Live Coding 2

Oct 25.2015

Please pair with someone, collaborate and share unit-test cases. Each section lists points earned(100 means superb). For how to implement unit-tests see Appendix.

A. Create a Trie Node – 35 points

Background: Look up what a trie is in Wikipedia. We'll be implementing a trie. Very important: Each node will only contain a single letter, which we'll store as a string for maximum flexibility. Note: Some of the diagrams and algorithms assume a whole word in each node, please ignore those, we'll only be using a single letter per node.

a. Create a trie node that uses a dictionary for its children and a String for its data. (5 points)

```
class TrieNode {
      char letter;
      TreeMap<String,TrieNode> child;

      TrieNode(char aLetter);

      // adds a single child node for a letter
      void addChildForLetter(char aLetter);

      // get the child node corresponding to the supplied letter
      TrieNode getChildForLetter(char aLetter);

      int totalChildren(); // returns num of children

      // get child at index,where children are organized alphabetically
      TrieNode childAtIndex(int index);

      void addEndOfString()   { add(“”,new TrieNode(“”)); }
      boolean isEndOfString() { return get(“”) != null;   }
}
```

b. Write tests for each method. (15 points)

c. Implement each method. (15 points)

B. Create a basic Trie Class – Code 30 pts Unit Tests 30 pts

a. Create a Trie class. Please create stubs for `TrieClass()`, `inTrie(String str)`,`InsertString(String str)`. *(5 points)*

```
class Trie {
      TrieNode root;
      Trie();
      void insertString(String str); // insert string into the trie
      boolean inTrie(String str);    // is str inside Trie
}
```

b. Write tests for each method. (20 points)

c. Implement each method. See pseudo-code below. (15 points)

```
Trie() {
      root = new TrieNode("");
}

void insertString(String str) {
      TrieNode c = root;
      for ( int i=0; i<str.length(); i++ ) {
            char key = str.charAt(i);
            ...
      }
      c.addEndOfString();
}

boolean inTrie(String str) {
      TrieNode c = root;
      for ( int i=0; i<str.length(); i++ ) {
            char key = str.charAt(i);
            ...
      }
      return c.isEndOfString();
}
```

C. Print all the words in alphabetical order – Code 20 pts

a. Figure out a way to print all the words out in alphabetical order and implement a method in Trie that does that.

```
class Trie {
	…
	printStringsAlphabetically() {
		…
	}
}
```

D. If you get really stuck I've placed my IntelliJ project for reference up at:

a. Find the github repo at:
https://github.com/JonathanRitchey03/JavaTrie

Please let me know if I can help...

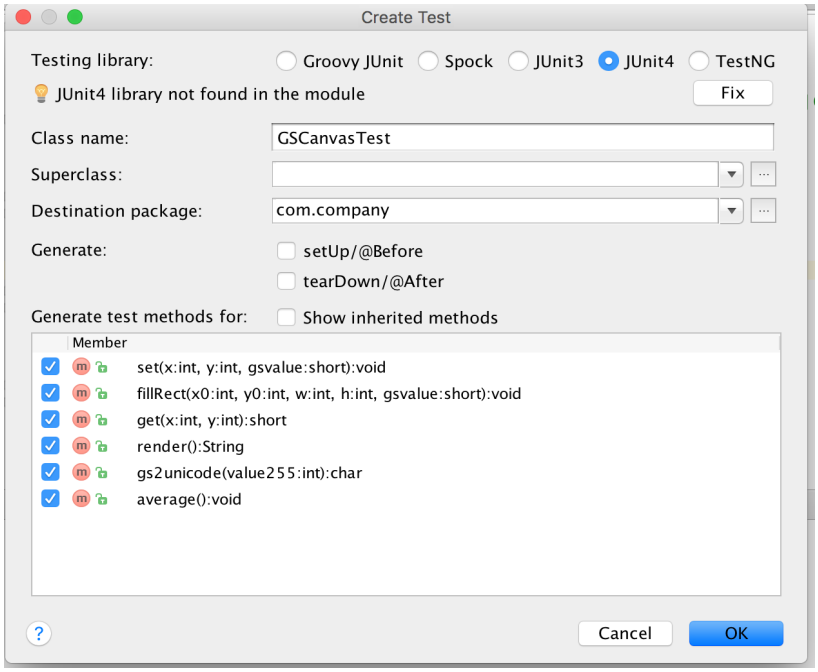
E. Bonus Challenge - Implement a N-ary Pretty Algorithm - Code 30 pts

a. Implement a method in Trie that prints out the trie to the console vertically. See if you can make it "pretty-print" the results. A very crude implementation is below:

```
void printTrie() {
	printTrie(root,"");
}
void printTrie(TrieNode c, String indent) {
	System.out.println(indent + c.data);
	indent += "  ";
	for ( int i = 0; i < c.totalKids(); i++ ) {
		printTrie(c.kidAtIndex(i), indent);
	}
}
```

Appendix – How to add Unit Tests to IntelliJ

1. In IntelliJ select the file you want to add tests to and press Command-Shift-T. (⌘ + shift + T)
2. Select "Create New Test".
3. On the pop that comes up, select JUnit 4. Now click on the "Fix" button so it includes the JUnit 4 package.
4. Click on all the methods. (see screenshot)



Here are some example unit tests:

```
@Test
public void testingCrunchifyAddition() {
assertEquals("Here is test for Addition Result: ", 30, addition(27, 3));
}

@Test
public void testingHelloWorld() {
assertEquals("Here is test for Hello World String: ", "Hello + World", helloWorld());
}
```

Appendix – How to add Unit Tests in PyCharm

https://confluence.jetbrains.com/display/PYH/Creating+and+running+a+Python+unit+test

1. In IntelliJ select the file you want to add tests to and press Command-Shift-T. (⌘ + shift + T)
2. Click on the check mark on all the method then press "OK". See screenshot:

Here are some example test cases:

```
def test_get_team_and_score_from_string(self):

    self.assertEqual(rank_teams.get_team_and_score_from_string("My Team 5"), ("My Team", '5'))
    self.assertEqual(rank_teams.get_team_and_score_from_string("My Team 5        "), ("My Team", '5'))

def test_update_rank_dict_for_team_by_points(self):

    rank_dict = {"teamA": 1}
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamA", 1)
    self.assertEquals(rank_dict, {"teamA": 2})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamA", 0)
    self.assertEquals(rank_dict, {"teamA": 2})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamB", 0)
    self.assertEquals(rank_dict, {"teamA": 2, "teamB": 0})
    rank_teams.update_rank_dict_for_team_by_points(rank_dict, "teamB", 3)
    self.assertEquals(rank_dict, {"teamA": 2, "teamB": 3})
```

Appendix – How to add Unit Tests in Visual C#

Check online. One link I found was:

http://www.codeproject.com/Articles/391465/Creating-Unit-tests-for-your-csharp-code

Appendix – How to add Unit Tests in JavaScript

If using WebStorm, JetBrains has support for this.

http://www.codeproject.com/Articles/391465/Creating-Unit-tests-for-your-csharp-code