Lab 6: Permutation using Recursion

```
Location: CourseWeb -> Labs/Recitations -> Lab 6: Tower of Hanoi
Download the following files:
  1. TpwersOfHanoi.java
  2. TowersTester.java
  3. THFrame.java
  4. THComponent.java
All files shown above including this instruction can be found in lab06_toh.zip.
```

Introduction

The **Tower of Hanoi**[1], is a mathematical game or puzzle. It consists of three towers, and a number of discs of different sizes which can be slide onto any rod. The puzzle starts with the discs in a neat stack in ascending order of size on one rod, the smallest at the top, the making a conical shape.

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disc can be moved at a time.
2. Each move consists of taking the upper disc from one of the stacks and placing it on top of another stack (i.e., a disc can only be moved if it is the uppermost disc on a stack).
3. No disc may be placed on top of a smaller disc.

With three discs, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of discs.

What to do

For this lab, imagine that we are going to use computer to solve Tower of Hanoi puzzle. To do so, we need a way to represent the Tower of Hanoi puzzle in computer. In the real world, the Tower of Hanoi puzzle is **an object consists of three towers (rod), and a number of discs where each disc can be uniquely identified by its color and size**. Since we are using Java which is an Object-Oriented Programming, why don't we construct an object that can be used to represent the actual Tower of Hanoi puzzle.

To do so, we need a class. Let's call this `TowerOfHanoi`. The starter file for this class can be found in `TowerOfHanoi.java`. An instance of this class can be constructed by specifying the number of discs (n). After it is constructed, it consist of the following:

- three towers where each tower can be uniquely identify by an integer number 0, 1, or 2, and
- n discs where each disc can be uniquely identify by an integer number between 0 to $n - 1$. **The disc number** 0 **is always the smallest disc.** Note that all discs should be on tower 0 and no disc on tower 1 and tower 2.

Operations that we can perform on this Table of Hanoi puzzle are as follows:

- `public int getNumberOfDiscs()`: We can check the number of discs in the Tower of Hanoi puzzle (total number of discs).

[1] From Wikipedia

- `public int getNumberOfDiscs(int tower)`: We can check the number of discs in each tower.
- `public int[] getArrayOfDiscs(int tower)`: We can check the **order** of discs in each tower. This results in an array of integer where the size of the array is the number of discs in the tower and the content of the array are disc ID numbers. The first element in the array should be the disc ID number of the disc that is located at the bottom of the tower. Thus, the last element in the array should be the disc ID number of the disc that is located at the top of the tower.
- `public void moveTopDisc(int fromTower, int toTower)`: We can move the top disc from one tower (`fromTower`) to another tower (`toTower`). This operation should return `true` if all of these conditions are met:
  1. `fromTower` contains at least one disc, and
  2. the `toTower` contains no disc or the top disc of `toTower` is larger than the top disc of `fromTower` unless the `fromTower` and `toTower` are the same tower.

  Otherwise, this operation should return false and no disc should be moved.

Example

Suppose a user construct the Tower of Hanoi using the following statement:

```
TowerOfHanoi toh = new TowerOfHanoi(7);
```

After the above statement, `toh` can be represented by the Tower of Hanoi puzzle shown in Figure 1 (a). Return values of each method should be as follows:

Method	Return Value
`toh.getNumberOfdiscs()`	7
`toh.getNumberOfdiscs(0)`	7
`toh.getArrayOfdiscs(0)`	`[6,5,4,3,2,1,0]`
`toh.getNumberOfdiscs(1)`	0
`toh.getArrayOfdiscs(1)`	`[]`
`toh.getNumberOfdiscs(2)`	0
`toh.getArrayOfdiscs(2)`	`[]`

If a user wants to move the top disc from tower 0 to tower 1, the following statement should be used (assuming that the variable named `returnValue` of type `boolean` has be declared earlier:

```
returnValue = toh.moveTopdisc(0,1);
```

In this case `returnValue` should be `true` and the Table of Hanoi puzzle should look like Figure 1 (b). Return values of each method should be as follows:

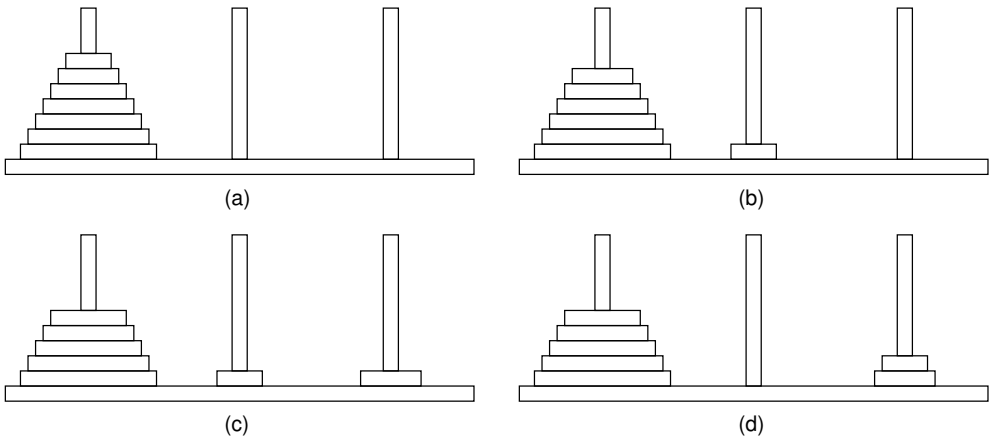

Figure 1: Tower of Hanoi Puzzle

Method	Return Value
`toh.getNumberOfdiscs()`	7
`toh.getNumberOfdiscs(0)`	6
`toh.getArrayOfdiscs(0)`	`[6,5,4,3,2,1]`
`toh.getNumberOfdiscs(1)`	1
`toh.getArrayOfdiscs(1)`	`[0]`
`toh.getNumberOfdiscs(2)`	0
`toh.getArrayOfdiscs(2)`	`[]`

If a user wants to move the top disc from tower 0 to tower 2, the following statement should be used (assuming that the variable named `returnValue` of type `boolean` has be declared earlier:

```
returnValue = toh.moveTopdisc(0,2);
```

In this case `returnValue` should be `true` and the Table of Hanoi puzzle should look like Figure 1 (c). Return values of each method should be as follows:

Method	Return Value
`toh.getNumberOfdiscs()`	7
`toh.getNumberOfdiscs(0)`	6
`toh.getArrayOfdiscs(0)`	`[6,5,4,3,2]`
`toh.getNumberOfdiscs(1)`	1
`toh.getArrayOfdiscs(1)`	`[0]`
`toh.getNumberOfdiscs(2)`	1
`toh.getArrayOfdiscs(2)`	`[1]`

If a user wants to move the top disc from tower 1 to tower 2, the following statement should be used (assuming that the variable named `returnValue` of type `boolean` has be declared earlier:

```
returnValue = toh.moveTopdisc(1,2);
```

In this case `returnValue` should be `true` and the Table of Hanoi puzzle should look like Figure 1 (d). Return values of each method should be as follows:

Method	Return Value
`toh.getNumberOfdiscs()`	7
`toh.getNumberOfdiscs(0)`	6
`toh.getArrayOfdiscs(0)`	`[6,5,4,3,2]`
`toh.getNumberOfdiscs(1)`	0
`toh.getArrayOfdiscs(1)`	`[]`
`toh.getNumberOfdiscs(2)`	2
`toh.getArrayOfdiscs(2)`	`[1,0]`

Test Class

A test class (`TowersTester.java`) is provided. Simply compile and run this test class. This program will test your `TowerOfHanoi.java` and show your total points (out of 10). If you do not get the full 10 points, you should keep trying to fix your code until you get 10 points. **Note** that this test class is not perfect. It cannot tell you why your program is incorrect. You may have to look at the source code of `TowersTester.java` and see why it says `FAIL` and trace your code.

If your `TowerOfHanoi.java` works correctly, you can run `THFrame.java` to see an animation of

Due Date and Submission

For the due date, please check the lab in the CourseWeb. Submit your `Permutation.java` to the CourseWeb under this lab by the due date. **No late submission will be accepted**.