Chapter-4: Control Statements

PART-II: SAMPLE SHORT QUESTIONS

Q1. What are iteration / Looping Statements?

Iteration or looping statements allow us to execute a block of code repeatedly until a condition becomes false. Java provides three types of loops: for, while, and do-while. These are used when tasks need repetition, like printing numbers or processing arrays.

Example:

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

This prints numbers 1 to 5.

Q2. What are jump / Branching Statements?

Jumping statements are used to change the normal flow of control in a program. Java provides break, continue, and return as jump statements. They are useful for exiting loops, skipping iterations, or leaving a method.

Example:

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

Q3. What is Infinite while loop in Java?

An infinite loop is a loop that never terminates because its condition always remains true. They are often used in event-driven programs, like servers or games, which must keep running.

Example:

```
while(true) {
  System.out.println("Running...");
}
```

Q4. What is the importance of Java for-each loop?

The for-each loop is used to iterate through arrays and collections without using an index. It reduces coding effort, avoids errors like array index out of bound, and makes the code cleaner.

Example:

```
int arr[] = {10,20,30};
for(int x : arr) {
  System.out.println(x);
}
```

Q5. Is there a way to access an iteration-counter in Java's for-each loop?

No, the for-each loop does not provide a counter variable. If you need the index while looping, you should use a traditional for loop. However, you can maintain your own counter inside the loop.

Example:

```
int arr[] = {10,20,30};
int i=0;
for(int x : arr) {
  System.out.println("Index " + i + " = " + x);
  i++;
}
```

Q6. Advantages of declaring loop variable as final in enhanced for-loop.

When a loop variable is declared final, it cannot be reassigned inside the loop. This prevents accidental modification of loop elements and ensures data integrity. It is useful in multi-threaded or sensitive programs.

Example:

```
for(final int x : arr) {
  System.out.println(x);
  // x = 5; ❌ not allowed
}
```

Q7. Does returning a value from the loop body break the loop?

Yes, using return inside a loop not only breaks the loop but also exits the entire method. This is stronger than break because it terminates execution immediately.

Example:

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

Q8. What happens when we forget break in switch case?

If you forget break, Java will execute the matched case and continue executing subsequent cases until a break is found. This is called **fall-through behavior** and can lead to errors if not intended.

Example:

```
int day=2;
switch(day) {
  case 1: System.out.println("Mon");
  case 2: System.out.println("Tue"); // no break → prints Tue + Wed
  case 3: System.out.println("Wed");
```

```
}
```

Q9. How will you exit anticipatedly from a loop?

To exit a loop before its condition becomes false, we use the break statement. It is commonly used when a certain condition is met and further execution is unnecessary.

Example:

```
for(int i=1;i<=10;i++) {
  if(i==5) break;   // exit loop at 5
  System.out.println(i);
}
```

Q10. What are Flow Charts?

Flowcharts are graphical diagrams that represent the logical flow of a program. They use arrows for control flow, diamonds for decisions, and rectangles for processes. Flowcharts make programs easier to understand and debug.

Q11. What is the syntax of nested if statement?

A nested if means placing one if inside another if. It is used when multiple conditions must be checked one after another.

Example:

```
if(a > 0) {
  if(a % 2 == 0) {
    System.out.println("Positive Even");
  }
}
```

Q12. What can break statement do in Switch case?

The break statement stops execution of the current case and exits from the switch block. Without it, execution falls through to other cases.

Example:

```
switch(num) {
  case 1: System.out.println("One"); break;
  case 2: System.out.println("Two"); break;
}
```

Q13. Draw the Flow Chart of While Loop.

Flowchart:
Start → [Condition?] → True → Execute body → Back to condition → False → End.

Q14. Difference between dead code and unreachable code in Java.

- **Dead code:** Code that logically never executes (but compiles).
- **Unreachable code:** Code that compiler detects will never be executed (error).

Example:

```
if(false) { System.out.println("Dead"); }   // dead code
return;
System.out.println("Unreachable");        // compile error
```

Q15. Which is considered as a selection statement?

Selection statements allow choosing different paths of execution. In Java, these are if, if-else, if-else-if, and switch.

Example:

```
if(x > 0) System.out.println("Positive");
else System.out.println("Negative");
```

Q16. What is the difference between while and do-while loop?

- while: Condition checked before execution → may run zero times.
- do-while: Condition checked after execution → runs at least once.

Example:

```
do { System.out.println("Runs once"); } while(false);
```

Q17. What are cases in a switch?

Cases define blocks of code that run when the switch expression matches their value. Each must be unique and end with break.

Example:

```
switch(day) {
  case 1: System.out.println("Monday"); break;
  case 2: System.out.println("Tuesday"); break;
}
```

Q18. Can we write switch case using a double variable?

No, switch does not allow float or double types. It only supports byte, short, int, char, String, enum.

Example:

```
// switch(2.5) ❌ not allowed
```

Q19. What is the output of Java program with IF statement?

```
if(1) {
  System.out.println("OK");
}
```

This gives a **compilation error**, because Java expects a boolean in if, not an integer (unlike C/C++).

Q20. What is the output of IF-ELSE-IF program?

Marks = 55.

- First condition (marks>=80) → false.
- Second condition (marks>=35) → true.
  So the output will be: **PASS**.

PART-III: SAMPLE LONG QUESTIONS

Q1. Write a program in Java to display the pattern like right angle triangle with a number.

Explanation:

We use a nested loop: the outer loop controls rows, and the inner loop prints numbers in each row. Each row increases by one digit until the given number of rows is completed.

Program:

```
import java.util.Scanner;


class TrianglePattern {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Input number of rows: ");
      int rows = sc.nextInt();

      for(int i=1; i<=rows; i++) {
         for(int j=1; j<=i; j++) {
            System.out.print(j);
         }
         System.out.println();
      }
   }
}
```

Output (rows = 10):

```
1
12
123
1234
12345
123456
1234567
12345678
```

123456789

12345678910

Q2. What Types of Loops Does Java Support?

Explanation:
Java supports three main loops:

1. **For Loop:** Best when the number of iterations is known.
2. for(int i=1; i<=5; i++) { System.out.println(i); }
3. **While Loop:** Used when the condition is checked before execution.
4. int i=1;
5. while(i<=5) { System.out.println(i); i++; }
6. **Do-While Loop:** Executes the loop body at least once.
7. int i=1;
8. do { System.out.println(i); i++; } while(i<=5);

Q3. When Is It Preferable to Use a Switch Over an If-Then-Else Statement and Vice Versa?

Explanation:

- **Switch** is preferable when:
    - We are comparing the same variable against multiple values.
    - It makes code cleaner than multiple if-else.
- int day=2;
- switch(day) {
- case 1: System.out.println("Monday"); break;
- case 2: System.out.println("Tuesday"); break;
- default: System.out.println("Other day");
- }
- **If-Else** is preferable when:
    - Conditions involve ranges or complex boolean expressions.
- int marks=75;
- if(marks>=80) System.out.println("A Grade");
- else if(marks>=50) System.out.println("B Grade");
- else System.out.println("Fail");

Q4. What are Conditional Statements in Java? Explain these with Examples.

Explanation:
Conditional statements allow decision-making in programs. They execute different blocks of code based on boolean conditions.

```
1. If Statement:
2. int x=10;
3. if(x>0) System.out.println("Positive");
4. If-Else Statement:
5. int x=-5;
6. if(x>=0) System.out.println("Non-negative");
7. else System.out.println("Negative");
8. If-Else-If Ladder:
9. int marks=70;
10. if(marks>=80) System.out.println("Distinction");
11. else if(marks>=50) System.out.println("Pass");
12. else System.out.println("Fail");
13. Switch Statement:
14. int num=2;
15. switch(num) {
16.     case 1: System.out.println("One"); break;
17.     case 2: System.out.println("Two"); break;
18.     default: System.out.println("Other");
19. }
```

Q5. Write a Program that prints the Multiplication Table of 12 on the screen.

Explanation:
We use a for loop to iterate from 1 to 10 and multiply each number with 12.

Program:

```
class MultiplicationTable {
   public static void main(String[] args) {
     int num = 12;
     for(int i=1; i<=10; i++) {
        System.out.println(num + " x " + i + " = " + (num*i));
     }
   }
}
```

Output:

```
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
```

12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120