 **Name** : RIZQI DEWANTO

` **Nim** : 2341760151

**Kelas** : 1-G

**Number :** 22

**Materi** : LOOPING

# BASIC PROGRAMMING PRACTICUM REPORT

**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**

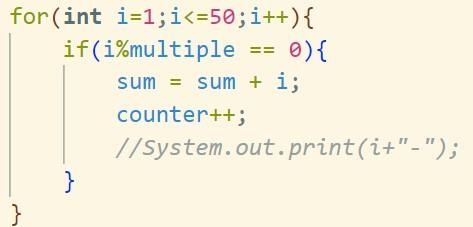
## Experiment Time: 60 minutes

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!



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



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



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

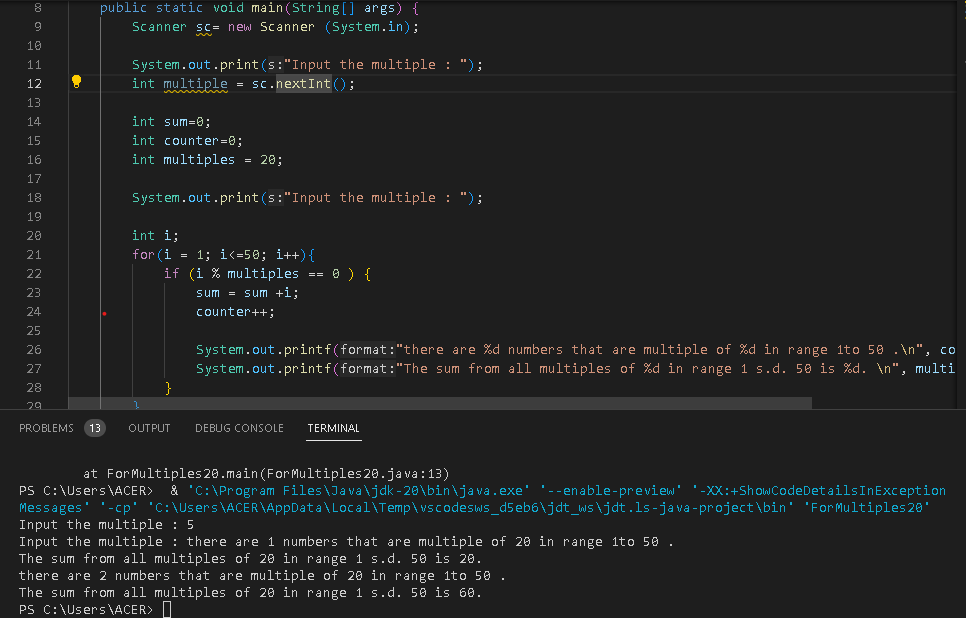


1. Commit and push the changes to GitHub.

**Questions**

1. There are 3 main components in FOR loop. Based on experiment 1 above, identify and explain these 3 components!

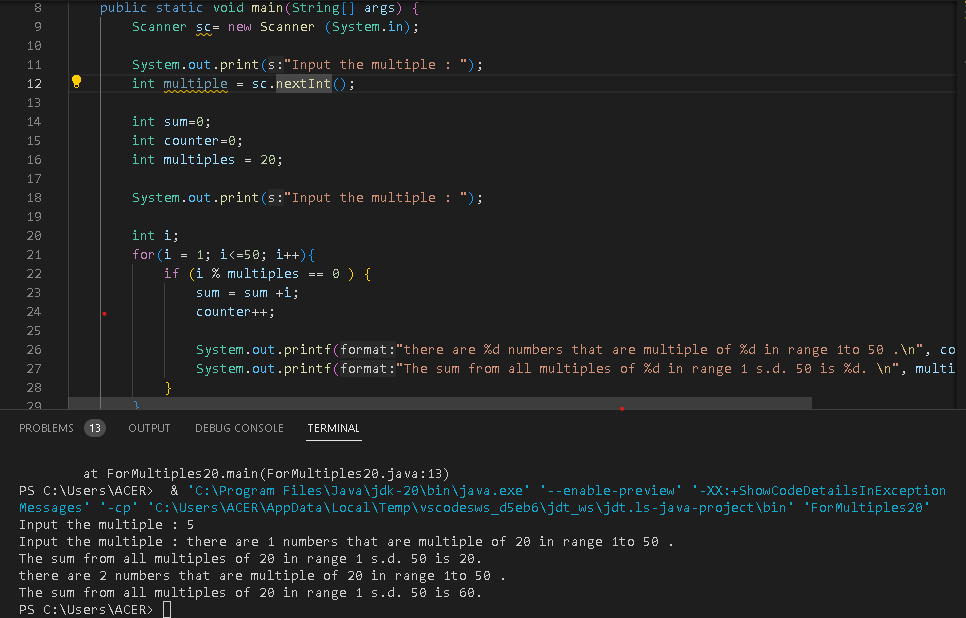
**There are Conmponent of the for and I its for mark/sign increment or to looping** 2. Explain how the following code works!



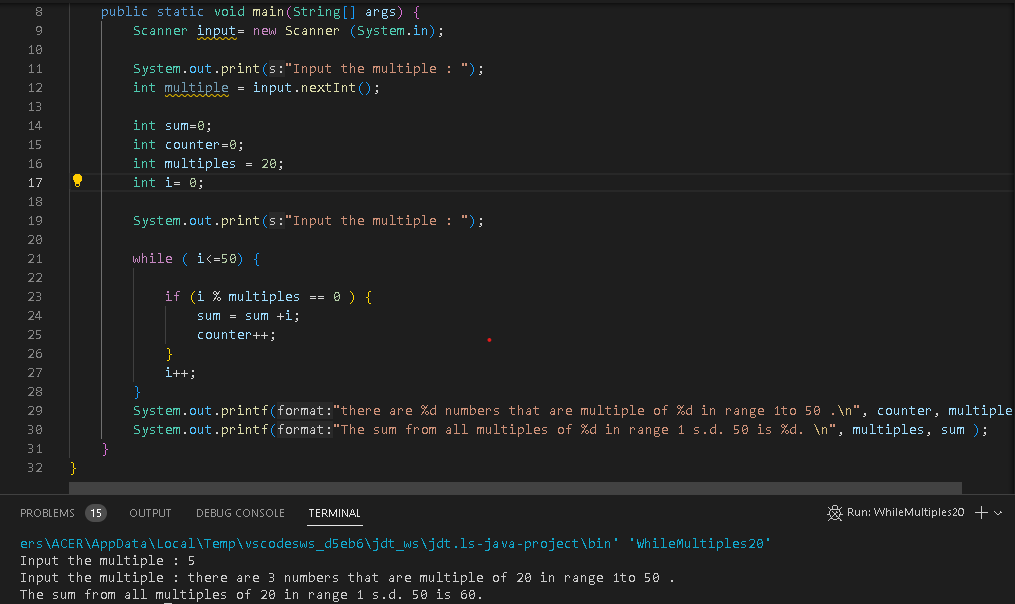
**The loop will start from 1 to 50 and in each loop the value of i is increased by 1 then each loop the value of i is entered into the if operation, if i is divisible by a multiple then I will be entered into the entered the sum operation += i which will add up all i that enter if. And**

**counter++ counts the number of times the if is operated on**

1. 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.



1. 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**

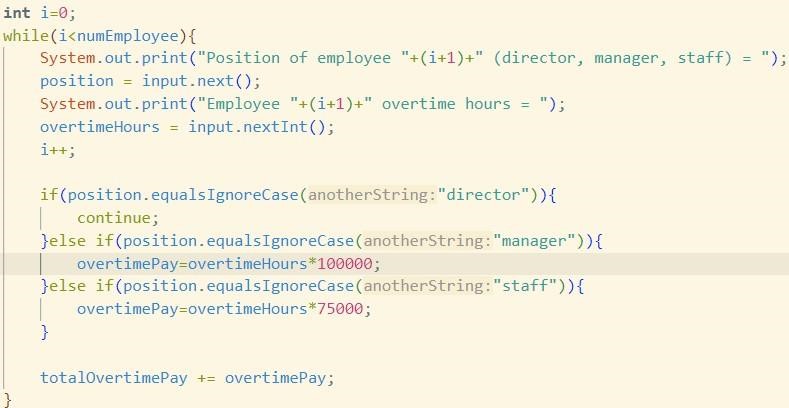
## CONTINUE Experiment Times: 60 minutes

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.



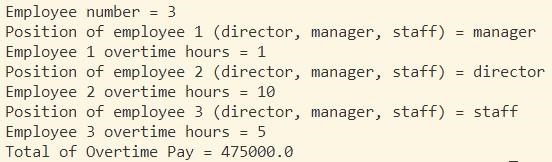
1. Create a WHILE loop structure with an IF-ELSE conditional and CONTINUE to determine overtime pay based on employee positions



1. Display the **totalOvertimePay**



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



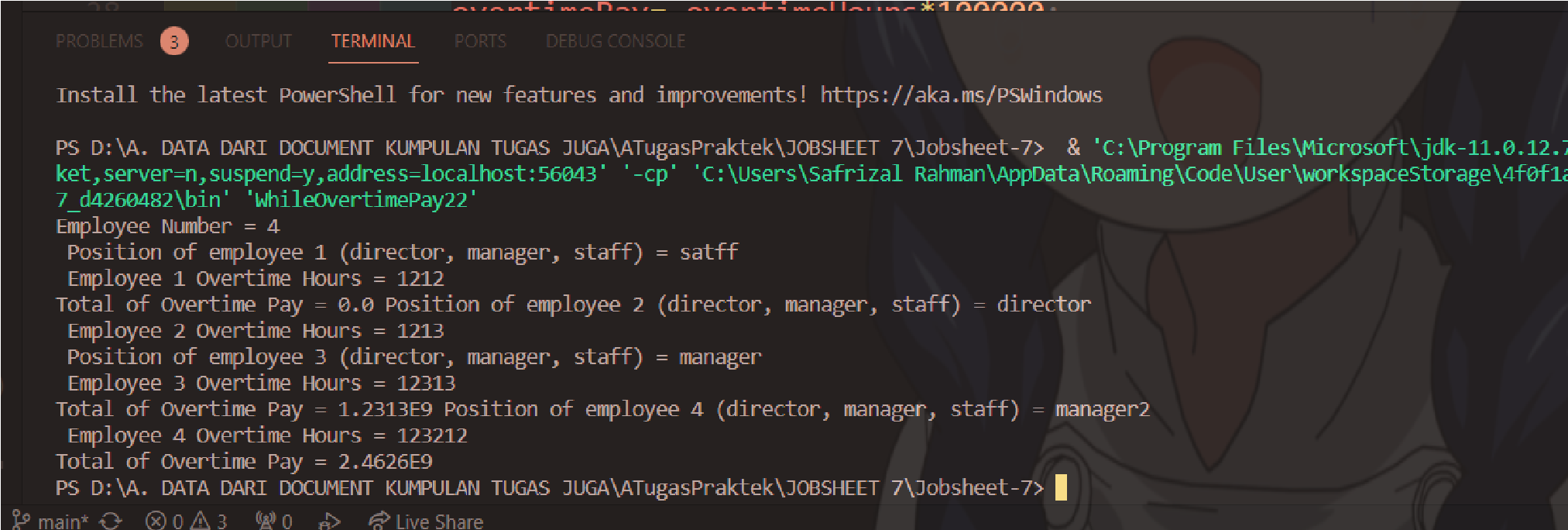
1. Commit and push the changes to GitHub

|  |
| --- |
| public class WhileOvertimePay22 {  public static void main(String[] args) {  Scanner input22 = new Scanner (System.in);  *//Declare* String position; int numEmployee; int overtimeHours; double overtimePay = 0; double totalOvertimePay = 0;    System.out.print("Employee Number = "); numEmployee = input22.nextInt();  int i=0;  while(i<numEmployee){  System.out.print(" Position of employee  "+(i+1)+" (director, manager, staff) = "); position = input22.next();  System.out.print(" Employee "+(i+1)+" Overtime Hours = "); overtimeHours = input22.nextInt(); i++;  if(position.equalsIgnoreCase("director")){ continue;  }else if (position.equalsIgnoreCase("manager"))  { overtimePay= overtimeHours\*100000;  }else if (position.equalsIgnoreCase("staff")){ overtimePay= overtimeHours\*75000;  } |

totalOvertimePay += overtimePay;

System.out.print("Total of Overtime Pay = "

+totalOvertimePay);

 }

}

}

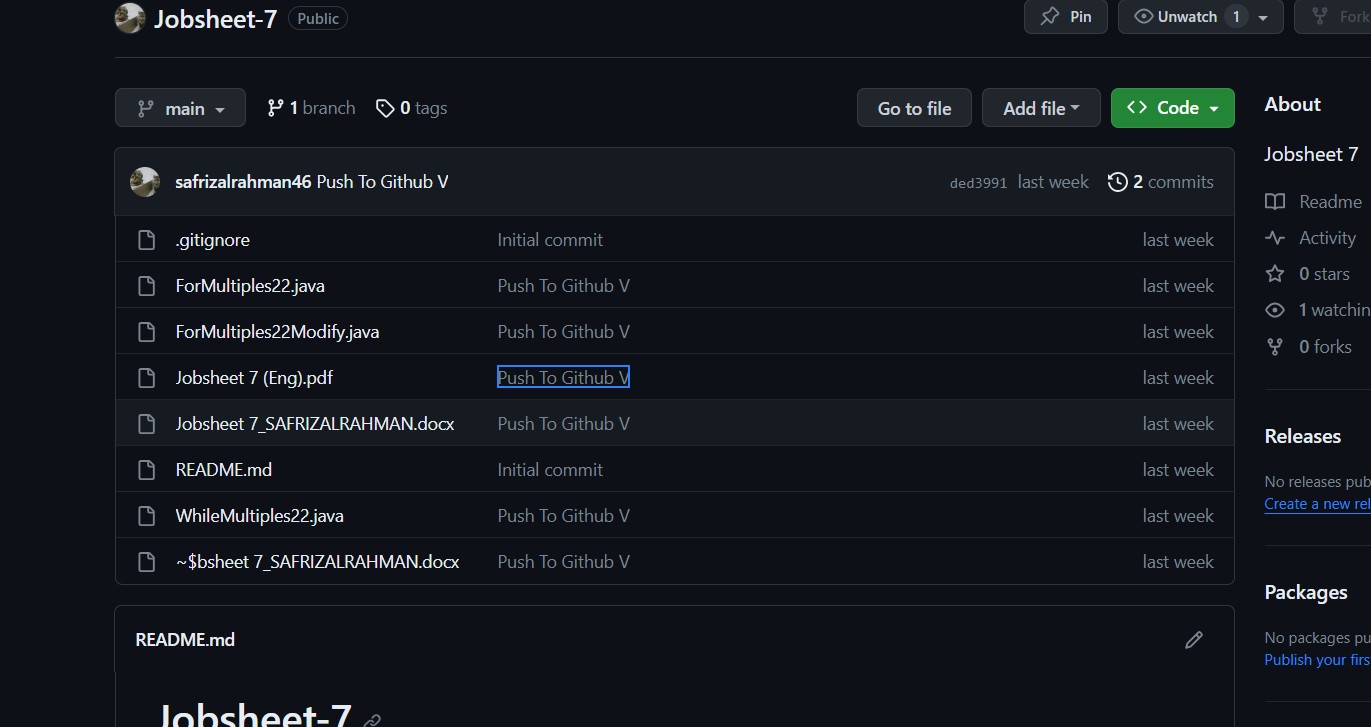
### Questions

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

**The number of times the loop is performed depends on this code**

1. System.out.print("Employee Number = "); 3. numEmployee = input22.nextInt();

4.

1. int i=0;
2. while(i<numEmployee){

2. In this code,



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

* **Because there is equalsIgnoreCase, as long as the string is written by the director, it is still**

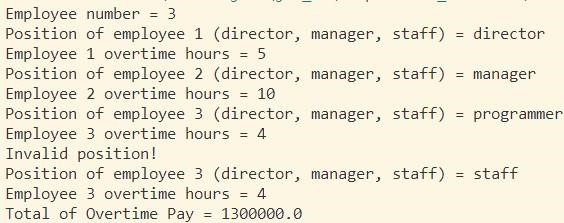
**considered the same string**

* **The role of continue so that the loop jumps to the next increment**

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!

**- If placed at the end when entering "director" continue will jump to i++ so that the loop loop will move to the next**

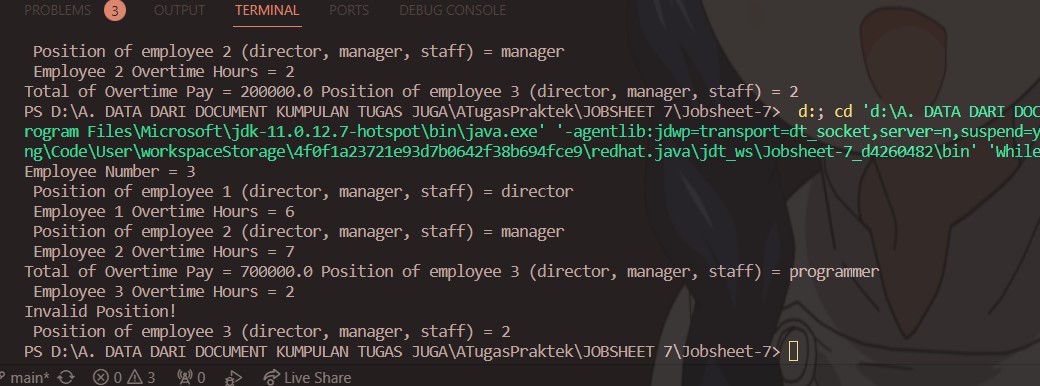
4.Modify the program code to handle invalid positions like the following example:



}else{

System.out.println("Invalid Position!"); i--; continue;

}



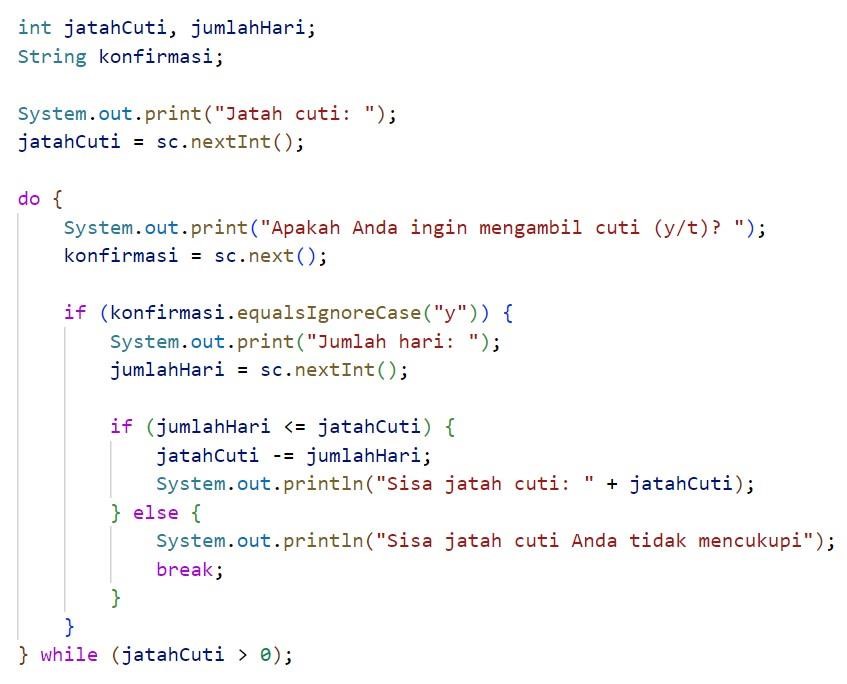
7. Commit and push the changes to GitHub.

**2.3 Experiment 3: Calculating Leave Entitlement Using DO-WHILE**

### Experiment Times: 50 minutes

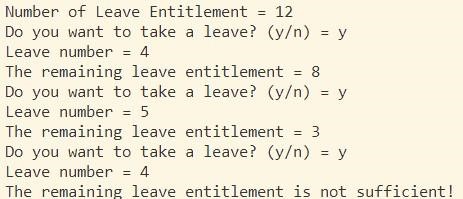
In this experiment, a program code is created using DO-WHILE to calculate the **leave** **entitlement** of an employee. Employees are entitled to 5 days of leave. Leave days will be deducted each time they are used. When there are only 2 days of leave remaining, the employee receives a warning to stop using their leave

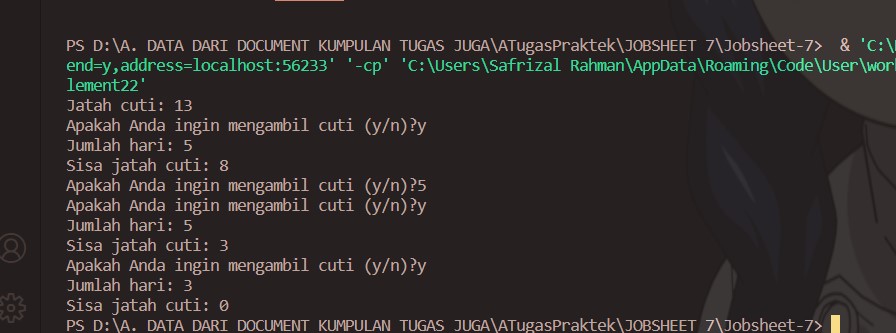
1. Open the text editor and create a new Java file named **DoWhileLeaveEntitlementStudentIDNumber.java**
2. Create the basic structure of Java program containing class declaration and main() method
3. Add the **Scanner** library.
4. Create or declare variable named **input** from **Scanner** library.
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



|  |
| --- |
| import java.util.Scanner;  public class DoWhileLeaveEntitlement22 { public static void main(String[] args) { Scanner sc = new Scanner(System.in); int jatahCuti, jumlahHari;  String konfirmasi;    System.out.print("Jatah cuti: "); jatahCuti = sc.nextInt();    do {  System.out.print("Apakah Anda ingin mengambil cuti (y/n)?"); 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);  }  } |

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





1. Commit and push the changes to GitHub.

### Questions

1. What is the use of the BREAK within the loop syntax?

**To stop the current loop**

1. 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.

**Remove break: in else condition**

1. Commit and push the program code to GitHub.

1. When typing "t" as the confirmation input, what happens? Why?

**Because there is no condition for choosing "t"**

1. Modify the program code so that when the user enters "t" as the confirmation input, the program will stop.

**Just add this but not nested, you know this is outside if**

1. else {
2. System.out.println("Sudah Cukupu Baiklah");
3. break;

9.

* 1. }

* 1. 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

### import java.util.Scanner;

public class CodinganKedua {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

|  |
| --- |
| *// Declaration Login* boolean signUp = false, login = false; int option;  String name="Mitra", nameLogin , pass="Mitra123", passLogin, pass1, again;  *// Declaration*  String expeditionName, startDate, endDate; String totalGoods; double totalDistance, rangeTime, totalTime, costPerTime, totalCost, costPerKilometer;  *// Login And SignUp*  System.out.print("\033[H\033[2J");  System.out.flush();  System.out.println(" ----------------");  System.out.println("| EXPEDITION |");  System.out.println(" ----------------"); System.out.println("1. SignUp First if you dont have account");  System.out.println("2. Login If you Have an account Before");  option = scanner.nextInt();  System.out.print("\033[H\033[2J"); System.out.flush(); switch (option) { case 1 : do{ } break; |

default:

|  |
| --- |
| break;  }    *// Input*  System.out.println("Welcome to the Expedition  Java Application!");  System.out.print("Please enter the name of the expedition: "); expeditionName = scanner.nextLine();  System.out.print("Please enter the start date  (DD-MM-YYYY): "); startDate = scanner.nextLine();  System.out.print("Please enter the end date (DDMM-YYYY): "); endDate = scanner.nextLine();  System.out.print("Please enter the total distance (in kilometers): "); totalDistance = scanner.nextDouble();    System.out.print("Please enter Range Time days :  "); rangeTime = scanner.nextDouble();    scanner.nextLine(); *// Consume newline*  System.out.print("Please enter the total number of Goods: "); totalGoods = scanner.nextLine(); |

System.out.print("Please enter the cost per

|  |
| --- |
| Goods (in currency): "); costPerTime = scanner.nextDouble();  *// Calculations* totalTime = totalDistance / rangeTime; totalCost = Double.parseDouble(totalGoods) \* costPerTime; costPerKilometer = totalCost / totalDistance;  *// Output*  System.out.println("\nExpedition Details:");  System.out.println("Name: " + expeditionName);  System.out.println("Start Date: " + startDate);  System.out.println("End Date: " + endDate); System.out.println("Total Distance (km): " + totalDistance);  System.out.println("Range Time (days): " + rangeTime);  System.out.println("Total Time (hours): " + totalTime);  System.out.println("Total Goods: " + totalGoods);  System.out.println("Cost per Goods: " + costPerTime);  System.out.println("Total Cost: " + totalCost); System.out.println("Cost per Kilometer: " + costPerKilometer);    *// Additional Information* if (totalDistance > 1000) {  System.out.println("This is a long |

expedition!");

} else {

System.out.println("This is a short

|  |
| --- |
| expedition.");  }    if (Double.parseDouble(totalGoods) > 50) { System.out.println("Large group! Special arrangements may be needed.");  } else {  System.out.println("Manageable group size.");  }    if (costPerKilometer > 2.5) {  System.out.println("The expedition may be costly.");  } else {  System.out.println("The expedition is reasonably priced.");  }    *// Switch case*  System.out.println("\nPlease select an option:");  System.out.println("1. View expedition details");  System.out.println("2. View additional information");  System.out.println("3. Exit");  int option = scanner.nextInt(); |

switch (option) {

case 1:

System.out.println("\nExpedition

|  |
| --- |
| Details:");  System.out.println("Name: " + expeditionName);  System.out.println("Start Date: " + startDate);  System.out.println("End Date: " + endDate);  System.out.println("Total Distance (km):  " + totalDistance);  System.out.println("Range Time (days): " + rangeTime);  System.out.println("Total Time (hours):  " + totalTime);  System.out.println("Total Goods: " + totalGoods);  System.out.println("Cost per Goods: " + costPerTime);  System.out.println("Total Cost: " + totalCost);  System.out.println("Cost per Kilometer:  " + costPerKilometer); break; case 2:  System.out.println("\nAdditional Information:");  if (totalDistance > 1000) {  System.out.println("This is a long expedition!");  } else {  System.out.println("This is a short |

expedition.");

}

|  |
| --- |
| if (Double.parseDouble(totalGoods) > 50) {  System.out.println("Large group!  Special arrangements may be needed.");  } else {  System.out.println("Manageable group size.");  }    if (costPerKilometer > 2.5) {  System.out.println("The expedition may be costly.");  } else {  System.out.println("The expedition is reasonably priced."); } break; case 3:  System.out.println("\nThank you for using the Expedition Java Application!"); break; default:  System.out.println("\nInvalid option selected."); break;  }  scanner.close();  }  } |