Red-Black Tree Debug

COMP 550 Homework 4b

Tye Zasacky

Collaborators: None

RedBlackNodeTests: Appears Correct

RedBlackTreeTests: 20 Failures, 2 Errors

Failed Test: findNodeFindsNode

Point of Failure: ‘Assert.assertEquals(expectedResult, actualResult)’

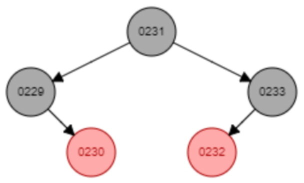
Input: Root (B,3)

Expected Output: (B,3)

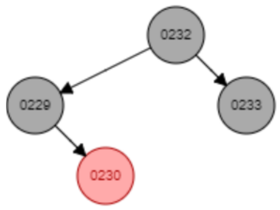
Actual Output: Null

Failed Test: deleteNodeDeletesRootWithGrandchildren

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

Input: 

, delete (231)

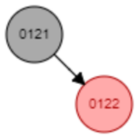
Expected Output: 

Actual Output:

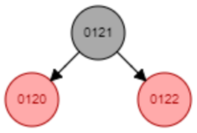
```
232
     /   \
229      229
     \          \
      230        230
```

Failed Test: insertNodeInsertsSecondChildAsLeft

Point of Failure: 'Assert.assertTrue(tree.isValid());'

Input: 

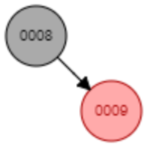
, insert (120)

Expected Output: 

Actual Output:
```
            121
           /   \
        120    120
```

Failed Test: findMaximumFindsMaximumNode

Point of Failure: 'Assert.assertEquals(expectedResult, actualResult);'

Input: 

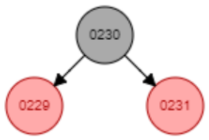
, findMaximumNode()

Expected Output: (9)

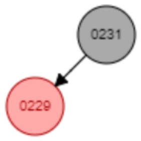
Actual Output: (8)

Failed Test: deleteNodeDeletesRootWithBothChildren

Point of Failure: 'Assert.assertTrue(tree.isValid());'

Input: 

, delete (230)

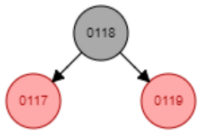
Expected Output: 

Actual Output:
```
      231
     /   \
  229     229
```

Failed Test: insertNodeInsertsLeftTwice

Point of Failure: Null Pointer Exception at RedBlackTree 151 (insertNode(insertFixup))

Input: insert(119), insert(118), insert(117)

Expected Output: 

Actual Output: Null Pointer Exception

Failed Test: deleteNodePassesComplexTest1

Point of Failure: 'Assert.assertTrue(tree.isValid());'

Input:

```
                 B
             /       \
        oldNode        D
        /   \       /      \
    Alpha   A      C          E
                 /   \      /  \
               Gam  Delt   Eps  Zeta
```

Expected Output:

```
                D
            /       \
          B           E
        /   \       /    \
      A       C   Epsilon  Zeta
     /       / \
  Alpha    Gam  Delta
```

```
Actual Output:          A
                      /     \
                 Alpha        B
                            /    \
                        Alpha      D
                                 /      \
                               C          E
                             /   \       /   \
                        Gamma   Delta  Epsilon  Zeta
```

Failed Test: deleteNodePassesComplexTest2

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:       D
           /      \
   oldNode          B
     /    \      /      \
Epsilon    E    A           C
              /    \      /    \
         Alpha    Beta Gamma   Delta
```

```
Expected Output:     B
                   /     \
                 A          D
              /    \      /     \
         Alpha    Beta   C          E
                       /   \      /
                  Gamma   Delta Epsilon
```

```
Actual Output:          B
                     /        \
                  A             D
               /      \       /        \
      Alpha        Beta    C            E
          \              /      \        /
           D         Gamma      Delta Epsilon
```

Failed Test: deleteNodePassesComplexTest3

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:         B
           /        \
   oldNode             D
     /      \       /        \
Alpha        A    C             E
```

```
Expected Output:         B
                     /        \
                  A            D
             /              /        \
        Alpha             C             E
```

```
Actual Output:           A
                     /        \
               Alpha            B
                            /        \
                       Alpha           D
                                      /     \
                                     C       E
```

Failed Test: deleteNodePassesComplexTest4

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:          D
             /         \
          B            oldNode
      /       \        /         \
   A           C    Epsilon        E
```

```
Expected Output:    D
                         /         \
                      B               E
                  /       \       /
               A           C    Epsilon
```

```
Actual Output:         D
                         /         \
                      B               E
                  /       \       /      \
               A           C Epsilon Epsilon
```

Failed Test: deleteNodePassesComplexTest5

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:                      B
                         /         \
                  oldNode          D
                  /       \        /      \
              Alpha        A    C           E
                                /      \
                          Gamma   Delta
```

```
Expected Output:     C

                  /          \

                 B               D

            /     \           /       \

         A     Gamma  Delta      E

      /

   Alpha

Actual Output: :   A

                  /      \

      Alpha         C

                  /          \

                 B               D

            /     \           /       \

      Alpha      Gamma  Delta      E
```

Failed Test: deleteNodePassesComplexTest6

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:                        D

                  /          \

                 B             oldNode

            /    \            /       \

         A             C   Epsilon    E

                  /      \

            Gamma   Delta
```

```
Expected Output:    C
                 /       \
               B           D
            /   \       /     \
         A     Gamma Delta     E
                            /
                       Epsilon
Actual Output:      C
                 /       \
               B           D
            /   \       /     \
         A     Gamma Delta     E
                            /     \
                       Epsilon   Epsilon
```

Failed Test: deleteNodePassesComplexTest7

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:                 B
                 /       \
            oldNode        D
            /     \      /     \
        Alpha       A  C         E
                              /    \
                         Epsilon  Zeta
```

```
Expected Output:     D
                 /       \
                B         E
             /     \       /     \
            A     C   Epsilon  Zeta
          /
     Alpha
Actual Output:   A
               /   \
          Alpha     B
                 /       \
            Alpha          D
                          /     \
                        C        E
                               /   \
                         Epsilon  Zeta
```

Failed Test: deleteNodePassesComplexTest8

Point of Failure: 'Assert.assertTrue(tree.isValid());'

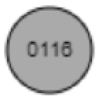
```
  Input:         D
            /       \
           B        oldNode
         /   \        /     \
       A      C  Epsilon     E
    /     \
Alpha   Beta
```

```
Expected Output:      B
                    /        \
                  A            D
               /    \        /     \
          Alpha  Beta   C         E
                                /
                         Epsilon
```

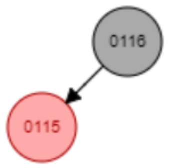
```
Actual Output:           B
                    /        \
                  A            D
               /    \        /     \
          Alpha  Beta   C         E
                                /     \
                         Epsilon    Epsilon
```

Failed Test: insertNodeInsertsRootsLeftChild

Point of Failure: 'Assert.assertTrue(tree.isValid());'

Input: 

, insert(115)

Expected Output: 

```
Actual Output:        116
                    /      \
                 115       115
```

Failed Test: insertNodePassesComplexTest2

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:      C
          /     \
        B           D
Expected Output:  C
                       /     \
                     B           D
                   /
                  A
Actual Output:      C
                       /     \
                     B          D
                   /    \
                  A        A
```

Failed Test: insertNodePassesComplexTest4

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

```
Input:  B
       /     \
     A          D
Expected Output: B
                       /     \
                     A          D
                               /
                             C
```

```
Actual Output: B
             /     \
           A          D
                   /    \
                 C       C
```

Failed Test: insertNodePassesComplexTest5

```
Point of Failure: Null Pointer Exception

Input:    C
       /
     A

Expected Output: B
                /     \
              A          C

Actual Output: Null Pointer Exception
```

Failed Test: insertNodePassesComplexTest6

```
Point of Failure: 'Assert.assertTrue(tree.isValid());'

Input: A
        \
          C

Expected Output: B
                /     \
              A          C

Actual Output: A
                \
                  B
                    \
                      C
```

Failed Test: findMinimumNodeFindsMinimumNode

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

Input: insert(n1(7)), insert(n2(8)), insert(n3(9)), getmin(n3)

Expected Output: 9

Actual Output: 9 (I don’t get it, this should be correct)

Failed Test: findMinimumFindsMinimumNode

Point of Failure: ‘Assert.assertTrue(tree.isValid());’

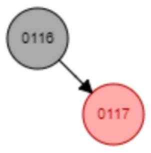
Input: insert(5), insert(6), insert(7), tree.findMinimum()

Expected Output: 5

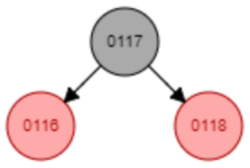
Actual Output: 5 (I don’t get it, this should be correct)

Failed Test: insertNodeInsertsRightTwice

Point of Failure: ‘Assert.assertTrue(tree.isValid());’



Input:



Expected Output:

```
Actual Output:   116
                    \
                     117
                        \
                         118
```

Fixes

1. Changed (RedBlackTree 18) in findNode
From: “if (this.root != RedBlackTree.nil) {“

To: “if (this.root == RedBlackTree.nil) {“

Tests Fixed by Change: findNodeFindsNode

2. Changed (RedblackTree 84) in insertNode
From: “} {“
To: “} else {“
Tests Fixed by Change: insertNodeInsertsSecondChildAsLeft, insertNodeInsertsRootsLeftChild, insertNodePassesComplexTest2, insertNodePassesComplexTest4
Tests Broken by Change: findMinimumFindsMinimumNode

3. Changed (RedBlackTree 84 & 85) in findMaximum
From: “while (node.hasLeftChild()) {
        node = node.getLeftChild();”
To: “while (node.hasRightChild()) {
        node = node.getRightChild();”
Tests Fixed by Change: findMaximumFindsMaximumNode

4. Changed (RedBlackTree 164) in insertFixup
From: “this.rightRotate(z.getParent().getParent());”
To: “this.leftRotate(z.getParent().getParent());”
Tests Fixed by Change: insertNodePassesComplexTest6, findMinimumNodeFindsMinimumNode, findMinimumFindsMinimumNode, insertNodeInsertsRightTwice

5. Changed (RedBlacktree 111) in deleteNode
From: “if (y.getParent() == y) {“
To: “if (y.getParent() == z) {“
Tests Fixed by Change: deleteNodeDeletesRootWithBothChildren, deleteNodePassesComplexTest1, deleteNodePassesComplexTest3-8

6. Changed (RedBlackTree 115) in deleteNode
From: “y.setRightChild(z.getLeftChild());”
To: “y.setRightChild(z.getRightChild());”
Tests Fixed by Change: deleteNodeDeletesRootWithBothChildren

7. Changed (RedBlackTree 146) in insertFixup
From: “z.getParent().setColor(Color.RED);”
To: “z.getParent().getParent().setColor(Color.RED);”
Tests Fixed by Change: insertNodeInsertsSecondChildAsLeft, insertNodePassesComplexTest2

8. Changed (RedBlackTree 207) in deleteFixup
From: “x.getParent().setColor(Color.BLACK);”
To: “x.getParent().setColor(Color.RED);”
Tests Fixed by Change: deleteNodePassesComplexTest2