JOBSHEET 7
LOOPING 1

1. Objective

After finishing this topic, students must be able to:

- Explain the format of loop programming part 1
- Implement a loop part 1 flowchart using the Java programming language

2. Laboratory

2.1 Experiment 1: Counting Multiples Using FOR

Experiment Time: 60 minutes

In this experiment, the code is created to display multiples of a specific number within the range of 1 to 50 using a FOR loop, and to calculate the total of these numbers.

1. Open text editor. Create a new Java File named **ForMultiplesStudentIDNumber.java**
2. Create the basic structure of Java program containing **class** declaration and **main()** method
3. Add the **Scanner** library.
4. Create or declare variable named **input** from **Scanner** library.
5. Create **int** variables named **multiple**, **sum**, and **counter**. Initialize variable **sum** and **counter** with 0.
6. Add the following code to get the user input!

```
System.out.print(s:"Input the multiple = ");
multiple = input.nextInt();
```

7. Create the FOR loop with IF condition to evaluate the multiples number

```
for(int i=1;i<=50;i++){
    if(i%multiple == 0){
        sum = sum + i;
        counter++;
        //System.out.print(i+"-");
    }
}
```

8. Display the sum and counter of multiples number in range from 1 to 50.

```
System.out.printf(format:"There are %d numbers that are multiple of %d in range 1 to 50.\n", counter, multiple);
System.out.printf(format:"The sum from all multiples of %d in range 1 s.d. 50 is %d. \n", multiple, sum);
```

9. Run the program and analyze the result. Your result must be like this:

```
Input the multiple = 5
There are 10 numbers that are multiple of 5 in range 1 to 50.
The sum from all multiples of 5 in range 1 s.d. 50 is 275.
```

10. Commit and push the changes to GitHub.

Questions

1. There are 3 main components in FOR loop. Based on experiment 1 above, identify and explain these 3 components!
2. Explain how the following code works!

```
for(int i=1;i<=50;i++){
    if(i%multiple == 0){
        sum = sum + i;
        counter++;
        //System.out.print(i+"-");
    }
}
```

3. Modify the existing code by adding a new variable to calculate the average of all the specified multiples! Push and commit the program code to GitHub.
4. Create a new Java program file named **WhileMultiplesStudentIDNumber.java**. Create the equivalent code by using while loop. Push and commit the code to GitHub.

2.2 Experiment 2: Calculating Employee Overtime Pay Using WHILE and CONTINUE

Experiment Times: 60 minutes

A company provides overtime pay to its employees every week. The overtime pay is calculated based on the employee's position and the number of overtime hours in a week. Employees with the position of 'director' do not receive any additional overtime pay even if they work overtime, employees with the position of 'manager' receive overtime pay of 100,000 per hour, while employees with the position of 'staff' receive overtime pay of 75,000 per hour. In this experiment, a program code is created using WHILE and CONTINUE to calculate the company's expenses.

1. Open the text editor and create a new Java file named **WhileOvertimePayStudentIDNumber.java**
2. Create the basic structure of Java program containing **class** declaration and **main**() method
3. Add the **Scanner** library.
4. Create or declare variable named **input** from **Scanner** library.
5. Declare **int** variable named **numEmployee** and **overtimeHours,** and then **overtimePay** and **totalOvertimePay** with **double** datatype. Initialize **overtimePay** and **totalOvertimePay** with 0
6. Declare variable **position** with **String** datatype.
7. Add the following code to get user input for **numEmployee** variable.

```
System.out.print(s:"Employee number = ");
numEmployee = input.nextInt();
```

8. Create a WHILE loop structure with an IF-ELSE conditional and CONTINUE to determine overtime pay based on employee positions

```
int i=0;
while(i<numEmployee){
    System.out.print("Position of employee "+(i+1)+" (director, manager, staff) = ");
    position = input.next();
    System.out.print("Employee "+(i+1)+" overtime hours = ");
    overtimeHours = input.nextInt();
    i++;

    if(position.equalsIgnoreCase(anotherString:"director")){
        continue;
    }else if(position.equalsIgnoreCase(anotherString:"manager")){
        overtimePay=overtimeHours*100000;
    }else if(position.equalsIgnoreCase(anotherString:"staff")){
        overtimePay=overtimeHours*75000;
    }

    totalOvertimePay += overtimePay;
}
```

9. Display the **totalOvertimePay**

```
System.out.println("Total of Overtime Pay = "+totalOvertimePay);
```

10. Run the program and analyze the result. Your result must be like this:

```
Employee number = 3
Position of employee 1 (director, manager, staff) = manager
Employee 1 overtime hours = 1
Position of employee 2 (director, manager, staff) = director
Employee 2 overtime hours = 10
Position of employee 3 (director, manager, staff) = staff
Employee 3 overtime hours = 5
Total of Overtime Pay = 475000.0
```

11. Commit and push the changes to GitHub.

Questions

1. Show the part of the program code used as a condition to stop the WHILE loop! How many times is the loop executed?
2. In this code,

```
if(position.equalsIgnoreCase(anotherString:"director")){
    continue;
```

What actually happens if the **'position'** variable contains the value 'DIRECTOR'? What is the use of CONTINUE within the loop structure?

3. Why is the 'i++' iteration component placed in the middle, not at the end of the while block? Move 'i++' to the end of the while block, then run the program again by entering 'DIRECTOR' as the **position** for the first employee. What happens? Explain!
4. Modify the program code to handle invalid positions like the following example:

```
Employee number = 3
Position of employee 1 (director, manager, staff) = director
Employee 1 overtime hours = 5
Position of employee 2 (director, manager, staff) = manager
Employee 2 overtime hours = 10
Position of employee 3 (director, manager, staff) = programmer
Employee 3 overtime hours = 4
Invalid position!
Position of employee 3 (director, manager, staff) = staff
Employee 3 overtime hours = 4
Total of Overtime Pay = 1300000.0
```

5. Commit and push the changes to GitHub.

2.3 Experiment 3: Calculating Leave Entitlement Using DO-WHILE

Experiment Times: 50 minutes

In this experiment, a program code is created using DO-WHILE to calculate the **leave entitlement** of an employee. Employees are entitled to 5 days of leave. Leave days will be

deducted each time they are used. When there are only 2 days of leave remaining, the employee receives a warning to stop using their leave

1. Open the text editor and create a new Java file named **DoWhileLeaveEntitlementStudentIDNumber.java**
2. Create the basic structure of Java program containing class declaration and main() method
3. Add the **Scanner** library.
4. Create or declare variable named **input** from **Scanner** library.
5. Create variables **leaveEntitlement** and **numLeave** with int datatype.
6. Create variable **confirmation** with **String** datatype.
7. Create a DO-WHILE loop structure to get the user input from the keyboard and calculate leave entitlement

```
int jatahCuti, jumlahHari;
String konfirmasi;

System.out.print("Jatah cuti: ");
jatahCuti = sc.nextInt();

do {
    System.out.print("Apakah Anda ingin mengambil cuti (y/t)? ");
    konfirmasi = sc.next();

    if (konfirmasi.equalsIgnoreCase("y")) {
        System.out.print("Jumlah hari: ");
        jumlahHari = sc.nextInt();

        if (jumlahHari <= jatahCuti) {
            jatahCuti -= jumlahHari;
            System.out.println("Sisa jatah cuti: " + jatahCuti);
        } else {
            System.out.println("Sisa jatah cuti Anda tidak mencukupi");
            break;
        }
    }
} while (jatahCuti > 0);
```

8. Run the program and analyze the result. It must be the same with the following output.

```
Number of Leave Entitlement = 12
Do you want to take a leave? (y/n) = y
Leave number = 4
The remaining leave entitlement = 8
Do you want to take a leave? (y/n) = y
Leave number = 5
The remaining leave entitlement = 3
Do you want to take a leave? (y/n) = y
Leave number = 4
The remaining leave entitlement is not sufficient!
```

9. Commit and push the changes to GitHub.

Questions

1. What is the use of the BREAK within the loop syntax?
2. Modify the program so that if the number of leave days requested is greater than the remaining entitlement, the program does not stop, allowing the user to enter the number of days according to the entitlement.
3. Commit and push the program code to GitHub.
4. When typing "t" as the confirmation input, what happens? Why?
5. Modify the program code so that when the user enters "t" as the confirmation input, the program will stop.
6. Commit and push the program code to GitHub.

3. Assignment

Experiment Times: 130 minutes

- Implement the flowchart that was already created in the assignment for the Basic Programming course related to the project into your program.
- Push and commit the result of your program code to your project repository.
- Note: the assignment should only apply the material from sessions 1 to 7