

Control Statements in Java

Aim:

To understand and implement decision-making and looping control statements in Java.

PRE LAB EXERCISE

QUESTIONS

- ✓ List different control statements in Java.

Selection (Decision-making) statements

- if
- if-else
- else-if
- switch

Looping (Iteration) statements

- for
- while
- do-while

Jump (Branching) statements

- break
- continue
- return

- ✓ Difference between for, while, and do-while loops.

Feature	for loop	while loop	do-while loop
Condition check	Before loop starts	Before loop starts	After loop body
Minimum execution	0 times	0 times	At least 1 time
Best used when	Number of iterations is known	Condition-based looping	Loop must run once
Syntax	Compact	Simple	Slightly different

Example:

If condition is false initially:

- for → not executed
- while → not executed
- do-while → executed once

- ✓ What is the use of break and continue?

break

- Used to **terminate the loop or switch statement immediately**
- Control moves **outside** the loop

Example use: stop loop when a condition is met.

continue

- Used to **skip the current iteration**
- Control moves to the **next iteration** of the loop

Example use: skip unwanted values but continue looping.

IN LAB EXERCISE

Objective:

To implement if-else and looping statements.

INPUT STATEMENT:

SCANNER CLASS

- ✓ The Scanner class in Java is used to read input from the user through the keyboard.
It is available in the package java.util.
- ✓ The Scanner object reads different types of input such as integer, float, double, and string and stores them in variables.
- ✓ To use the Scanner class, it must be imported before using it in the program.

SYNTAX:

- ✓ `Scanner sc = new Scanner(System.in);`

Commonly Used Scanner Methods:

- ✓ `nextInt()` – reads an integer value
- ✓ `nextFloat()` – reads a float value
- ✓ `nextDouble()` – reads a double value
- ✓ `next()` – reads a single word
- ✓ `nextLine()` – reads a complete line of text

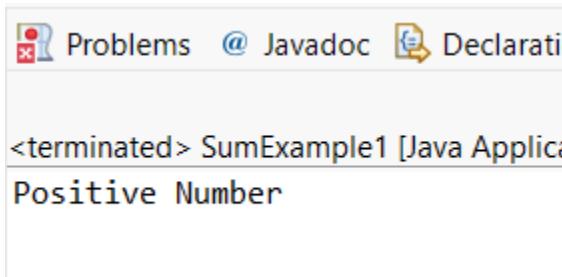
PROGRAMS:

Program 1: Check Whether a Number is Positive

```
class PositiveNumber {  
    public static void main(String[] args) {  
        int n = 5;  
        if (n > 0) {  
            System.out.println("Positive Number");  
        }  
    }  
}
```

Output:

Positive Number



The screenshot shows a Java application window with the title bar 'SumExample1 [Java Application]'. Below the title bar, there are tabs for 'Problems', 'Javadoc', and 'Declarations'. The main area of the window contains the text '<terminated> SumExample1 [Java Application]' followed by the output 'Positive Number'.

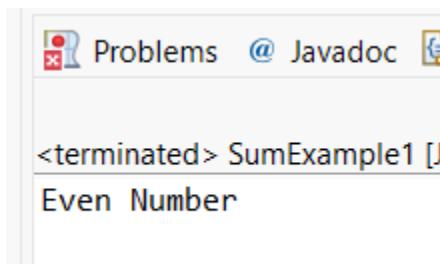
Program 2: Check Whether a Number is Even or Odd

```
class EvenOdd {  
    public static void main(String[] args) {  
        int n = 6;  
        if (n % 2 == 0)  
            System.out.println("Even Number");  
        else  
            System.out.println("Odd Number");  
    }  
}
```

```
}
```

Output:

Even Number



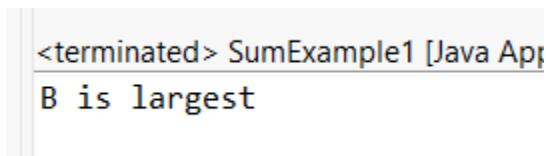
```
<terminated> SumExample1 [J
Even Number
```

Program 3: Find Largest of Two Numbers

```
class LargestTwo {
    public static void main(String[] args) {
        int a = 10, b = 20;
        if(a > b)
            System.out.println("A is largest");
        else
            System.out.println("B is largest");
    }
}
```

Output:

B is largest



```
<terminated> SumExample1 [Java App
B is largest
```

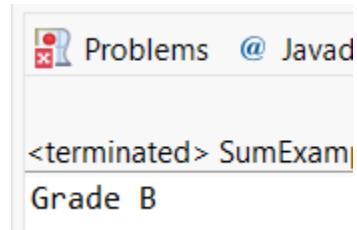
Program 4: Grade Calculation

```
class Grade {
    public static void main(String[] args) {
        int marks = 75;
        if(marks >= 90)
            System.out.println("Grade A");
```

```
else if (marks >= 75)
System.out.println("Grade B");
else if (marks >= 50)
System.out.println("Grade C");
else
System.out.println("Fail");
}
```

Output:

Grade B



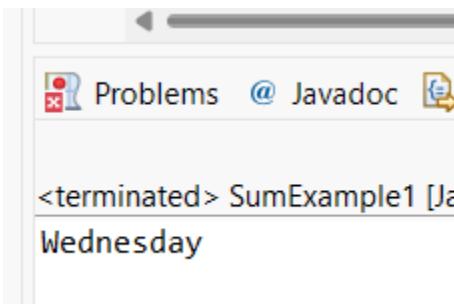
A screenshot of a Java IDE interface. At the top, there's a toolbar with icons for file operations. Below it, a menu bar shows "Problems" and "Javadoc". The main area displays the command "<terminated> SumExam" followed by the output "Grade B".

Program 5: Day of the Week

```
class DaySwitch {
public static void main(String[] args) {
int day = 3;
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;
default: System.out.println("Invalid Day");
}
}
}
```

Output:

Wednesday



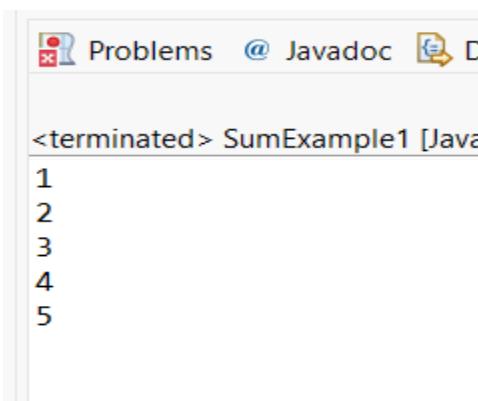
A screenshot of a Java IDE's terminal window. The title bar says 'Problems @ Javadoc'. The terminal output shows '<terminated> SumExample1 [Java]'. Below that, the word 'Wednesday' is printed.

Program 6: Print Numbers from 1 to 5

```
class ForLoop {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

Output:

1
2
3
4
5



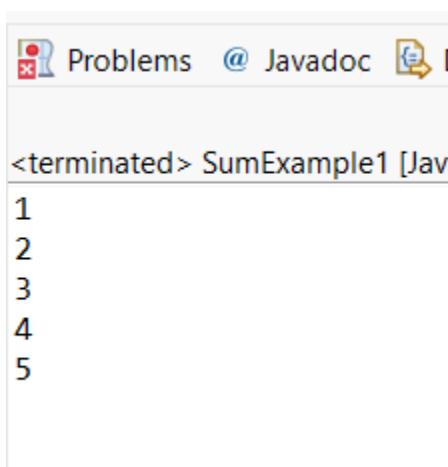
A screenshot of a Java IDE's terminal window. The title bar says 'Problems @ Javadoc'. The terminal output shows '<terminated> SumExample1 [Java]'. Below that, the numbers 1 through 5 are printed on separate lines.

Program 7: Print Numbers from 1 to 5

```
class WhileLoop {  
    public static void main(String[] args) {  
        int i = 1;  
        while (i <= 5) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

Output:

```
1  
2  
3  
4  
5
```



A screenshot of a Java IDE interface. At the top, there are tabs for 'Problems' and 'Javadoc'. Below the tabs, the status bar shows '<terminated> SumExample1 [Java]'. The main workspace displays the numerical output from the previous code: '1', '2', '3', '4', and '5' on separate lines.

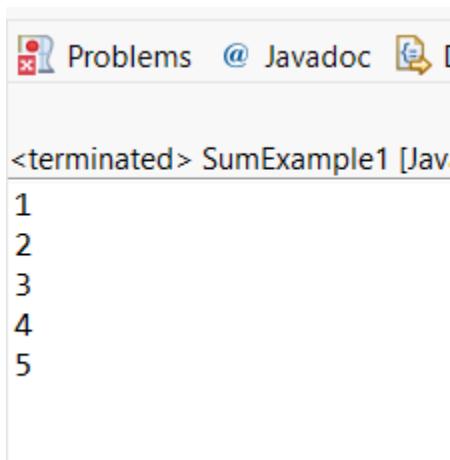
Program 8: Print Numbers from 1 to 5

```
class DoWhileLoop {
```

```
public static void main(String[] args) {  
    int i = 1;  
    do {  
        System.out.println(i);  
        i++;  
    } while (i <= 5);  
}
```

Output:

```
1  
2  
3  
4  
5
```



A screenshot of a Java IDE interface. At the top, there are tabs for 'Problems' and 'Javadoc'. Below the tabs, the text '<terminated> SumExample1 [Java]' is displayed. The main area shows the output of the program, which consists of the numbers 1, 2, 3, 4, and 5, each on a new line.

```
1  
2  
3  
4  
5
```

Program 9: Sum of First 5 Natural Numbers

```
class SumNumbers {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 5; i++) {  
            sum = sum + i;
```

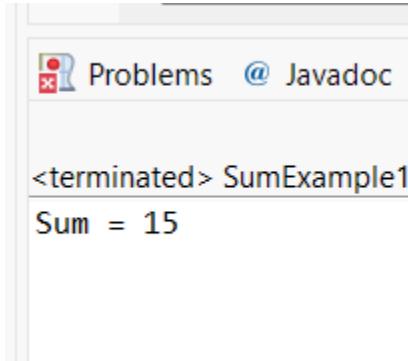
```
}

System.out.println("Sum = " + sum);

}
```

Output:

Sum = 15



```
<terminated> SumExample1
Sum = 15
```

Program 10: Multiplication Table of a Number

```
class MultiplicationTable {

public static void main(String[] args) {

int n = 5;

for (int i = 1; i <= 10; i++) {

System.out.println(n + " x " + i + " = " + (n * i));

}

}

}
```

Output:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

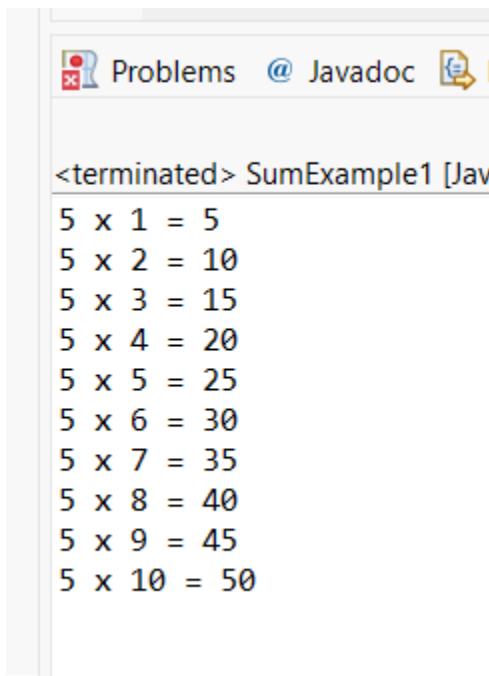
5 x 6 = 30

$5 \times 7 = 35$

$5 \times 8 = 40$

$5 \times 9 = 45$

$5 \times 10 = 50$



The screenshot shows a Java IDE interface with a 'Problems' tab selected. Below it, the output window displays the results of a multiplication program named 'SumExample1'. The output consists of ten lines, each showing the multiplication of 5 by an integer from 1 to 10, followed by the result.

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

POST LAB EXERCISE

- ✓ What is the use of if statement?

The if statement is used to check a condition.

If the condition is true, the given block of code is executed.

Use: Decision making in a program.

- ✓ Difference between if-else and else-if ladder

if-else

Used to check **two conditions**

Only one if and one else

Simple decision making

else-if ladder

Used to check **multiple conditions**

Multiple else-if blocks

Multiple choice decision making

- if-else → pass / fail
- else-if ladder → grade system (A, B, C, D)

- ✓ Why is switch statement used?

The switch statement is used to select one option from many choices based on the value of a variable.

Use:

Makes code cleaner and easier to read

Alternative to long else-if ladder

- ✓ Difference between for, while, and do-while loops.

Feature	for	while	do-while
Condition check	Before loop	Before loop	After loop
Executes at least once	No	No	Yes
Best used when	Iterations known	Condition-based	Must run once

- ✓ Which loop executes at least once?

The do-while loop executes at least once, even if the condition is false.

Result:

Thus the different control statements were executed successfully with expected output.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	5	
In Lab Exercise	10	
Post Lab Exercise	5	
Viva	10	
Total	30	
Faculty Signature		