**Assignment: (Core Java)**

**3 : Control Flow Statements**

**Que.1 If-Else Statements**

**Ans.1** In Java, if-else statements are decision-making statements that allow your program to execute different blocks of code based on conditions. They are part of control flow statements.

Types of if-else as under below:

1. **Simple if Statement**

Example:

public class simpleIf {

public static void main(String[] args) {

int age = 20;

if (age >= 18) {

System.out.println("You are eligible to vote.");

}

}

}

1. **if-else Statement**

public class ifElse {

public static void main(String[] args) {

int number = 15;

if (number % 2 == 0) {

System.out.println(number + " is Even.");

} else {

System.out.println(number + " is Odd.");

}

}

}

1. **if-else-if Ladder**

public class ifElseIf {

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: A");

} else if (marks >= 60) {

System.out.println("Grade: B");

} else if (marks >= 40) {

System.out.println("Grade: C");

} else {

System.out.println("Fail");

}

}

}

1. **Nested if**

public class neestedIf {

public static void main(String[] args) {

int age = 25;

boolean hasVoterID = true;

if (age >= 18) {

if (hasVoterID) {

System.out.println("You are eligible to vote.");

} else {

System.out.println("You must have a voter ID to vote.");

}

} else {

System.out.println("You are not eligible to vote.");

}

}

}

**Que.2 Switch Case Statements**

**Ans.2** In Java, the switch statement is another type of decision-making statement. It is often used as an alternative to multiple if-else-if statements when you need to compare the same variable with multiple values.

**Example:**

public class dayOfWeek {

public static void main(String[] args) {

int day = 3; // 1 = Monday, 2 = Tuesday ...

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;

default:

System.out.println("Invalid day number");

}

}

}

**Que.3 Loops (For, While, Do-While)**

**Ans.3**

| Feature | for Loop | while Loop | do-while Loop |
| --- | --- | --- | --- |
| Condition Checking | At the beginning | At the beginning | At the end |
| Execution | Executes 0 or more times | Executes 0 or more times | Executes 1 or more times |
| Best Use Case | When the number of iterations is known | When the number of iterations is not known but condition must be checked first | When you want to execute at least once, regardless of condition |
| Syntax | Compact (initialization, condition, update in one line) | Initialization & update must be written separately | Similar to while, but condition comes after execution |
| Readability | Good for count-controlled loops | Good for condition-controlled loops | Good for menu-driven programs |

**Que.4 Break and Continue Keywords**

| Feature | **Break** | **Continue** |
| --- | --- | --- |
| Definition | Exits the loop or switch entirely. | Skips the current iteration and moves to the next iteration. |
| Scope | Works in **loops and switch statements**. | Works only in **loops**. |
| Effect on Loop | Terminates the loop immediately. | Loop continues, but skips execution for the current cycle. |
| Flow Control | Jumps **out of the loop/switch**. | Jumps to the **next iteration of the loop**. |
| Use Case | When you want to **stop execution** once a condition is met. | When you want to **ignore certain cases** but keep looping.. |