**Further Object- Oriented Programming**

U08026: Week 7

Recursion

David Lightfoot

Department of Computing and Communication Technologies

# Recursion

# Agenda

## The object of this practical is for you to practise the technique of recursion.

# Exercise 1 – Calculating Exponents using Recursion

## Raising a number to a power, an exponent, such as *y* to the power 2 (*y* squared), is written with circumflex (^) in some notations and as \*\* in others. So y squared, *y2*, is written *y*^2 or y \*\*2. When we write *y^n* we mean *y* to the power *n*.

## Using the facts that y0 = 1 and yn = y × yn-1 , for n > 0, write a **recursive** method

## public static double power(double y, int n) {

## *// pre: n >= 0*

## *// post: the value returned is y^n*

## }

## What is the base case of a recursive method? What is the recursive case?

## Specifically, what are the base case(s) and recursive case(s) of the code from part (a)?

# Exercise 2 – Getting some practice code set up

## Download the supplied code files from the website and get them up and running as a project as usual. The main file is *RecursionPractice.java*.

## Add the code of the method from Exercise 1, and try it out on various values by inserting into the main method lines such as:

## System.out.println("2 to the power 5 is " + power(2,5));

# Exercise 3 – Chessboard squares

## Write a **recursive** method, which, given a positive integer *n*, calculates how many 1× 1 squares there are on a *n × n* chessboard :

## public static int chessBoardSquares(int n) {

## // pre: n >= 1

## // post: the result is the number of squares on an n × n chessboard

## }

## For example, *chessBoardSquares(5)* should return the value 25. Test your method by adding code like the following to your main method:

## for (int i = 1; i <= 8; i++)

## System.out.println("chessBoardSquares(i) for i = " + i

## + " is " + chessBoardSquares(i));

## [Hints:

## 1. Use recursion on *n*, and have your base case be *n* = 1.

## 2. For the recursive case, draw a *n × n* chessboard, and then identify where inside that you can see a *(n-1) × (n-1)* chessboard.

## 3. See also extra ‘hint’ here:

# Exercise 4 – Array Matching

## Write a **recursive** method that returns true precisely when the first n integers of two given arrays of integers are the same.

## public static boolean arrayMatch(int n, int[] a, int[] b) {

## // pre: 0 <= n <= a.length and 0 <= n <= b.length

## // post: true is returned precisely when the values

## // a[0] ... a[n-1] are exactly the same as

## // the values b[0] ... b[n-1]

## }

## Test your method with some code such as this in the main method:

## int[] a = {1, 2, 3, 4, 5, 6};

## int[] b = {1, 2, 4, 8, 16};

## for (int i = 1; i <= 5; i++)

## System.out.println("arrayMatch(i, a, b) for i = " + i

## + " is " + arrayMatch(i,a,b));

## [Hints:

## 1. Use recursion on *n*, and have your base case be *n* = 0.

## 2. For the recursive case, draw a diagram of the two arrays indicating where the first *n* elements are. Draw also a diagram of the two arrays indicating where the first *n-1* elements are. Ask yourself: how can the result from comparing the first n-1 items be used to compare the first n items? Look at the difference between the diagrams.]

# Exercise 5 – Triangle numbers

## Write a **recursive** method, which, given an integer *n*, calculates the sum of the integers from 1 to *n*:

## public static int triangle(int n) {

## // pre: n >= 1

## // post: the result is the sum 1 + 2 + ... + n

## }

## Hint: use recursion on *n*, and have your base case be *n* = 1.

## Test your method by some code such as the following in your main method:

## for (int i = 1; i <= 10; i++)

## System.out.println("Triangle number " + i

## + " is " + triangle(i));

# Exercise 6 – Array printing

## Write a **recursive** method, to print out (using a System.out.println statement) all the contents of an array from positions 0 to n-1:

## public static void printFirstN(String[] a, int n) {

## // pre: 0 <= n <= a.length

## // post: the values a[0] ... a[n-1] are printed on the screen

## }

## Test your method with some code such as this in the main method:

## String[] a = {"Red", "Yellow", "Pink", "Green", "Orange"};

## printFirstN(a, a.length);

# What to submit

## Submit your answers to exercises 2 to 6, both as copies of source code and results as displayed in *NetBeans* pane (copy and paste).