OODP Workshop 7

Modularizing (or breaking down) of code into methods is used so that code becomes reusable, easier to read and also easier to debug.

1. Find definitions for the following terms as they relate to Java programming:

a.    Parameter:

A parameter is a variable defined within the method declaration that is used to pass values or data into the method when it is called. Parameters allow methods to accept input data, which can be used within the method's body for various operations. Parameters are specified within parentheses after the method name in the method declaration.

Example:

public void printGreeting(String name) { *// 'name' is a parameter*

System.out.println("Hello, " + name + "!");

}

b.   Method:

A method is a block of code that performs a specific task or operation. It is a reusable piece of code that can be called from different parts of a program. Methods are defined within a class and can accept input parameters and return a value. They promote code reusability, modularity, and organization.

Example:

public class Calculator {

public int add(int a, int b) { // 'add' is a method

return a + b;

}

}

c.    Return value:

A return value is the value that a method sends back to the caller after it has completed its execution. It represents the output or result of the method's operations. Methods can optionally return a value, and the return type is specified in the method declaration before the method name. If a method does not return a value, its return type is void.

Example:

public int multiply(int x, int y) {

int result = x \* y; // Perform some operation

return result; // 'result' is the return value

}

d.   Method signature

The method signature is the combination of the method's name and the number, types, and order of its parameters. It is used to uniquely identify a method within a class. The return type is not part of the method signature.

Example:

public int calculateArea(int length, int width) { // 'calculateArea(int, int)' is the method signature

return length \* width;

}

public int calculateArea(double radius) { // 'calculateArea(double)' is a different method signature

return (int) (Math.PI \* radius \* radius);

}

2. Consider the following method:

public static int addValues(int x, int y){ System.out.println(x + y);

return (x + y);

}

1. What parameters does the method have?

Ans: The method addValues has two parameters: int x and int y.

1. What are appropriate values for the arguments when this method is called?

Ans: Any integer values can be used as arguments when calling the addValues method. For example, you could call it with addValues(2, 3), addValues(10, -5), or addValues(0, 0).

1. What is return value?

Ans: The return value of the addValues method is the sum of the two integer parameters x and y. It returns an int value, which is the result of x + y.

1. Provide one example of how you might call this method.

Ans: int result = addValues(4, 6);

System.out.println("The result is: " + result);

program will print "The result is: 10"

Or

System.out.println("The sum is: " + addValues(2, 5))

Putput will be : 7

The sum is: 7

3. Consider the following method:

public static void addValues(int x, int y){   
System.out.println(x + y);

}

1. What parameters does the method have?

Ans: The method addValues has two parameters: int x and int y.

1. What are appropriate values for the arguments when this method is called?

Ans: Any integer values can be used as arguments when calling the addValues method. For example, you could call it with addValues(2, 3), addValues(10, -5), or addValues(0, 0).

1. What is return value?

Ans: The method addValues has a void return type, which means it does not return any value.

1. Provide one example of how you might call this method

Ans: Example of how you might call the addValues method:

addValues(10, 20);

This will print the sum 30 to the console due to the System.out.println(x + y) statement inside the method.

4. Practice modularization

**Objective of this activity:- converting given code to modularised code**

Open menu.java and do following tasks:

Create a method to display menu that will display the menu which is given in file and ask to enter their selection and return that selection to calling method. Call this method in main method where there is a need to print menu.

Ans:

package lecture4;

import java.util.Scanner;

public class Menu {

public static void main(String[] args) {

// Create a Scanner object for user input

Scanner in = new Scanner(System.in);

// Call the displayMenu method and get the user's selection

int menuSel = displayMenu(in);

// Process the user's selection using a switch statement

switch(menuSel) {

case 1:

System.out.println("Menu item 1");

break;

case 2:

System.out.println("Menu item 2");

break;

case 3:

System.out.println("Exiting....");

break;

default:

System.out.println("Invalid selection!");

break;

}

// Close the Scanner object

in.close();

}

/\*\*

\* Method to display the menu, prompt the user for a selection,

\* and return the user's choice.

\* @param scanner The Scanner object for user input

\* @return The user's selection as an integer

\*/

public static int displayMenu(Scanner scanner) {

// Display the menu

System.out.println("Menu");

System.out.println("1. Enter data");

System.out.println("2. Display data");

System.out.println("3. Exit");

System.out.print("Enter selection: ");

// Read the user's selection from the scanner

int selection = scanner.nextInt();

// Return the user's selection

return selection;

}

}

Move the switch case to a separate method which will take menuSel as input and print necessary details. In case 1, ask user to enter their name and age, in case 2, print the details of user and case 3, terminate the program.