LAB # 3

JAVA CONDITIONAL STATEMENT

OBJECTIVE

To Study Java Conditional Statements and Java Input.

THEORY

JAVA Conditional Statements

❖ The if Statement

An **if statement** consists of a boolean expression followed by one or more statements. Use if to specify a block of code to be executed if a specified condition is true.

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

```
if (20 > 18) {
  System.out.println("20 is greater than 18");
}
```

❖ The else Statement

An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is false.

Use else to specify a block of code to be executed, if the same condition is false.

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

❖ The if-else Statement

The Java **if-else** statement works much like the IF statement in any other language. Here, *condition* is a Boolean expression. If *condition* is true, then the statement is executed. If *condition* is false, then the statement is bypassed.

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

❖ The short hand if-else Statement

There is also a short-hand if else, which is known as the **ternary operator** because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

```
variable = (condition) ? expressionTrue :  expressionFalse;
```

```
int time = 20;
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```

❖ Switch Statement

The **switch** statement is Java's multiway branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression. As such, it often provides a better alternative than a large series of **if-else-if** statements.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

```java
int day = 4;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  case 4:
    System.out.println("Thursday");
    break;
  case 5:
    System.out.println("Friday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
  case 7:
    System.out.println("Sunday");
    break;
}
```

Java Input

Java provides different ways to get input from the user. However, in this lab, you will learn to get input from user using the object of Scanner class.

In order to use the object of Scanner, we need to import java.util.Scanner package.

```java
import java.util.Scanner;  // Import the Scanner class
```

Then, we need to create an object of the Scanner class. We can use the object to take input from the user.

```
Scanner myObj = new Scanner(System.in);  // Create a Scanner object
System.out.println("Enter username");

String userName = myObj.nextLine();  // Read user input
System.out.println("Username is: " + userName);  // Output user input
```

Example:

```
import java.util.Scanner; // import the Scanner class

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    String userName;

    // Enter username and press Enter
    System.out.println("Enter username");
    userName = myObj.nextLine();

    System.out.println("Username is: " + userName);
  }
}
```

Output:

```
Enter username
DUET
Username is: DUET
```

Input Types

In the example above, we used the nextLine() method, which is used to read Strings. To read other types, look at the table below:

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

LAB TASK

1. Write a program that displays the user input of First name, Last Name, Age, and Salary.

2. Write a program that can input 2 values from user and show the results of 4 different arithmetic operators on user wish (input operator from the user).

3. Write a program to generate Marksheet and shows the percentage as well. The conditions should be as follows:
   Subjects = your semester subjects and input their marks
   Add marks of all subjects as Total Marks, calculate the percentage of total marks.
   5 subjects = 500 use this for percentage

   Marks >= 85 then A+ Grade
   Marks >= 80 and < 85 then A Grade
   Marks >= 75 and < 80 then B+ Grade
   Marks >= 70 and < 75 then B Grade
   Marks >= 65 and < 70 then C+ Grade
   Marks >= 60 and < 65 then C Grade
   Marks < 60 then Fail