```java
/* Ali Mojarrad
 * Comp282 Mon-Wed
 * Assignment 2 - FIX
 * 03/11/2015
 * AVL TREE IMPLEMENTATION */
class StringAVLNode {
	private String val;
	private int balance;
	private StringAVLNode left, right;

	// I believe we have all agreed that one constructor should suffice
	public StringAVLNode(String str) {

		balance = 0;
		val = str;

	}

	public int getBalance() {
		return balance;

	}

	public void setBalance(int bal) {
		balance = bal;
	}

	public String getVal() {
		return val;

	}

	public StringAVLNode getLeft() {
		return left;
	}

	public void setLeft(StringAVLNode pt) {
		left = pt;
	}

	public StringAVLNode getRight() {
		return right;
	}

	public void setRight(StringAVLNode pt) {
		right = pt;
	}

}

// StringAVLNode
class StringAVLTree {

	private StringAVLNode root;

	// the one and only constructor
	public StringAVLTree() {
```

```
		root = null;
	}

	// this is here to make it easier for me to write a test
	// program you would never do this in real life!
	public StringAVLNode getRoot() {
		return root;

	}

	// Rotate the node to the right
	private static StringAVLNode rotateRight(StringAVLNode t) {
		StringAVLNode pt;
		// pointer to left of t
		pt = t.getLeft();
		// perform rotation
		t.setLeft(pt.getRight());
		pt.setRight(t);
		// return new root
		return pt;
	}

	// Rotate the node to the left
	private static StringAVLNode rotateLeft(StringAVLNode t) {
		StringAVLNode pt;
		// pointer to right of t
		pt = t.getRight();
		t.setRight(pt.getLeft());
		pt.setLeft(t);
		// return new root
		return pt;
	}

	// For these next four, be sure not to use any global variables

	// Return the height of the tree
	public int height() {
		int height = 0;

		height = height(root);

		return height;
	}

	private int height(StringAVLNode pt) {
		int result;
		// if empty tree
		if (pt == null) {
			result = 0;
		} else {
	// Find Larger subtree recursively and increment result by 1 each time
			if (height(pt.getLeft()) > height(pt.getRight())) {
				result = height(pt.getLeft()) + 1;
			} else {
				result = height(pt.getRight()) + 1;
			}
		}
```

```
	return result;
}

// returns the number of leafs in the tree
// nodes with no children
public int leafCt() {

	int c = noChild(root);

	return c;
}// close method

private int noChild(StringAVLNode t) {

	int count = 0;

	// easiest case
	if (t == null) {
		// second easy case NO CHILDREN
	} else if (t.getRight() == null && t.getLeft() == null) {
		count++;
	}

	// only right children
	else if (t.getRight() != null && t.getLeft() == null) {
		count = noChild(t.getRight());
	}

	// only left children
	else if (t.getLeft() != null && t.getRight() == null) {
		count = noChild(t.getLeft());
	}

	// both children
	else {
		count = noChild(t.getRight());
		count += noChild(t.getLeft());
	}

	return count;
}

// Return the number of perfectly balanced AVL nodes
public int balanced() {

	int count = balanced(root);

	return count;
}

private int balanced(StringAVLNode t) {
	int count = 0;
	int leftC = 0;
	int rightC = 0;

	// empty tree
	if (t == null) {
```

```
            }

            // add one to count when you find balanced node
            else if (t.getBalance() == 0) {
                  count += 1;
            // if both children are not null add their difference to count
                  if (t.getRight() != null && t.getLeft() != null) {
                        leftC = balanced(t.getLeft());
                        rightC = balanced(t.getRight());
                        count += leftC + rightC;
                  }
            }
            // if only left child add its balance
            else if (t.getRight() == null && t.getLeft() != null) {
                  count += balanced(t.getLeft());
            }
            // if only right child add its balance
            else if (t.getLeft() == null && t.getRight() != null) {
                  count += balanced(t.getRight());
            }
            // otherwise
            else {
                  leftC = balanced(t.getLeft());
                  rightC = balanced(t.getRight());

                  count += leftC + rightC;
            }

            return count;
      }

      // Return the inorder successor or null if there is none
      public StringAVLNode successor(String str) {

            StringAVLNode pt = root;
            // set a flag
            boolean flag = false;

            // go through loop until your find successor or null
            while (flag != true) {

                  // if pt matches str exit loop
                  if (pt.getVal() == str) {
                        flag = true;
                  }
                  // if pt smaller go left
                  else if (str.compareTo(pt.getVal()) < 0) {
                        pt = pt.getLeft();

                  }
                  // if pt bigger go right
                  else if (str.compareTo(pt.getVal()) > 0) {
                        pt = pt.getRight();
                  }
            }
            // initialiaze succ node
            StringAVLNode succ = null;
```

Previous successor method relied on search method and other functions , new one is redone completely

```
            // set another flag
            boolean flag2 = false;
            // go through the loop to find successor
            while (flag2 != true) {
            // if pt has both children successor is pt and pt goes right
                  if (pt.getLeft() != null) {
                        if (pt.getRight() != null) {
                              succ = pt;
                              pt = pt.getRight();

                        }

                        // special case with no right child
                        else if (pt.getRight() == null) {
                              succ = pt;

                              flag2 = true;
                        }
                  }

            }

            return succ;

      }

      // inserts into AVL tree
      public void insert(String str) {
            root = insert(str, root);
      }

      private StringAVLNode insert(String str, StringAVLNode t) {
            int newBalance, oldBalance;
            // easiest case - empty tree
            if (t == null) {
                  t = new StringAVLNode(str);
                  newBalance = 0;
                  // already in the tree - do nothing
            } else if (t.getVal() == str) {
            }
            // smaller strings - go left
            else if (str.compareToIgnoreCase(t.getVal()) < 0) {
                  if (t.getLeft() == null) {
                        oldBalance = 0;
                        t.setBalance(t.getBalance() - 1);
                  } else {
                        oldBalance = t.getLeft().getBalance();
                  }
                  t.setLeft(insert(str, t.getLeft()));
                  newBalance = t.getLeft().getBalance();
                  // height increase
                  if (oldBalance == 0 && newBalance != 0) {
                        // fix the balance value
                        t.setBalance(t.getBalance() - 1);
                        // out of balance? must rotate
                        if (t.getBalance() == -2) {
                              // single rotation and balance update
```

```
                        if (t.getLeft().getBalance() == -1) {
                              t = rotateRight(t);
                              t.setBalance(0);
                              t.getRight().setBalance(0);
                        }
                        // double rotation and balance update
                        else {
                              t.setLeft(rotateLeft(t.getLeft()));
                              t = rotateRight(t);
                              if (t.getBalance() == 1) {
                                    t.setBalance(0);
                                    t.getRight().setBalance(0);
                                    t.getLeft().setBalance(-1);
                              } else if (t.getBalance() == -1) {
                                    t.setBalance(0);
                                    t.getLeft().setBalance(0);
                                    t.getRight().setBalance(1);
                              }
                              // bal=0
                              else {
                                    t.setBalance(0);
                                    t.getRight().setBalance(0);
                                    t.getLeft().setBalance(0);
                              }
                        }
                  }
            }
      } else if (str.compareToIgnoreCase(t.getVal()) > 0) {
            if (t.getRight() == null) {
                  oldBalance = 0;
                  t.setBalance(t.getBalance() + 1);
            } else {
                  oldBalance = t.getRight().getBalance();
            }
            t.setRight(insert(str, t.getRight()));
            newBalance = t.getRight().getBalance();
            // heigh increase?
            if (oldBalance == 0 && newBalance != 0) {
                  // fix the balance value
                  t.setBalance(t.getBalance() + 1);
                  // out of balance ? must rotate
                  if (t.getBalance() == 2) {
                        // single rotation and balance update
                        if (t.getRight().getBalance() == 1) {
                              t = rotateLeft(t);
                              t.setBalance(0);
                              t.getLeft().setBalance(0);
                        }
                        // double rotation and balance update
                        else {
                              t.setRight(rotateRight(t.getRight()));
                              t = rotateLeft(t);
                              if (t.getBalance() == 1) {
                                    t.setBalance(0);
                                    t.getRight().setBalance(0);
                                    t.getLeft().setBalance(-1);
                              } else if (t.getBalance() == -1) {
```

```
                                    t.setBalance(0);
                                    t.getLeft().setBalance(0);
                                    t.getRight().setBalance(1);
                              }
                              // bal=0 case
                              else {
                                    t.setBalance(0);
                                    t.getRight().setBalance(0);
                                    t.getLeft().setBalance(0);
                              }
                        }
                  }
            }
            return t;
      }

      // delete method
      public void delete(String str) {
            root = delete(root, str);
      }

      private StringAVLNode delete(StringAVLNode t, String str) {

            int oldBalance, newBalance;

            // empty tree
            if (t == null) {

            }
            // go left if smaller
            else if (str.compareToIgnoreCase(t.getVal()) < 0) {

                  // get the oldBal
                  if (t.getLeft() == null) {
                        // special case where node is not in the tree
                        oldBalance = 99;
                  }

                  else {
                        oldBalance = t.getLeft().getBalance();
                  }

                  t.setLeft(delete(t.getLeft(), str));

                  // gets newBal
                  if (t.getLeft() == null) {

                        newBalance = 99;
                  }

                  else {
                        newBalance = t.getLeft().getBalance();
                  }

                  // heigh increased?
                  if (oldBalance == 0 && newBalance == 99 || oldBalance != 0
```

Many many changes were made to delete and replace method. Made from scratch again but this time starting from the easy cases and debugging using test result to fix the method as it goes along

```
                                && newBalance == 0) {
                        // update balance
                        t.setBalance(t.getBalance() + 1);

                        // need rotation?
                        if (t.getBalance() == 2) {

                                // 3 cases for balance update and rotation
                                // single rotation
                                if (t.getRight().getBalance() == 1) {
                                        t = rotateLeft(t);
                                        t.setBalance(0);
                                        t.getLeft().setBalance(0);

                                }
                                // single rotation
                                else if (t.getRight().getBalance() == 0) {
                                        t = rotateLeft(t);

        t.getLeft().setBalance(t.getLeft().getBalance() - 1);
                                        t.setBalance(t.getBalance() - 1);

                                }
                                // double rotation
                                else {
                                        t.setLeft(rotateLeft(t.getLeft()));
                                        t = rotateRight(t);

                                        if (t.getBalance() == 1) {
                                                t.setBalance(0);
                                                t.getRight().setBalance(0);
                                                t.getLeft().setBalance(-1);
                                        }

                                        else if (t.getBalance() == -1) {
                                                t.setBalance(0);
                                                t.getRight().setBalance(1);
                                                t.getLeft().setBalance(0);
                                        }
                                        // if balance ==0
                                        else {
                                                t.setBalance(0);
                                                t.getRight().setBalance(0);
                                                t.getLeft().setBalance(0);
                                        }
                                }
                        }
                }
        }
        // if bigger go right
        else if (str.compareTo(t.getVal()) > 0) {

                // get the oldBal
                if (t.getRight() == null) {
                        // special case where node is not in the tree
                        oldBalance = 99;
                }
```

```
                else {
                        oldBalance = t.getRight().getBalance();
                }

                t.setRight(delete(t.getRight(), str));

                // gets newBal
                if (t.getRight() == null) {
                        newBalance = 99;
                }

                else {
                        newBalance = t.getRight().getBalance();
                }

                // height increased?
                if (oldBalance != 0 && newBalance == 0 || oldBalance == 0
                                && newBalance == 99) {

                        t.setBalance(t.getBalance() - 1);

                        if (t.getBalance() == -2) {

                                // need rotaiton?
                                // 3 cases rotation and balance update
                                // single rotation
                                if (t.getRight().getBalance() == -1) {
                                        t = rotateRight(t);
                                        t.setBalance(0);
                                        t.getRight().setBalance(0);
                                }
                                // single rotation
                                else if (t.getRight().getBalance() == 0) {
                                        t = rotateRight(t);

    t.getRight().setBalance(t.getRight().getBalance() + 1);
                                        t.setBalance(t.getBalance() + 1);

                                }
                                // double rotation
                                else {
                                        t.setRight(rotateRight(t.getRight()));
                                        t = rotateLeft(t);

                                        if (t.getBalance() == 1) {
                                                t.setBalance(0);
                                                t.getLeft().setBalance(0);
                                                t.getRight().setBalance(1);
                                        }

                                        else if (t.getBalance() == -1) {
                                                t.setBalance(0);
                                                t.getLeft().setBalance(-1);
                                                t.getRight().setBalance(0);
                                        }
```

```
                                    // balance ==0
                                    else {
                                          t.setBalance(0);
                                          t.getLeft().setBalance(0);
                                          t.getRight().setBalance(0);
                                    }
                              }
                        }
                  }
            }

            // Easies cases
            else {
                  // if its a leaf and node has no children
                  if (t.getRight() == null && t.getLeft() == null) {
                        t = null;
                  }

                  // delete node only has a left child
                  else if (t.getRight() == null && t.getLeft() != null) {
                        t = t.getLeft();

                  }
                  // delete node only has a right child
                  else if (t.getRight() != null && t.getLeft() == null) {
                        // parent.setRight(t.getRight());
                        t = t.getRight();

                  }
                  // node has both children so we can have replace it
                  else {
                        oldBalance = t.getBalance();

                        // finds the replacement node and moves it up
                        t = replace(t, null, t.getLeft());

                        newBalance = t.getBalance();

// check after replacing if rotation is needed and balance update CASE 1
                        if (newBalance != 0 && oldBalance == 0 ||
                              newBalance == 0 && oldBalance != 0)
                        {

                              t.setBalance(t.getBalance() - 1);

                              if (t.getBalance() == 2) {
                                    // single rotatation
                                    if (t.getRight().getBalance() == 1) {
                                          t = rotateLeft(t);
                                          t.setBalance(t.getBalance() - 1);
                        t.getLeft().setBalance(t.getLeft().getBalance() - 2);
                                    }
                  // double rotation and balance update
                  else if (t.getRight().getBalance() == -1) {

                  t.setRight(rotateRight(t.getRight()));
```

```
                    t = rotateLeft(t);

                    t.getRight().setBalance(t.getRight().getBalance() + 1);

                    t.getLeft().setBalance(t.getLeft().getBalance() - 2);
                                        }

                                  }
// check after replacing if rotation is needed and balance update CASE 1
                                  else if (t.getBalance() == -2) {
                                        // single rotation
                                        if (t.getLeft().getBalance() == -1) {
                                              t = rotateRight(t);
                                              t.setBalance(0);
                                              t.getRight().setBalance(0);
                                        }

                                        // double rotation
                                        else {
                                              t.setLeft(rotateLeft(t.getLeft()));
                                              t = rotateRight(t);

                                              if (t.getBalance() == 1) {
                                                    t.setBalance(0);
                                                    t.getRight().setBalance(0);
                                                    t.getLeft().setBalance(-1);
                                              }

                                              else if (t.getBalance() == -1) {
                                                    t.setBalance(0);
                                                    t.getRight().setBalance(1);
                                                    t.getLeft().setBalance(0);
                                              }

                                              else {
                                                    t.getRight().setBalance(0);
                                                    t.getLeft().setBalance(0);
                                              }
                                        }
                                  }
                            }
                      }
                      return t;
            }

// The code to find and replace the node being deleted must be recursive
// so that we have easy access to the nodes that might have balance changes
            public StringAVLNode replace(StringAVLNode t, StringAVLNode prev,
                        StringAVLNode replacement) {

                      int oldBalance;
                      int newBalance;
                      // if we get to the replacement node
                      if (replacement.getRight() == null) {

                            // if replacement is not child
```

```
                    if (prev != null) {

                          // replace t with replacement node
                          replacement.setLeft(t.getLeft());
                          replacement.setRight(t.getRight());
                          t = replacement;

                    }

                    // move the replacement node
                    replacement.setRight(t.getRight());
                    replacement.setBalance(replacement.getBalance() + 1);
                    t = replacement;
                    t.setBalance(t.getBalance() + 1);

              } else {
                    // get the old balance
                    oldBalance = t.getBalance();
                    t = replace(t, replacement, replacement.getRight());

                    // find the new balance
                    newBalance = t.getRight().getBalance() + 1;

                    // heigh increased?
                    if (oldBalance == 0 && newBalance != 0) {

                          // update balance
                          t.setBalance(t.getBalance() + 1);
                          // need rotation?
                          if (t.getBalance() == 2) {
                                // single rotation
                                if (t.getRight().getBalance() == 1) {

                                      t = rotateLeft(t);
                                      t.setBalance(0);
                                      t.getLeft().setBalance(0);
                                }
                                // single rotaiton
                                else if (t.getRight().getBalance() == 0) {
                                      t = rotateLeft(t);

                    t.getLeft().setBalance(t.getLeft().getBalance() - 1);
                                      t.setBalance(t.getBalance() - 1);

                                }
                                // double rotation
                                else {

                                      t.getRight().setBalance(1);
                                      rotateRight(t.getRight());
                                      rotateLeft(t);
                                      if (t.getBalance() == 1) {
                                            t.setBalance(0);
                                            t.getRight().setBalance(-1);
                                            t.getLeft().setBalance(0);

                                      } else if (t.getBalance() == -1) {
```

```
                                        t.setBalance(0);
                                        t.getLeft().setBalance(1);
                                        t.getRight().setBalance(0);

                                  } else {// bal=0
                                        t.setBalance(0);
                                        t.getRight().setBalance(0);
                                        t.getLeft().setBalance(0);
                                  }

                            }
                      }
                // height increased?
                if (newBalance != 0 && oldBalance == 0) {

                      // update balance
                      t.setBalance(t.getBalance() - 1);
                      // need rotation?
                      if (t.getBalance() == -2) {
                            // single rotation
                            if (t.getLeft().getBalance() == -1) {
                                  t = rotateRight(t);
                                  t.setBalance(0);
                                  t.getRight().setBalance(0);
                            }
                            // single rotaiton
                            else if (t.getLeft().getBalance() == 0) {
                                  t = rotateRight(t);

t.getRight().setBalance(t.getRight().getBalance() + 1);
                                  t.setBalance(t.getBalance() + 1);

                            }
                            // double rotation
                            else {

                                  t.setLeft(rotateLeft(t.getLeft()));
                                  t = rotateRight(t);
                                  // update balances
                                  if (t.getBalance() == 1) {
                                        t.setBalance(0);
                                        t.getRight().setBalance(0);
                                        t.getLeft().setBalance(-1);

                                  } else if (t.getBalance() == -1) {
                                        t.setBalance(0);
                                        t.getLeft().setBalance(0);
                                        t.getRight().setBalance(1);

                                  } else {// bal=0
                                        t.setBalance(0);
                                        t.getRight().setBalance(0);
                                        t.getLeft().setBalance(0);
                                  }
                            }
                      }
```

```
                        }
                }

                return t;
        }

        // name method
        public static String myName() {
                return "Ali Mojarrad";
        }

}
```