**1. What are control flow statements in Java?**

**Answer:**  
Control flow statements determine the **order in which instructions are executed** in a Java program. Normally, code runs line by line from top to bottom. However, using control flow, we can **make decisions**, **repeat blocks**, or **jump out of loops**. Java provides 3 types:

* Conditional (if, if-else, switch)
* Looping (for, while, do-while)
* Branching (break, continue, return)

These structures help build **dynamic**, **efficient**, and **logic-based programs**.

**✅ 2. Differentiate between if, if-else, and if-else-if statements.**

**Answer:**

* if: Checks a single condition; executes block if true.
* if-else: Executes one block if condition is true, else another block.
* if-else-if: Used to check **multiple conditions** sequentially.

if (x > 0) {

// positive

} else if (x < 0) {

// negative

} else {

// zero

}

This structure ensures only **one block is executed**, making decisions more efficient.

**✅ 3. Explain the switch statement with a use case.**

**Answer:**  
The switch statement is a multi-way branch that compares a variable against multiple values.

switch (day) {

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

default: System.out.println("Invalid");

}

It improves **readability** over long if-else-if chains, especially in menu systems or option selection tasks.

**✅ 4. What happens if we don’t use break in a switch statement?**

**Answer:**  
If break is not used, **fall-through** behavior occurs:  
All subsequent case blocks are executed until a break or end of switch.

switch (2) {

case 1: System.out.println("One");

case 2: System.out.println("Two");

case 3: System.out.println("Three");

}

**Output:**

Two

Three

This may cause **logic errors** if not handled carefully.

**✅ 5. Differentiate between while and do-while loops.**

**Answer:**

| **Feature** | **while** | **do-while** |
| --- | --- | --- |
| Condition check | Before loop body | After loop body |
| Runs at least once | ❌ No | ✅ Yes |

// do-while example

int i = 1;

do {

System.out.println(i);

i++;

} while (i <= 3);

Used when **minimum one execution** is required.

**✅ 6. Write a program to print even numbers from 1 to 10 using for loop.**

**Answer:**

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

if (i % 2 == 0) {

System.out.println(i);

}

}

This loop checks each number from 1 to 10, and prints it if divisible by 2. It's a typical case of using conditional logic inside a loop.

**✅ 7. Explain the structure and parts of a for loop.**

**Answer:**  
A for loop has 3 parts:

for (initialization; condition; update) {

// loop body

}

Example:

for (int i = 0; i < 5; i++) {

System.out.println(i);

}

* Initialization: once at the start
* Condition: checked before each iteration
* Update: executed after each loop

It is compact and ideal when the **iteration count is known**.

**✅ 8. What is the role of continue? Give an example.**

**Answer:**  
The continue statement **skips the current iteration** and moves to the next one.

Example:

for (int i = 1; i <= 5; i++) {

if (i == 3) continue;

System.out.println(i);

}

Output:

1

2

4

5

Use it to **ignore certain values** or conditions in a loop.

**✅ 9. What is the purpose of break in loops?**

**Answer:**  
break immediately **terminates** the loop and control moves to the next statement after the loop.

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

if (i == 5) break;

System.out.println(i);

}

Output:

1 2 3 4

It’s useful in **search operations** or **exiting early**.

**✅ 10. How is return used in Java methods?**

**Answer:**  
return ends method execution and optionally returns a value.

public int add(int a, int b) {

return a + b;

}

* In void methods: return is optional
* In non-void methods: must return a value of correct type

It's essential for **modular programming** and **reusability**.

**✅ 11. Compare break and continue.**

**Answer:**

| **Feature** | **break** | **continue** |
| --- | --- | --- |
| Function | Exits loop | Skips current iteration |
| Loop Ends? | Yes | No |

for (int i = 1; i <= 5; i++) {

if (i == 3) break;

}

vs

for (int i = 1; i <= 5; i++) {

if (i == 3) continue;

}

Both are used for **flow control** inside loops.

**✅ 12. What is an infinite loop? Give example.**

**Answer:**  
A loop that never ends due to a **missing or incorrect exit condition**.

Example:

while (true) {

System.out.println("Running...");

}

Used in:

* Game loops
* Servers
* Waiting systems (with break condition inside)

Must be handled carefully to avoid **performance issues**.

**✅ 13. When do we use nested if statements?**

**Answer:**  
Nested if is used when we want to **check multiple dependent conditions**.

if (age > 18) {

if (citizen) {

System.out.println("Eligible to vote");

}

}

Only if both conditions are true, the action executes. Use carefully to avoid **deep nesting** which reduces readability.

**✅ 14. Can a switch work with String in Java?**

**Answer:**  
Yes. Since Java 7, switch supports String.

String fruit = "Apple";

switch (fruit) {

case "Apple": System.out.println("Red"); break;

case "Banana": System.out.println("Yellow"); break;

}

It’s more readable than using multiple if-else for strings.

**✅ 15. Explain the concept of loop nesting.**

**Answer:**  
Placing one loop inside another is **loop nesting**.

Example:

for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 2; j++) {

System.out.println("i=" + i + ", j=" + j);

}

}

Used in:

* Matrices
* Patterns
* Table generation

Control flows **row-wise and then column-wise**.

**✅ 16. What are labeled breaks?**

**Answer:**  
Used to break out of **nested loops**.

outer:

for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 3; j++) {

if (j == 2) break outer;

}

}

break outer; exits both loops. Helpful in **complex nested structures**.

**✅ 17. Give an example of using for-each loop.**

**Answer:**

int[] arr = {10, 20, 30};

for (int num : arr) {

System.out.println(num);

}

The enhanced for loop is used to **iterate over arrays or collections** without index. It improves code readability.

**✅ 18. Can return be used in void methods?**

**Answer:**  
Yes. return; (without value) is used to **exit early** from a void method.

void show() {

if (x < 0) return;

System.out.println("Value: " + x);

}

Useful for **avoiding unnecessary execution**.

**✅ 19. What is the default case in a switch?**

**Answer:**  
default is the fallback block in a switch when **no cases match**.

switch (n) {

case 1: break;

default: System.out.println("Invalid choice");

}

It's like the **else** in if-else. Optional, but recommended for **completeness**.

**✅ 20. When is it better to use switch over if-else?**

**Answer:**  
Use switch when:

* You're checking the **same variable** against multiple values
* The values are **constant** and known

switch (option) {

...

}

It improves code clarity and performance in some cases. But for **range-based** conditions, if-else is better.