**Lab: Red-Black Trees and AA-Trees Lab**

This document defines the lab for ["Data Structures – Advanced (Java)" course @ Software University](https://softuni.bg/trainings/2814/data-structures-advanced-with-java-april-2020). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2335/03-Red-Black-Trees-and-AA-Trees-Lab).

Write Java code for solving the tasks on the following pages. Code should compile under the Java 8 and above standards you can write and locally test your solution with the Java 13 standard, however **Judge will run the submission with Java 10 JRE**. Avoid submissions with **features included after Java 10** release doing **otherwise** will result in **compile time error**.

Any code files that are part of the task are provided as **Skeleton**. In the beginning import the project skeleton, do not change any of the interfaces or classes provided. You are free to add additional logic in form of methods in both interfaces and implementations you are not allowed to delete or remove any of the code provided. Do not change the names of the files as they are part of the tests logic. **Do not change the packages** or move any of the files provided inside the skeleton if you have to add new file add it in the same package of usage.

Some **tests may be provided** within the skeleton – use those for local **testing and debugging**, however there **is no guarantee that there are no hidden tests added inside Judge**.

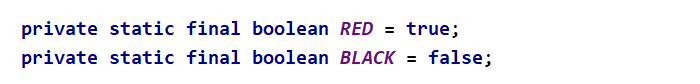
Please follow the exact instructions on uploading the solutions for each task. Submit as **.zip archive** the files contained inside **"...\src\main\java"** folder this should work for all tasks regardless of current DS implementation.

In order for the solution to compile the tests **successfully** the project **must** have **single** **Main.java** file containing single **public static void main(String[] args)** method even empty one within the **Main class**.

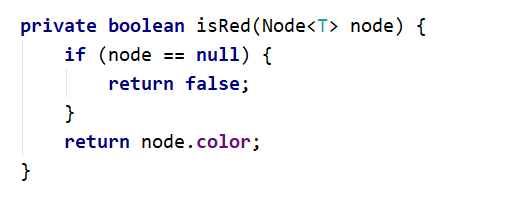
Some of the problem will have simple **Benchmark** **tests** inside the skeleton. You can try to run those with **different** **values** and **different** **implementations** in order to **observe** behaviour. However **keep** in mind that the result comes **only as numbers** and this data may be **misleading** in some situations. Also the tests are not started from the command prompt which may **influence** the **accuracy** of the results. Those tests are only added as an **example** of **different** **data** **structures** **performance** on their **common** operations.

## Check Red Node

Add the following constants to your RedBlackTree class:

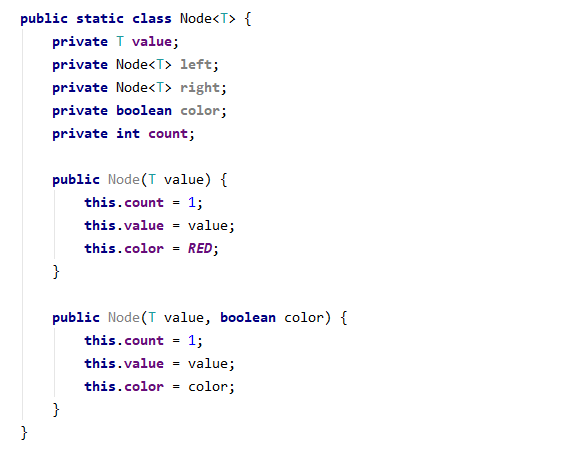


Now create a helper method that will check if a node is red:



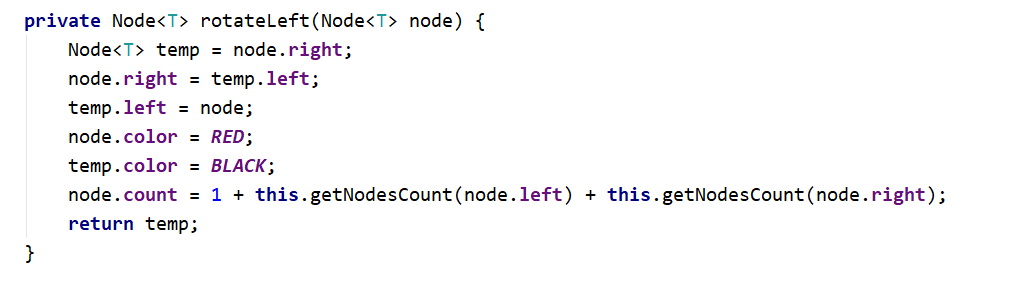
## Change the Node Data Structure

First, you will need to add a **color bit** to our node class:



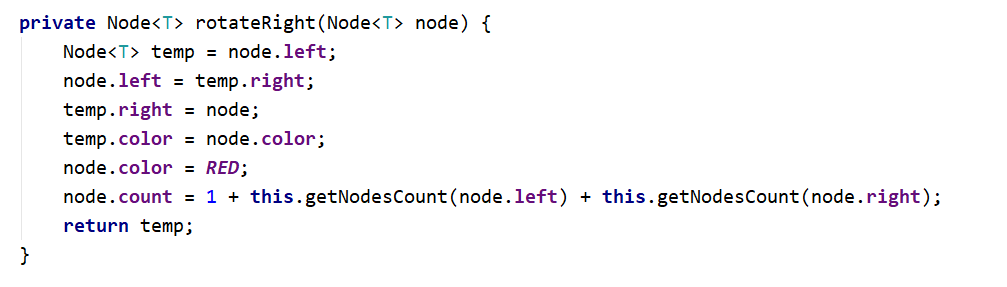
## Left Rotation

Create a method that will accomplish the left rotation for a given node.



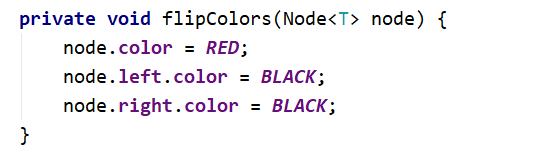
## Right Rotation

Create a method that will perform right rotation on a given node. The code is similar to the left rotation.



## Flip Colours

Implement a method that will make a node "**black**" and its children "**red**".



## Insert

Implement the existing insert() method. It should create new **red** node for every insert, **balance** the tree and **recolour** the nodes if needed.

**Hint:** There are hints inside the presentation.

## Run Unit Tests

