse the key, and call CengChristmasList.searchGift(parsedKey).
    // 4) print : Print the whole tree, call CengChristmasList.printTree().

    // Commands (quit, add, search, print) are case-insensitive.
  }
}
```

3.3 CengTree.java

```
// CengTree.java

public class CengTree
{
  public CengTreeNode root;

  public CengTree(Integer order)
  {
    CengTreeNode.order = order;

    // TODO: Create an empty Tree
  }

  public void addGift(CengGift gift)
  {
    // TODO: Insert Gift to Tree
  }

  public ArrayList<CengTreeNode> searchGift(Integer key)
  {
    // TODO: Search within whole Tree, return visited nodes.
    // Return null if not found.

    return null;
  }

  public void printTree()
  {
    // TODO: Print the whole tree to console
  }
}
```

3.4 CengTreeNode.java

```
public abstract class CengTreeNode
{
  static protected Integer order;
  private CengTreeNode parent;
  protected CengNodeType type; // Type needs to be set for proper GUI. Check ↩
     CengNodeType.java.

  // GUI Accessors - do not modify
  public Integer level;
  public Color color;

  public CengTreeNode(CengTreeNode parent)
  {
    this.parent = parent;

    this.color = CengGUI.getRandomBorderColor();

    // TODO: Extra initializations, if necessary.
  }
}
```

3.5 CengTreeNodeInternal.java

```
public class CengTreeNodeInternal extends CengTreeNode
{
  private ArrayList<Integer> keys;
  private ArrayList<CengTreeNode> children;

  public CengTreeNodeInternal(CengTreeNode parent)
  {
    super(parent);

    // TODO: Extra initializations, if necessary.
  }
}
```

3.6 CengTreeNodeLeaf.java

```
public class CengTreeNodeLeaf extends CengTreeNode
{
  private ArrayList<CengGift> gifts;

  public CengTreeNodeLeaf(CengTreeNode parent)
  {
    super(parent);

    // TODO: Extra initializations, if necessary.
  }
}
```

4 Command Line Specifications

(Assumption: Every file needed is available in current folder.)

Usage:

```
javac *.java
```

```
java CengChristmasList <orderCount> <inputFileName> <enableGUI>
```

Ex: `java CengChristmasList 2 20InputSorted.txt True`

5 Commands

5.1 Adding Gift:

`add|-giftKey-|-giftName-|-giftMaterial-|-giftColor-`

Ex: `add|0|MacBook|Metal|Space Gray`

Output: None.

5.2 Searching Gift:

`search|-giftKey-`

Ex: `search|5`

Output:

- **If Found:** Print visited nodes, and the found gift. *Details are in Output Format section.*
- **Else:** Print `Could not find -giftKey-.`
- **Ex:** `Could not find 5.`

5.3 Printing Tree:

`print`

Output: Print every node, and every gifts. *Details are in Output Format section.*

5.4 Quit:

`quit`

Output: None. End the application, gracefully.

6 Input Format

Input file only holds gifts, line by line. A line is shown as: *(Delimiter is pipe, |)*

`-giftKey-|-giftName-|-giftMaterial-|-giftColor-`

Ex: `14|Barbie|Metal|Space Gray`

- There will not be any erroneous input, neither in inputFile, nor in command line.
- Gift keys are unique, and always numerous. Other attributes may not be unique.
- As in the example, gift attributes can have spaces.
- You are not expected to handle duplicates (gifts with same keys). Hence, it will not be tested.

7 Output Format

7.1 Internal Nodes:

Indices of internal nodes should be printed like:

```
<index>
---
</index>
```

Ex:

```
<index>
19
37
55
73
</index>
```

7.2 Leaf Nodes:

Leaf nodes should start with `data` tags, and `record` tag for every gift.

```
<data>
<record>-giftKey-|-giftName-|-giftMaterial-|-giftColor-
</data>
```

Ex:

```
<data>
<record>0|MacBook|Metal|Space Gray</record>
<record>1|Ball|Glass|Space Gray</record>
</data>
```

7.3 Tree Printing

Starting with the root, you will print the nodes in a depth-first order. Indentation will be done level-wise with tabs.

Hint: A similar leveling is already done in CengGUI.java.

```
<index>
7
</index>
	<index>
	3
	5
	</index>
		<data>
		<record>1|test1|test|test</record>
		<record>2|test2|test|test</record>
		</data>
		<data>
		<record>3|test3|test|test</record>
		<record>4|test4|test|test</record>
		</data>
		<data>
		<record>5|test5|test|test</record>
		<record>6|test6|test|test</record>
		</data>
	<index>
	9
	11
	</index>
		<data>
		<record>7|test7|test|test</record>
		<record>8|test8|test|test</record>
		</data>
		<data>
		<record>9|test9|test|test</record>
		<record>10|test10|test|test</record>
		</data>
		<data>
		<record>11|test11|test|test</record>
		<record>12|test12|test|test</record>
		<record>13|test13|test|test</record>
		<record>14|test14|test|test</record>
		</data>
```

7.4 Search Printing

Same rules apply with tree printing, except that you should print only the visited nodes while searching, and the found gift.

8 Submission

What to submit?

1. CengTreeParser.java
2. CengTree.java
3. CengTreeNode.java
4. CengTreeNodeInternal.java
5. CengTreeNodeLeaf.java

Archive them with:

```
tar -cvf <yourID>.tar.gz CengTreeParser.java CengTree.java CengTreeNode.java
CengTreeNodeInternal.java CengTreeNodeLeaf.java
```

It will be extracted with:

```
tar -xvf <yourID>.tar.gz
```

Note: If extracting results within a folder, your submission cannot be graded.

Note: If you are using an IDE (like Eclipse), do not change the package of the classes.

As mentioned in Command Line Specifications, your submission should be able to run with:

```
javac *.java
```

```
java CengChristmasList <orderCount> <inputFileName> <enableGUI>
```

Also as mentioned before, GUI will **NOT** be used while grading.

Good luck, have fun.

- engin