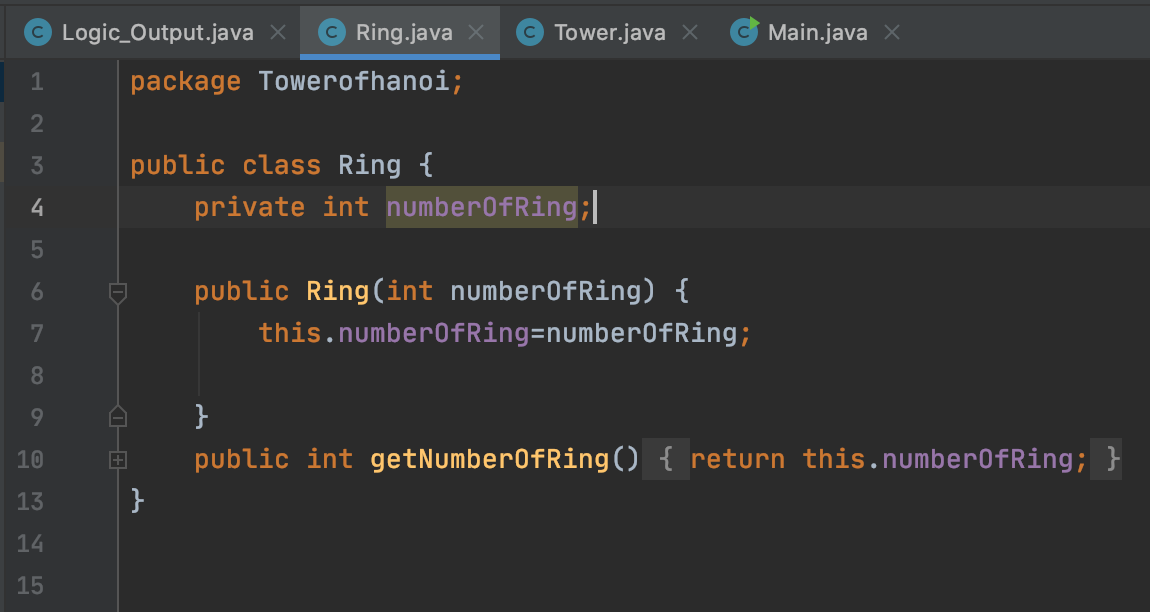
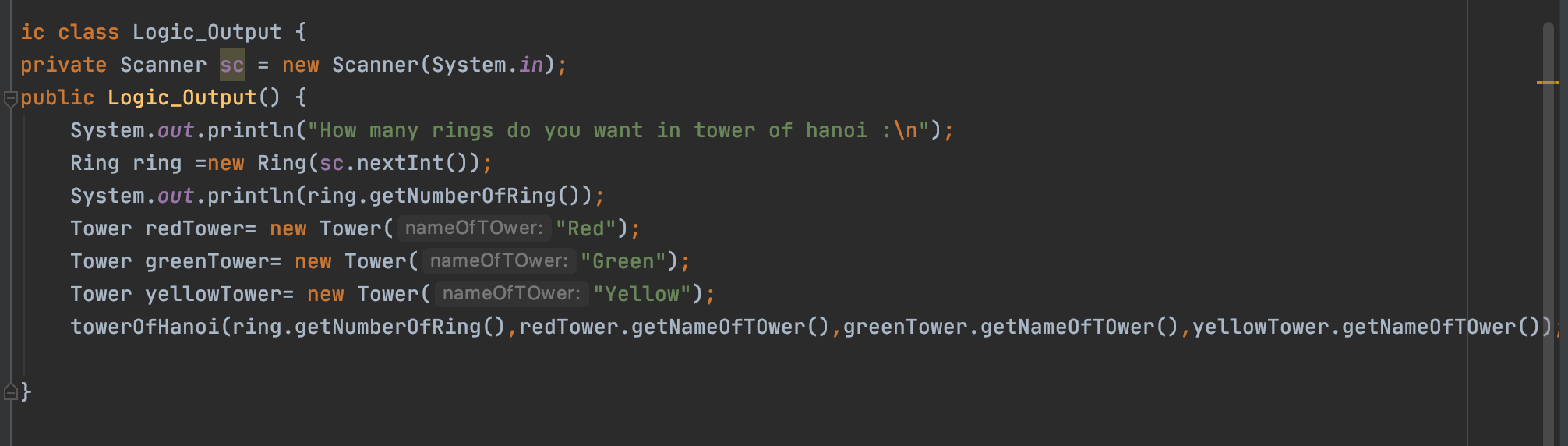
Final Report

Topic: Tower of Hanoi Submitted to: Kaveh Bakhtiyari

Submitted by: Gurwinder Singh

Tower of Hanoi is a very interesting game, while making Tower of Hanoi, I had to make sure that I do not break its rules such as I can only move one disk in a rod at a time. I cannot put a bigger size disk on the top of a smaller size disk either I can put a smaller disk on to a bigger disk in a respective order or I can put them to an empty rod. So, while taking care of all these rules and regulations. I made four classes in my project Logic\_Output, Ring, Tower, and Main. First, I would explain my Ring and Tower classes, where in my Ring class I have simply created a private variable called numberOfRing. Then I have made its constructor and getter. Exactly in my Tower class I have made a private variable called nameOfTOwer, then its constructor and getter as well just like I did for Ring class. The purpose of creating constructor and getter for these classes to access their private variables from outside classes as well. As you can see here:



Furthermore, I put all of my logic inside of a Logic\_Output class, where first I have imported a scanner form a Java library (import java.util.Scanner;) so that I can ask the users to provide number of disks as per their wish. Then inside of my Logic\_Output class I created a constructor called Logic\_Output() in which I made some objects of my Tower class such as redTower, greenTower, and yellowTower. Also, I have made an object of ring class as well that I used in a scanner (Ring ring =new Ring(sc.nextInt());). Moreover, in this constructor I have made a function called towerOfHanoi which is accessing all the instances that I have made from my Ring and Tower classes. This is how it looks:

Moreover, in order to define the main functioning of this Tower of Hanoi program I have used Java recursive method. I created a function name towerOfHanoi which is taking four arguments number of disks, beginning rod, ending rod, and auxiliary rod (middle rod, which is also known as temporary rod). This function is connected with Logic\_Output() constructor towerOfHanoi function in order to fetch the variables. Then I have put the condition which says if numberofrings is equal to 1 then it should move first disk from a beginning rod to an ending rod and then it would simply return. Moving forward, for n number of disks the complicity of this program becomes very hard so I set numberofrings equal to n –1 to solve this issue. For instance, in this towerOfHanoi(numberofrings-1, beg\_rod, mid\_rod, end\_rod); here you can see numberofrings-1 will decrease the rings over and over until it becomes 0 and also until my first conduction is true this function will automatically going to move disks from a beginning rod to an ending rod using an auxiliary rod. For example, if I have numberofrings = 3 then this function will work like this:

towerOfHanoi(numberofrings-1, beg\_rod, mid\_rod, end\_rod);

Where numberofrings == 3;

Function will work in this way:

towerOfHanoi(3-1, beg\_rod, mid\_rod, end\_rod);

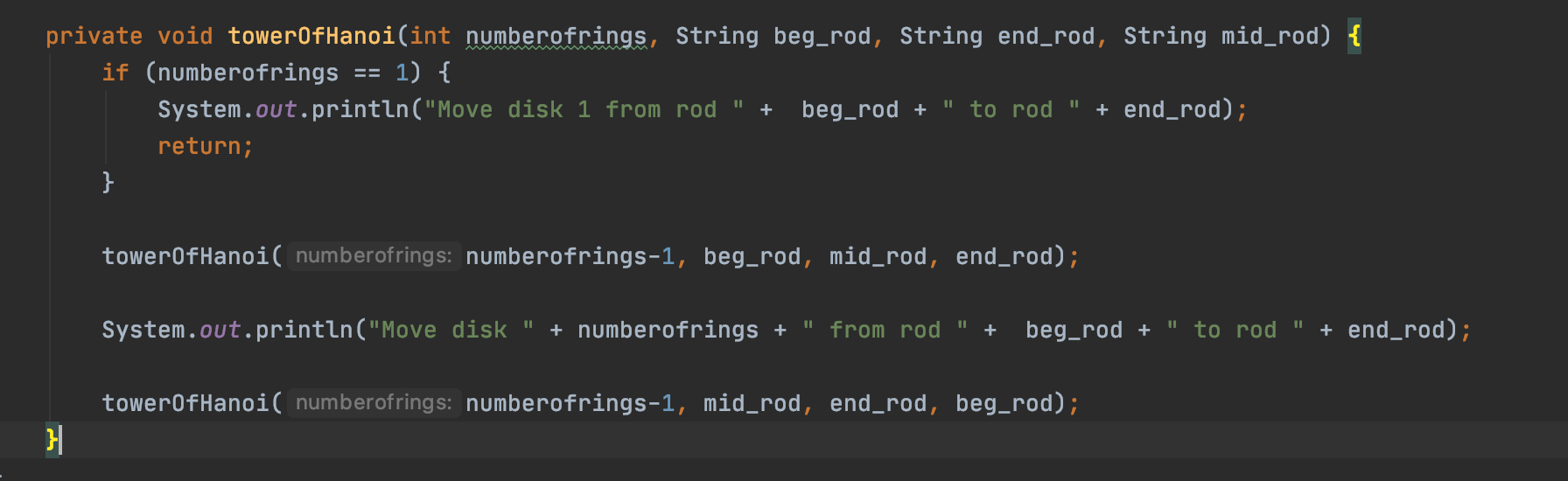
towerOfHanoi(2-1, beg\_rod, mid\_rod, end\_rod);

This function will execute numberofrings until they become zero. Also, at the same time it will move number of rings from a beginning rod to an ending rod using auxiliary rod. Auxiliary rod is acting like a temporary rod here. Its role is to store the rings temporally in itself until they get stored in a right rod.

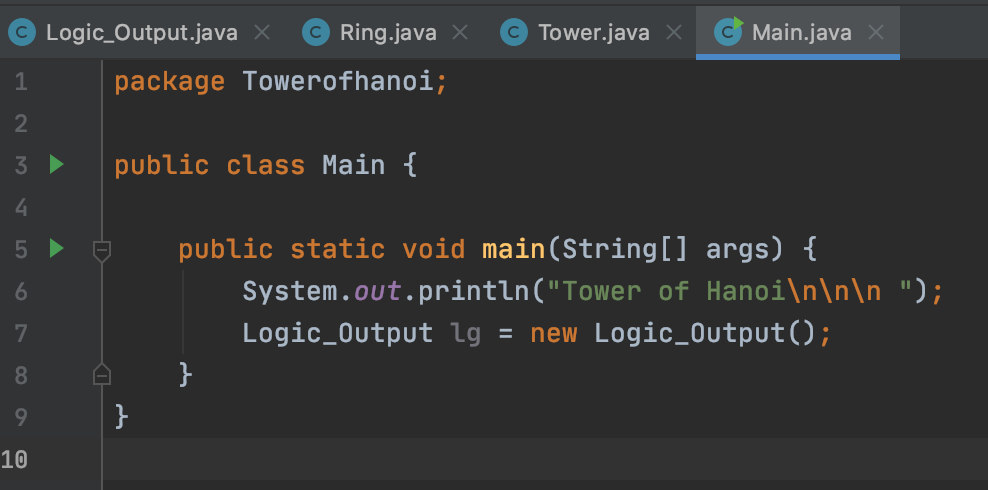
System.out.println("Move disk " + numberofrings + " from rod " + beg\_rod + " to rod " + end\_rod);

towerOfHanoi(numberofrings-1, mid\_rod, end\_rod, beg\_rod);

This method will be called until all the rings are moved to their new rod. As you can see here:



In addition, I have my main class where I have made instance object of my Logic\_Output class to execute my program. As you can see here!



Result of my program is here!