Practical 04: Recursion - Part II

Due 15 Sep by 22:00 **Points** 100 **Available** after 25 Aug at 0:00

The Practical 4 - II has been granted one week extension, up to the 15th of September.

Introduction

Recursion in computer science is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem (as opposed to iteration). The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science.

This assignment will focus on using Recursion. You should use Java Programming to complete this practical.

Submission

Note: all class work must be submitted to your SVN repository. For this practice, you shall use:

```
https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-06/practical-04
```

Signature on your files

Note that all your coding files must contain on the top of it this information:

```
//================================
// Foundations of Computer Science
// Student: you name
// id: your id
// Semester:
// Year:
// Practical Number:
//=================================
```

Practical 04 - Part I

Problem 01 - Tower of Hanoi (TOH)

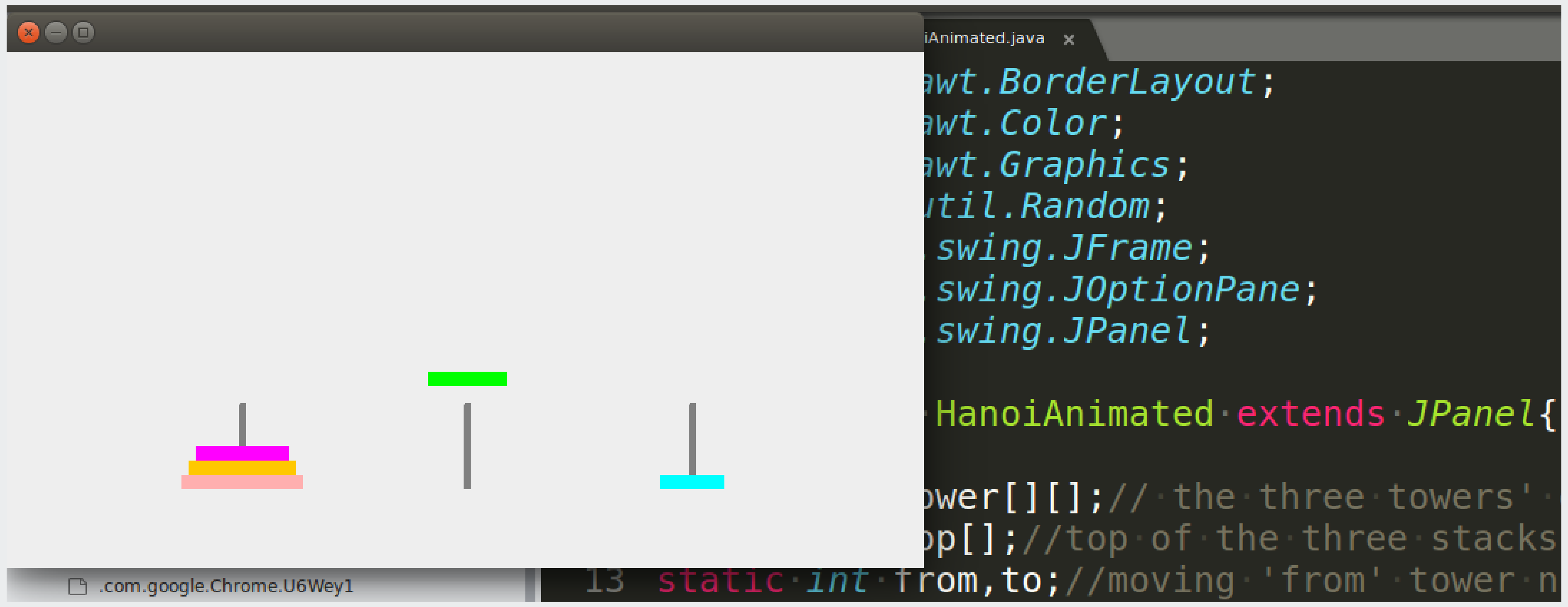
Tower of Hanoi is a mathematical puzzle. It consists of three poles and a number of disks of different sizes which can slide onto any poles. The puzzle starts with the disk in a neat stack in ascending order of size in one pole, the smallest at the top thus making a conical shape. The objective of the puzzle is to move all the disks from one pole (say 'source pole') to another pole (say 'destination pole') with the help of the third pole (say auxiliary pole). [Find more here] (https://www.geeksforgeeks.org/iterative-tower-of-hanoi/)

In this problem, you are required to solve the Towers of Hanoi (TOH) (https://www.geeksforgeeks.org/iterative-tower-of-hanoi/) using **Recursion**. In addition, try to write an *iterative* algorithm for TOH (using loops).

```
Signature:

public class HanoiTower

constructor: HanoiTower(): initialize 3 disks
constructor: HanoiTower(dTmp): initialize dTmp disks

public void solve(): this function should print every step from your tower of Hanoi resolution;

For instance:
Move the disk 1 from Pole1 to Pole2
Move the disk 2 from Pole1 to Pole3
Move the disk 1 from Pole2 to Pole3
...
Move the disk 1 from Pole1 to Pole3


Repository

save this project as:
- week-06/practical-04/problem-01/HanoiTower.java

Important:

1. The class HanoiTower MUST NOT contain a public static main function.
2. In order to test your code, you can create another class Test.java and perform your test from this class.
3. In this problem, you are free to define the structure of the problem. For instance, by your choice, you can
arrays, variables, or other types to represent the poles, disks etc... It is part of this practice the choices of the data
structure and they will not directly influence the marks. However, you are required to provide a clear explanation of
your structure choice in a file: explanation.txt.

Challenge (optional):

1. Use graphical interface to show progress on Tower of Hanoi resolution
```

Challenge hint:

- You can find out more on the packages **javax.swing** and **java.awt**



```
awt.BorderLayout;
awt.Color;
awt.Graphics;
util.Random;
.swing.JFrame;
.swing.JOptionPane;
.swing.JPanel;

HanoiAnimated extends JPanel{

ower[][];// the three towers'
op[];//top of the three stacks
static int from,to;//moving 'from' tower n
```

Problem 02 - Shake Hands

Consider that last Friday you were at a party with several people. In this party of **N** people, each person shook her/his hand with each other person only once. In this problem, you are asked to develop a **recursive** method that calculates how many hand-shakes happened at this party;

```
Signature:

public class ShakingParty

attributes:
private nPeople (int): number of people in the party

Constructor: ShakingParty(): randomly generate the number of people in the party
Constructor: ShakingParty(nTmp): add nTmp to the number of people in the party


public int countHandShakes(): this function should recursively count the number of hand shakes;


Repository

save this project as:
- week-06/practical-04/problem-02/ShakingParty.java

Important

1. The class ShakingParty MUST NOT contain a public static main function.
2. In order to test your code, you can create another class Test.java and perform your test from this class.
```

Problem 03 - Shake Hands with constraints

Consider that last Saturday you were at a similar party to Friday. In this party of **N** couples, only one gender (either male or female) can shake handa with other guests. Please note, the guests will interact only with the guests selected to shake hands (`N` people shaking hands). In this problem, you are asked to develop a **recursive** method that calculates how many hand-shakes happened at this party;

```
Signature:

public class ShakingPartyConstrainted

attributes:
private nCouples (int): number of couples in the party

Constructor: ShakingPartyConstrainted(): randomly generate the number of couples in the party
Constructor: ShakingPartyConstrainted(nTmp): add nTmp to the number of couples in the party


public int countHandShakes(): this function should recursively count the number of handshakes;


Repository

save this project as:
- week-06/practical-04/problem-02/ShakingPartyConstrainted.java

Important

1. The class ShakingPartyConstrainted MUST NOT contain a public static main function.
2. In order to test your code, you can create another class Test.java and perform your test from this class.
3. If possible, you should use inheritance from the ShakingParty
```