Methods/Functions

Objective:

This lab will give you practical implementation of predefined mathematical, characters and String methods.

Activity Outcomes:

On completion of this lab student will be able

- Modularize a program by writing own methods
- Call user-defined methods in program
- Pass different types of arguments
- Develop program using recursion

Instructor Note:

As a pre-lab activity, read Chapter 06 from the text book “Java How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.

1) Useful Concepts

It is usually a better approach to divide a large code in small methods. A method is a small piece of code used to perform a specific purpose. Methods are defined first then called whenever needed. A program may have as many methods as required. Similarly a method may be called as many times as required.

User-defined methods in Java are classified into two categories:

- **Value-returning methods**—methods that have a return data type. These methods return a value of a specific data type using the **`return`** statement.
- **Void methods**—methods that do not have a return data type. These methods do not use a **`return`** statement to return a value.

Defining a Method

A method definition consists of its method name, parameters, return value type, and body.

Syntax

```
modifier returnValueType methodName(list of parameters){
  // Method body;
}
```

Calling a Method

Calling a method executes the code in the method. Two ways to call a method.

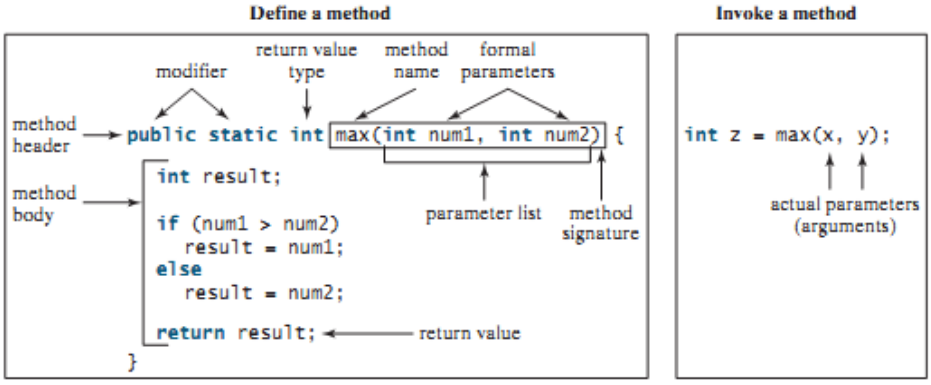
a) **Value Returning Method** should be called as follows

```
int larger = max(3, 4); or System.out.println(max(3, 4));
```

b) **Void methods** should be called as a statement

```
System.out.prinln("Welcome to Java");
```

Defining and Invoking a Method



Passing Arguments by Values

The arguments are passed by value to parameters when invoking a method. When calling a method, you need to provide arguments, which must be given in the same order as their respective parameters in the method signature. This is known as *parameter order association*.

When you invoke a method with an argument, the value of the argument is passed to the parameter. This is referred to as *pass-by-value*. If the argument is a variable rather than a literal value, the value of the variable is passed to the parameter. The variable is not affected, regardless of the changes made to the parameter inside the method

Overloading Methods

Overloading methods enables you to define the methods with the same name as long as their signatures are different. The Java compiler deter-mines which method to use based on the method signature.

Recursion

Recursion is a technique that leads to elegant solutions to problems that are difficult to program using simple loops.

All recursive methods have the following characteristics:

- The method is implemented using an **`if-else`** or a **`switch`** statement that leads to different cases.
- One or more base cases (the simplest case) are used to stop recursion.
- Every recursive call reduces the original problem, bringing it increasingly closer to a base case until it becomes that case.

In general, to solve a problem using recursion, you break it into subproblems. Each sub-problem is the same as the original problem but smaller in size. You can apply the same approach to each subproblem to solve it recursively.

2) Solved Lab Activites

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	***10 mins***	***Midum***	***CLO-5***
Activity 2	***10 mins***	***Midum***	***CLO-5***
Activity 3	***10 mins***	***Midum***	***CLO-5***
Activity 4	***10 mins***	***High***	***CLO-5***
Activity 5	***10 mins***	***High***	***CLO-5***
Activity 6	***10 mins***	***High***	***CLO-5***

Activity 1:

This activity demonstrate defining and calling a method. In this program a method `max()`, is defined which accepts two numbers as parameter(`int` type) and it returns a maximum value (`int` type).

Solution:

```
public class Activity1{
      /** Main method */
      public static void main(String[] args) {
            int i = 5;
            int j = 2;
            int k = max(i, j); //Method calling
            System.out.println("The maximum of " + i + " and " + j + "
            is " + k);
      }
      /** Return the max of two numbers – Method Definition*/
      public static int max(int num1, int num2) {
            int result;
            if (num1 > num2)
                  result = num1;
            else
                  result = num2;
            return result; // returning the value
      }
}
```

Output

```
The maximum of 5 and 2 is 5
```

Activity 2:

This activity illustrate concept of void method. Following is a program that defines a method named `printGrade` and invokes it to print the grade for a given score.

Solution:

```
public class Activity2{
      public static void main(String[] args){
            System.out.print("The grade is ");
            printGrade(78.5);
            System.out.print("The grade is ");
            printGrade(59.5);
      }
      public static void printGrade(double score){
            if(score >= 90.0){
                  System.out.println('A');
            }
```

```
            else if(score >= 80.0){
                  System.out.println('B');
            }
            else if (score >= 70.0){
                  System.out.println('C');
            }
            else if (score >= 60.0){
                  System.out.println('D');
            }
            else{
                  System.out.println('F');
            }
      }
}
```

Output

```
The grade is C
The grade is F
```

Activity 3:

This activity demonstrates the effect of passing by value. Following program creates a method for swapping two variables. The swap method is invoked by passing two arguments.

Solution:

```
public class Activity3 {
      /** Main method */
      public static void main(String[] args){
            // Declare and initialize variables
            int num1 = 1;
            int num2 = 2;
            System.out.println("Before invoking the swap method, num1
            is " +num1 + " and num2 is " + num2);
            // Invoke the swap method to attempt to swap two variables
            swap(num1, num2);
            System.out.println("After invoking the swap method, num1
            is" + num1 + " and num2 is " + num2);
      }
            /** Swap two variables */
      public static void swap(int n1, int n2) {
            System.out.println("\tInside the swap method");
            System.out.println("\t\tBefore swapping, n1 is " + n +
            " and n2 is " + n2);
                  // Swap n1 with n2
            int temp = n1;
            n1 = n2;
```

```
            n2 = temp;
            System.out.println("\t\tAfter swapping, n1 is " + n1+ "
            and n2 is " + n2);
          }
  }
```

Output

```
Before invoking the swap method, num1 is 1 and num2 is 2
  Inside the swap method
    Before swapping, n1 is 1 and n2 is 2
    After swapping, n1 is 2 and n2 is 1
After invoking the swap method, num1 is 1 and num2 is 2
```

Activity 4:

This activity demonstrate the concept of method overloading. In the following program three methods are created. The first finds the maximum integer, the second finds the maximum double, and the third finds the maximum among three double values. All three methods are named **`max`** **.**

Solution:

```
public class Activity4 {
      /** Main method */
      public static void main(String[] args) {
         // Invoke the max method with int parameters
         System.out.println("The maximum of 3 and 4 is " + max(3, 4));
        // Invoke the max method with the double parameters
         System.out.println("The maximum of 3.0 and 5.4 is "+
         max(3.0,5.4));
         // Invoke the max method with three double parameters
        System.out.println("The maximum of 3.0, 5.4, and 10.14 is "
        + max(3.0, 5.4, 10.14));
       }
      /** Return the max of two int values */
      public static int max(int num1, int num2){
            if (num1 > num2)
                  return num1;
            else
            return num2;
      }
      /** Find the max of two double values */
      public static double max(double num1, double num2){
            if (num1 > num2)
                  return num1;
            else
                  return num2;
      }
```

```
	/** Return the max of three double values */
	public static double max(double num1, double num2, double
num3){
		return max(max(num1, num2), num3);
	}
}
```

Output

```
The maximum of 3 and 4 is 4
The maximum of 3.0 and 5.4 is 5.4
The maximum of 3.0, 5.4, and 10.14 is 10.14
```

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

a) *Write a method that computes the sum of the digits in an integer. Use the following method header: public static int sumDigits(long n). For example, sumDigits(234) returns 9 (2 + 3 + 4).*

b) *Write a method with the following header to display an integer in reverse order:*

`public static void reverse(int number)`

For example, **`reverse(3456)`** *displays* **`6543`**. *Write a test program that prompts the user to enter an integer and displays its reversal.*

Lab Task 2

Write the methods with the following headers

// Return the reversal of an integer, i.e., reverse(456) returns 654

`public static int reverse(int number)`

// Return true if number is a palindrome

`public static boolean isPalindrome(int number)`

Use the reverse method to implement ***isPalindrome****. A number is a palindrome if its reversal is the same as itself. Write a test program that prompts the user to enter an integer and reports whether the integer is a palindrome.*

Lab Task 3

Write a method with the following header to display three numbers in increasing order:

`public static void displaySortedNumbers(double num1, double num2, double num3)`

Write a test program that prompts the user to enter three numbers and invokes the method to display them in increasing order.

Lab Task 4

Write a method that returns the number of days in a year using the following header:

public static int numberOfDaysInAYear(int year)

Write a test program that displays the number of days in year from 2000 to 2020.

Lab Task 5

Write a method that counts the number of letters in a string using the following header:

public static int countLetters(String s)

Write a test program that prompts the user to enter a string and displays the number of letters in the string.

Lab Task 6

Write a function **capitalize(lower_case_word)** *that takes the lower case word and returns the word with the first letter capitalized. Eg.,* **System.out.println(capitalize("word"))** *should print the word* **Word***.Then, given a line of lowercase ASCII words (text separated by a single space), print it with the first letter of each word capitalized using the your own function* **capitalize()**.

Lab task 7

Write a method that displays an n-by-n matrix using the following header:

public static void printMatrix(int n)

Each element is 0 or 1, which is generated randomly. Write a test program that prompts the user to enter n and displays an n-by-n matrix. Here is a sample run:

```
Enter n: 3
0 1 0
0 0 0
1 1 1
```

Lab Task 8

Write a Java method to count all vowels in a string. Here is sample run

```
Enter a string: Welcome to Java
Number of  Vowels in the string: 6
```