Red-Black Binary Trees in Java

Intro and info about Red-Black Binary Trees

Red-Black Trees are binary trees based around the idea of balancing, meaning that no branch or path is twice the size of another. This is achieved by adding another property to the tree nodes, a color either black or red. In order to maintain balance, a red black binary tree must adhere to these properties:

1. Every node is either red or black
2. The root node is black
3. Every Null/Terminal leaf is black
4. If a node is red, both it's children must then be black
5. Every path from any given node to a descendent leaf contain the same number of black nodes This program implements a Red Black Tree as a generic java class (in the current code, for the sake of demo, the tree is of type integer), as well as a demo main function to test the functions of the tree.

Features

- Generic Red Black Tree Class
- Add and delete elements from the tree
- Display the tree in order (ascending), reverse order, and by level order

Technical Information

Required imports

- java.util.InputMismatchException
- java.util.Scanner

Enums

```
Color{ RED, BLACK } //for node color
```

Classes

Node

Members

```
Color color //Enum type defining if node is red or black
T data      //Generic type holding node data
Node left   //left child node
Node right  //right child node
```

Constuctors

```
public Node()           //sets color to black and other members to null
public Node(T _data)    //Sets data to node
public Node(T _data, Color _color)  //sets node data and color
```

Red Black Tree

Members

```
private Node root   //root node
private Node nilT   //the null/terminal leaf node, black color and null variables
```

Contructors

```
public RedBlackTree(T _headData)    //Initialize a tree with root node data
public RedBlackTree()               //Sets root to null
```

Functions

```
public boolean isEmpty()                //Returns if tree is empty
public Node searchFor(Node z, T _data)  //takes root node and searches for a
specific element
private int treeHeight(Node z)          //takes a given node to determine height
private Node treeMin(Node x)            //min value of tree and return the node
private Node treeSuccessor(Node x)      //Returns successor of a node
private void left_rotate(Node _x)       //completes left rotation about a given
node
private void right_rotate(Node _x)      //completes right rotation about a given
node
private void insertFix(Node z)          //fixes rules that are broken when
inserting a node
private void deleteFix(Node x)          //fixes rules that are broken when
deleting a node
public void insert(T _data)             //takes data and inserts it into tree
public T delete(Node z)                 //deletes a specified node from the tree
public void inorder_traverse_az(Node z) //Displays contents of tree in order
public void inorder_traverse_za(Node z) //Displays contents of tree in reverse
public void display_level_order(Node z) //Displays contents of tree by level order
private void display_single_level(Node z, int lvl)  //Implements recursion to
output one level of tree
public Node getRoot()                   //returns root node of tree
```

Example Input and Output

```
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:1
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:4
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:66
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:18
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:3
Elements in order: 1 4 18 66
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:-23
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
```

```
0. Quit
Enter choice:4
Elements in reverse order: 66 18 4 1 -23
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:1
Enter a number to input:100
1. Insert Item
2. Delete Item
3. Print in order
4. Print reverse order
5. Print Level Order
0. Quit
Enter choice:5
Elements in level order:
Level 0: 4
Level 1: 1 66
Level 2: -23 18 100
```