CMPINF0401 Recitation

TUESDAYS 11:00-12:50

MICHAEL BARTLETT

Overview

- Recursion
- Lab8
- Assignment 4

Recursion: an overview

- Recursion is the process of making a method call itself
- This provides a way to break larger problems down into simple subproblems.
- When writing a recursive method, you must ensure that you have:
  - A base case, otherwise known as a halting case, that is attainable
  - A recursive case in which the method calls itself

Recursion: an example

- Recursion can be difficult to wrap your head around, so we're going to explore this recursive method that adds a range of numbers together
  - (i.e. 5+4+3+2+1)
  - sumRecursionEx.java

```java
public static void main(String[] args) {
    int result = sum(5);
    System.out.println(result);
}

public static int sum(int k) {
    if (k > 0) {
        return k + sum(k - 1);
    } else {
        return 0;
    }
}
```

Recursive case

Base case

- First, the sum method is called; sum is a non-void method that returns an integer as well as taking an integer in as a parameter.
- We call sum(5), which takes us down to the sum method
  - In this situation, k = 5
  - If k > 0, we return k + sum(k-1)
  - In other words, we return 5 + sum(4)…

```java
public static void main(String[] args) {
    int result = sum(5);
    System.out.println(result);
}

public static int sum(int k) {
    if (k > 0) {
        return k + sum(k - 1);
    } else {
        return 0;
    }
}
```

- Since sum(4) was called in our last return statement, we go through sum again with k = 4.
- We call sum(4), which takes us back to the sum method
  - In this situation, k = 4
  - If k > 0, we return k + sum(k-1)
  - In other words, we return 4 + sum(3)…

```java
public static void main(String[] args) {
    int result = sum(5);
    System.out.println(result);
}

public static int sum(int k) {
    if (k > 0) {
        return k + sum(k - 1);
    } else {
        return 0;
    }
}
```

- Since sum(3) was called in our last return statement, we go through sum again with k = 3.
- We call sum(3)… this is getting repetitive
  - sum(3) will return 3 + sum(2)
  - sum(2) will return 2 + sum(1)
  - sum(1) will return 1 + sum(0)
- When sum(0) is called, k = 0… which isn't greater than 0!
  - We've hit our base case! Return 0!

```java
public static void main(String[] args) {
    int result = sum(5);
    System.out.println(result);
}

public static int sum(int k) {
    if (k > 0) {
        return k + sum(k - 1);
    } else {
        return 0;
    }
}
```

- We're not done yet!!!!!
- There is still the matter of all those other method calls to go…
  - sum(0) = 0
  - sum(1) = 1 + sum(0) = 1 + 0 = 1
  - sum(2) = 2 + sum(1) = 2 + 1 = 3
  - sum(3) = 3 + sum(2) = 3 + 3 = 6
  - sum(4) = 4 + sum(3) = 4 + 6 = 10
  - sum(5) = 5 + sum(4) = 5 + 10 = 15!
- Thus, result = 15, so 15 will be printed.

```java
public static void main(String[] args) {
    int result = sum(5);
    System.out.println(result);
}

public static int sum(int k) {
    if (k > 0) {
        return k + sum(k - 1);
    } else {
        return 0;
    }
}
```

```
PS C:\Users\lrojt\Documents> java Test
15
```

Recusion: Another Example

- RecursiveRemove.java

Why recursion?

- Despite it taking up more memory than iterative methods (i.e., ones that contain loops), recursion can reduce the amount of time that it takes to do certain problems
  - The most important of these is sorting; if you choose to move forward in your computer science career, you'll learn about sorting algorithms in future classes. The fastest ones use recursion!
- At its core, recursion is essentially dividing one large problem into several smaller subproblems until they are manageable; I'm sure you've done something similar on homework assignments!

Lab 8

- Due 3/28
  - https://canvas.pitt.edu/courses/127916/files/8050403?module_item_id=2735318

Lab 8

- Main Method:
  - Replace the while loop with a single call to the *factorial* method
- Factorial Method:
  - Need a base case (what number does a factorial always end on?)
  - If !baseCase:
    - Multiply *n on*to the current factorial
    - Print the current value
    - Return your recursive call (Think: What value has to change in this call? What does it change to?)

Assignment 4

- Due 4/4
  - https://canvas.pitt.edu/courses/127916/files/8690592?module_item_id=2882423

Assignment 4

- Main Method:
  - Get the name of the path from the user
    - Should be using the unzipped version of A4-FS.zip
  - Make a file object out of the name of the path
- listOfFiles Method:
  - Use dirPath.listFiles(); to make an array out of File objects
  - Use an enhanced for loop to:
    - Check if f is not a directory (the File object has a method for this)
      - If it isn't, print "File Path: " and then it's absolute path (another method of the File object) as shown in the sample picture
      - If it is, recursively call the method on that path