Module 3: Control Statements in Java

Introduction: Control statements in Java are tools that help manage the flow of execution in a program. These include decisions (if something happens, then do this), loops (repeat something), and jumps (skip or stop a task). They're like traffic signals in programming—guiding when to go, stop, or take a turn!

1. **Conditional Statements:** These help the program decide what to do based on conditions.

if Statement: Checks a condition; if it's true, the code inside runs.

```
int number = 10;
if (number > 0) {
   System.out.println("The number is positive");
}
```

Output: The number is positive

Real-life example: If your score is more than 35, you pass.

if-else Statement: Runs one block if true, another if false.

```
int number = -5;
if (number > 0) {
   System.out.println("Positive number");
} else {
   System.out.println("Negative number");
}
```

Output: Negative number

Real-life example: If it’s raining, take an umbrella; else, wear sunglasses.

else-if Ladder: Checks multiple conditions in order.

```
int marks = 85;
if (marks >= 90) {
   System.out.println("Grade A");
} else if (marks >= 80) {
   System.out.println("Grade B");
} else if (marks >= 70) {
   System.out.println("Grade C");
} else {
   System.out.println("Grade D");
```

```
}
```

Output: Grade B

Real-life example: If marks >= 90 then A, 80-89 then B, etc.

Nested if Statement: An if inside another if. Both must be true.

```
int age = 25;
int weight = 60;
if (age > 18) {
   if (weight > 50) {
      System.out.println("Eligible to donate blood");
   }
}
```

Output: Eligible to donate blood

Real-life example: You can donate blood if you're above 18 and weigh more than 50kg.

2. Switch Statement: Efficient when checking one variable against many values.

```
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;
   default:
      System.out.println("Invalid day");
}
```

Output: Wednesday

Real-life example: Choose lunch menu by selecting a day.

3. Looping Statements: When you want to repeat something multiple times.

for Loop: Runs a block a known number of times.

```
for (int i = 1; i <= 5; i++) {
   System.out.println("Hello " + i);
}
```

Output: Hello 1 Hello 2 Hello 3 Hello 4 Hello 5

Real-life example: Printing 5 invoices or serial numbers.

while Loop: Runs while a condition is true.

```
int i = 1;
while (i <= 3) {
   System.out.println("i = " + i);
   i++;
}
```

Output: i = 1 i = 2 i = 3

Real-life example: Keep stirring coffee until sugar dissolves.

do-while Loop: Runs at least once before checking condition.

```
int i = 1;
do {
   System.out.println("Count: " + i);
   i++;
} while (i <= 2);
```

Output: Count: 1 Count: 2

Real-life example: Try logging in at least once before checking success.

4. **Jump Statements**: Used to change the normal sequence in loops.

break Statement: Exits the loop immediately when a condition is met.

```
for (int i = 1; i <= 5; i++) {
   if (i == 3) break;
   System.out.println(i);
}
```

Output: 1 2

Real-life example: Stop the game if a player wins.

continue Statement: Skips the current iteration and goes to the next.

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) continue;
    System.out.println(i);
}
```

Output: 1 2 4 5

Real-life example: Skip a broken item in a checklist.

Summary Table:

Statement Type	Description	Real-life Analogy
if, if-else, else-if	Decision-making based on conditions	Traffic signal or pass/fail in exam
switch	Executes one case from many options	Menu selection or choosing an option
for, while, do-while	Looping or repeating a block of code	Playing a song 3 times or retrying a task
break, continue	Jump control during loop execution	Leaving a queue (break) or skipping a person (continue)

Note: Try using these concepts in daily logic. For example, write a Java program that checks what to wear based on the weather!