CS 234

Review - Methods

Methods (~functions)

- Why do we need them?
- Implementation
- Passing parameters
- Returning values
- Returning no values
- Method overloading
- Recursion

Why do we need them?

```
import java.util.Scanner;
public class Main
{
   public static void main(String args[])
   {
        Scanner in = new Scanner(System.in);
        int hours;

        // Read Hours
        do{
            System.out.printf("Enter a value between 0 and 23:");
            hours = in.nextInt();
        }while (hours < 0 || hours > 23);

        // Read minutes
        int minutes;
        do{
            System.out.printf("Enter a value between 0 and 59:");
            minutes = in.nextInt();
        }while (minutes < 0 || minutes > 59);
   }
}
```

```
Enter a value between 0 and 23:24
Enter a value between 0 and 23:12
Enter a value between 0 and 59:42
```

Why do we need them?

```
import java.util.Scanner;
public class Main
{
    public static int readIntBetween(int low, int high)
    {
        Scanner in = new Scanner(System.in);
        int input;

        // Read value
        do{
            System.out.printf("Enter a value between %d and %d:", low, high);
            input = in.nextInt();
        }while (input < low || input > high);
        return input;
    }

   public static void main(String args[])
   {
        int hours = readIntBetween(0,23);
        int minutes = readIntBetween(0, 59);
   }
}
```

```
Enter a value between 0 and 23:45
Enter a value between 0 and 23:12
Enter a value between 0 and 59:42
```

Implementation

- Describe what the method should do
- Determine the method inputs
- Determine the types of the parameter variables and the return value
- Write *pseudocode* for obtaining the desired result
- Implement the method body
- Test your method

Static methods

- In Java, a *static method* is **a method that belongs to a class rather than an instance of a class**. The method is accessible to every instance of a class, but methods defined in an instance are only able to be accessed by that object of a class

Souce: https://www.techopedia.com/definition/24034/static-method-java

```
public static int readIntBetween(int low, int high)
```

Returning values

```
public class Main
{
    public static double areaTriangle(double base, double
height)
    {
        double area = (base* height)/2;
        return area;
    }

   public static void main(String args[])
   {
       double base = 3.5;
       double height = 4.2;
       double area = areaTriangle(base, height);
       System.out.printf("The are of the triangle is %.3f",area);
   }
}
```

Returning no values

```
public class Main
{
    public static void drawTree(int rows)
    {
        for (int i=0; i<rows; i++)
        {
            for (int j=rows-i; j>1; j--)
            {
                System.out.print(" ");
            }
            for (int j=0; j<=i; j++ )
            {
                System.out.print("* ");
            }
        System.out.println();
        }
    }

    public static void main(String args[])
    {
        drawTree(6);
    }
}
```

Scope of variables

```
public class Main
{
    public static double addTax(double price, double rate)
    {
        double tax = price * rate / 100;
        price = price + tax;
        System.out.println("Price inside method: " + price);
        return tax;
    }

        public static void main(String[] args) {
                double price = 10.0;
                addTax(price, 7.5);
                System.out.println("Price outside method: " + price);
        }
}
```

What is the final value of price?

We'll see later that this is not always like that

Method overloading

```
public class Main
{

   static int addNumbers(int x, int y)
   {
     return x + y;
   }

   static double addNumbers(double x, double y)
   {
     return x + y;
   }

   static int addNumbers(int x, int y, int z)
   {
     return x + y + z;
   }

   public static void main(String[] args)
   {
     int num1 = addNumbers(4,5);
     double num2 = addNumbers(4.5, 7.4);
     int num3 = addNumbers(2,6,5);
     System.out.println("2 ints: " + num1);
     System.out.println("2 doubles: " + num2);
     System.out.println("3 ints: " + num3);
   }
}
```

```
2 ints: 9
2 doubles: 11.9
3 ints: 13
```

Recursion

Recursive method: A method that calls itself

```
function factorial (num){
  if(num===1) return 1
  return num * factorial(num-1)
}
```

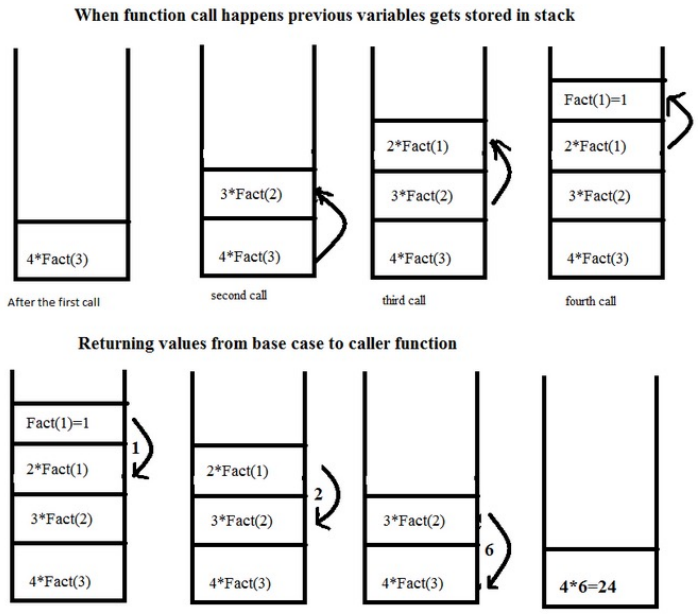
The base case / conditional:
When this condition is met the function returns, ending the recursive call. In this case, 1 is returned.

Modified input:
each time the factorial function is called it takes a modified parameter. (we don't want to keep passing in the same value).

The recursive call:
If the base case / conditional is <u>not</u> met, a recursive call is made. The function will return num multiplied by the value returned from the next function call.

Recursion

```
public static void main(args: Array<String>) {
    ... .. ...                    4
    result = factorial(number)
    ... .. ...
}
                                                return 4*6
static int factorial(int n) {
                                  4
   if (n != 0)
       return n* factorial(n-1)
    else      4                   3
       return 1
    }
}                                               returns 3*2
static int factorial(int n) {
                                  3
   if (n != 0)
       return n * factorial(n-1)
   else       3                   2
      return 1
}
                                                returns 2*1
static int factorial(int n) {
                                  2
   if (n != 0)
       return n * factorial(n-1)
   else       2                   1
      return 1
}
                                                returns 1*1
static int factorial(int n) {
                                  1
   if (n != 0)
       return n * factorial(n-1)
   else       1                   1
      return 1
}
static int factorial(int n) {                   returns 1
                                  0
   if (n != 0)
       return n * factorial(n-1)
   else
      return 1
}
```



Recursion

```java
/*****************************************************************************

Example of recursion

*****************************************************************************/
public class Main
{
    static int factorial(int n){
        if (n == 0) // Stoping condition
        {
            return 1;
        }
        else {
            return n * factorial(n - 1);
        }
    }

    public static void main(String[] args) {
        int number = 5;
        int result;
        result = factorial(number);
        System.out.println(number + "! = " + result);

    }
}
```

```
5! = 120
```

Recursion

```java
/*****************************************************************************

Example of recursion

Add the digits of a given number

12345 % 10 = 5

12345 / 10 = 1234

*****************************************************************************/
public class Main
{
    static int sum_digits(int n){
        if (n == 0){
            return 0;
        }
        else {
            return (n % 10 + sum_digits(n / 10));
        }
    }

    public static void main(String[] args) {
        int num = 12345;
        int result = sum_digits(num);
        System.out.println("Sum of digits in " + num + " is: " + result);
    }
}
```

```
Sum of digits in 12345 is: 15
```

Recursion

```java
/*****************************************************************************

Check if a string is a palindrom using a recursive method
For example: kayak, madam


*****************************************************************************/
public class Main
{
    public static boolean isPalindrome(String string) {
        // A string with just one character is a palindrome
        if (string.length() <= 1){
            return true;
        }
        // If my string has more than one character
        // we need call the function passing a smaller string
        else {
            return string.charAt(0) == string.charAt(string.length()-1) &&
                    isPalindrome(string.substring(1, string.length()-1));
        }

    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("kayak"));
        System.out.println(isPalindrome("hello"));
        System.out.println(isPalindrome("madam"));
    }
}
```

```
true
false
true
```