# 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:

Ans:-

* 1. Parameter

A parameter is a variable defined in the method declaration that accepts the value passed to the method when it is called. Parameters act as placeholders for the values (arguments) that will be supplied to the method.

* 1. Method

A method is a block of reusable code that performs a specific task. Methods in Java have a name, a return type, and a set of parameters. They allow you to organize your code, improve readability, and promote code reuse.

* 1. Return value

The return value is the output that a method provides after executing its code. A method can return a value of a specific data type, or it can return nothing (void). The return value allows the method to pass information back to the code that called it.

* 1. Method signature

The method signature is the unique identifier for a method. It consists of the method's name, the number and types of its parameters, and its return type. The method signature is used to define and call the method, and it allows the Java compiler to distinguish between different methods with the same name but different parameters.

1. Consider the following method:

public static int addValues(int x, int y){

System.out.println(x + y);

return (x + y);

}

Ans:-

* 1. What parameters does the method have?

The method addValues() has two parameters: int x and int y. These are the variables that are passed into the method when it is called.

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

Appropriate values for the arguments would be any valid integers. For example, you could call the method with addValues(5, 10) or addValues(-3, 7).

* 1. What is return value?

The return value of the addValues() method is the sum of the two input values, x and y. The method explicitly returns the result of x + y using the return statement.

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

One way to call the addValues() method could be:

int result = addValues(2, 3);

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

output:

The result is: 5

1. Consider the following method:

public static void addValues(int x, int y){

System.out.println(x + y);

}

Ans:-

* 1. What parameters does the method have?

The method addValues() has two parameters: int x and int y. These are the variables that are passed into the method when it is called.

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

Appropriate values for the arguments would be any valid integers. For example, you could call the method with addValues(5, 10) or addValues(-3, 7).

* 1. What is return value?

The method addValues() does not have a return value. It is declared as void, which means it does not return anything. Instead, it simply prints the sum of the two input values to the console.

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

We can call the addValues() method :

addValues(2, 3);

This would output: 5

1. 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.
* 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.

Ans:-

package lecture4;

import java.util.Scanner;

public class Menu {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int menuSel;

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

menuSel = displayMenu(in);

// Call the processSelection method to handle the user's choice

processSelection(menuSel, in);

in.close();

}

// Method to display the menu and return the user's selection

public static int displayMenu(Scanner in) {

System.out.println("Menu");

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

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

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

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

return in.nextInt();

}

// Method to process the user's selection

public static void processSelection(int menuSel, Scanner in) {

String name = "";

int age = 0;

switch(menuSel){

case 1:

// Ask user to enter their name and age

System.out.println("Enter your name:");

name = in.next(); // read the name

System.out.println("Enter your age:");

age = in.nextInt(); // read the age

System.out.println("Data entered successfully.");

break;

case 2:

// Display the details of the user

System.out.println("Displaying user details:");

System.out.println("Name: " + name);

System.out.println("Age: " + age);

break;

case 3:

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

System.exit(0);

break;

default:

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

}

}

}