Practical 2 - Recursion

Updated: 7th December, 2021

Aims

- To practice recursion.
- To add validation and exception handling.
- To be able to trace through a recursive algorithm.
- To implement Towers of Hanoi.

Before the Practical

- Read this practical sheet fully before starting.

Activities

1. Factorial and Fibonacci

Given the code provided in the lecture slides, implement the iterative and recursive functions for Factorial and Fibonnaci.

Investigate their performance by running them with increasing values for n. How high can you go?

2. Greatest Common Denominator

Search online (or make your own) recursive algorithm for finding the Greatest Common Denominator. Implement it in Java or Python.

Remember to cite your resources, you can leave a comment in your code.

3. Number Conversions

Search online (or make your own) recursive algorithm for converting a Decimal (Base 10) to any Base (2-16). Implement it in Java or Python.

Remember to cite your resources, you can leave a comment in your code.

4. Wrappers and Exceptions

Update your code from Activities 1-3 to use Wrappers and Exceptions to validate and respond to bad input data. An example from the lecture slides is given below for Factorial.

```
// Wrapper
public long calcNFactorial ( int n )
{
if ( n < 0 )
{
        throw new IllegalArgumentException("Import␣must␣not␣be␣negative");
}
... // Add more exceptions as required.
else
{
        return calcNFactorialRecursive( n );
}
}
// Method
private long calcNFactorialRecursive( int n )
{
long factorial = 1;
if ( n == 0 )
{
        factorial = 1;
}
else
{
        factorial = n * calcNFactorialRecursive( n - 1 );
}
return factorial
}
```

```
# Wrapper
def calcNFactorial( n ):
        if n < 0:
                raise Exception("Import␣must␣not␣be␣negative")
        ... # Add more exceptions as required.
        else:
                return _calcNFactorialRecursive(n)
# Method
def _calcNFactorialRecursive( n ):
        if ( n == 0 ):
                factorial = 1
        else:
                factorial = n * calcNFactorialRecursive( n - 1 )
        return factorial
```

5. Towers of Hanoi

Implement the Towers of Hanoi algorithm as a Java or Python method.

- Write a main method to test this, ensuring you can input the number of disks at a minimum.
- You will also need to implement the `moveDisk(src, dest)` method. This can be as simple as an output statement printing *"Moving disk from peg source to peg destination"* where source and destination are 1, 2 or 3.
- Add indenting to your output to indicate the level of recursion. Some sample output is given below.
  Hint: You can pass a string of spaces or a number to keep track of recursion level.

```
towers(3, 1, 3)
                Recursion Level=3
                Moving Disk 1 from Source 1 to Destination 3
                n=1, src=1, dest=3

        Recursion Level=2
        Moving Disk 2 from Source 1 to Destination 2
        n=2, src=1, dest=2

                Recursion Level=3
                Moving Disk 1 from Source 3 to Destination 2
                n=1, src=3, dest=2

Recursion Level=1
Moving Disk 3 from Source 1 to Destination 3
n=3, src=1, dest=3

... # There are 7 moves for this problem.
```

Submission Deliverable

- Your code is due 2 weeks from your current tutorial session.
  - You will demonstrate your work to your tutors during that session
  - If you have completed the practical earlier, you can demonstrate your work during the next session
- You must **submit** your code and any test data that you have been using **electronically via Blackboard** under the *Assessments* section before your demonstration.
  - Java students, please do not submit the *.class files

Marking Guide

Your submission will be marked as follows:

- [2] Factorial and Fibonacci are implemented correctly and you can discuss their performance.
- [2] Greatest Common Denominator is implemented correctly and you can discuss its performance.
- [2] Number Conversions is implemented correctly and you can discuss its performance.
- [2] Wrappers and Exceptions are added to all your programs.
- [2] Towers of Hanoi is implemented correctly.

End of Worksheet