Module 1: Introduction to Java Fundamentals

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

January 6, 2026

Module 1: Introduction to Java Fundamentals

- OOP Paradigm and Features of Java
- JVM, Bytecode, Java Program Structure
- Data Types, Variables, Naming Conventions
- Operators, Control and Looping Constructs
- One- and Multi-dimensional Arrays
- Enhanced for-loop
- Strings, StringBuffer, StringBuilder, Math Class
- Wrapper Classes

`if--else` Statement — Explanation

The `if--else` statement allows a program to **choose between two actions** based on a condition.

`if--else` Statement — Explanation

The `if--else` statement allows a program to **choose between two actions** based on a condition.

In simple words:

- A condition is checked
- If it is **true** $\rightarrow$ one block executes
- If it is **false** $\rightarrow$ another block executes

`if--else` Statement — Explanation

The `if--else` statement allows a program to **choose between two actions** based on a condition.

In simple words:

- A condition is checked
- If it is **true** $\rightarrow$ one block executes
- If it is **false** $\rightarrow$ another block executes

Why do we need `if--else`?

- When there are exactly two outcomes
- To avoid writing multiple `if` statements
- To make decisions clear and readable

if--else — Syntax (Java)

Syntax:

```
if (condition) {
 // statements if condition is true
} else {
 // statements if condition is false
}
```

if--else — Syntax (Java)

Syntax:

```
if (condition) {
 // statements if condition is true
} else {
 // statements if condition is false
}
```

Important Rules:

- Condition must return **boolean**
- Exactly one block executes
- Curly braces define code blocks

if--else — Python vs Java

Python

```
marks = 40

if marks >= 50:
print("PASS")
else:
print("FAIL")
```

- Indentation-based
- No braces
- No semicolons

Java

```
int marks = 40;

if (marks >= 50) {
 System.out.println("PASS");
} else {
 System.out.println("FAIL");
}
```

- Uses braces {}
- Condition inside ()
- Statements end with ;

`if--else` — Question 1

Write a Java program to validate login:

- If username is `"admin"` and password is `1234` print `"Login Successful"`
- Else print `"Invalid Credentials"`

`if--else` — Question 1

Write a Java program to validate login:

- If username is `"admin"` and password is `1234` print `"Login Successful"`
- Else print `"Invalid Credentials"`

Test with: username = `"admin"`, password = `1234`

if--else — Solution 1

```
public class LoginValidation {
 public static void main(String[] args) {

  String username = "admin";
  int password = 1234;

  if (username.equals("admin") && password == 1234) {
   System.out.println("Login Successful");
  } else {
   System.out.println("Invalid Credentials");
  }
 }
}
```

`if--else` — Question 2

Write a Java program to check whether a number is:

- Even $\rightarrow$ print `"Even"`
- Odd $\rightarrow$ print `"Odd"`

`if--else` — Question 2

Write a Java program to check whether a number is:

- Even $\rightarrow$ print `"Even"`
- Odd $\rightarrow$ print `"Odd"`

Test with: number $= 17$

if--else — Solution 2

```
public class EvenOdd {
 public static void main(String[] args) {

  int num = 17;

  if (num % 2 == 0) {
   System.out.println("Even");
  } else {
   System.out.println("Odd");
  }
 }
}
```

`if--else` — Question 3

Write a Java program to find the greater of two numbers:

- If first number is greater $\rightarrow$ print `"First is "`
- Else $\rightarrow$ print `"Second is "`

`if--else` — Question 3

Write a Java program to find the greater of two numbers:

- If first number is greater $\rightarrow$ print `"First is "`
- Else $\rightarrow$ print `"Second is "`

Test with: a $= 12$, b $= 25$

if--else — Solution 3

```
public class MaxOfTwo {
 public static void main(String[] args) {

  int a = 12;
  int b = 25;

  if (a > b) {
   System.out.println("First is greater");
  } else {
   System.out.println("Second is greater");
  }
 }
}
```

`else--if` Ladder — Explanation

The `else--if` **ladder** is used when a program must **choose one option from many conditions**.

`else--if` Ladder — Explanation

The `else--if` **ladder** is used when a program must **choose one option from many conditions**.
In simple words:

- Conditions are checked one by one
- The first **true** condition executes
- Remaining conditions are skipped

`else--if` Ladder — Explanation

The `else--if` **ladder** is used when a program must **choose one option from many conditions**.

In simple words:

- Conditions are checked one by one
- The first **true** condition executes
- Remaining conditions are skipped

Why do we need it?

- When there are more than two outcomes
- To handle ranges (marks, salary slabs, age groups)
- Cleaner than many separate `if` statements

else--if Ladder — Syntax

Syntax:

```
if (condition1) {
 // statements
} else if (condition2) {
 // statements
} else if (condition3) {
 // statements
} else {
 // default statements
}
```

else--if Ladder — Syntax

Syntax:

```
if (condition1) {
 // statements
} else if (condition2) {
 // statements
} else if (condition3) {
 // statements
} else {
 // default statements
}
```

Important Rules:

- Conditions are checked **top to bottom**
- Only **one block** executes
- `else` is optional but recommended

else--if — Python vs Java

Python

```
marks = 78

if marks >= 90:
print("A")
elif marks >= 75:
print("B")
elif marks >= 50:
print("C")
else:
print("Fail")
```

- Uses `elif`
- Indentation-based

Java

```
int marks = 78;

if (marks >= 90) {
 System.out.println("A");
} else if (marks >= 75) {
 System.out.println("B");
} else if (marks >= 50) {
 System.out.println("C");
} else {
 System.out.println("Fail");
}
```

- Uses `else if`
- Braces define blocks

else--if Ladder — Question 1

Write a Java program to assign grades based on marks:

- $\geq 90 \rightarrow$ Grade A
- $\geq 75 \rightarrow$ Grade B
- $\geq 50 \rightarrow$ Grade C
- Else $\rightarrow$ Fail

`else--if` Ladder — Question 1

Write a Java program to assign grades based on marks:

- $\geq 90 \rightarrow$ Grade A
- $\geq 75 \rightarrow$ Grade B
- $\geq 50 \rightarrow$ Grade C
- Else $\rightarrow$ Fail

Test with: `marks = 78`

else--if Ladder — Solution 1

```
public class GradeCalculator {
 public static void main(String[] args) {

  int marks = 78;

  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("Grade: FAIL");
  }
 }
}
```

`else--if` Ladder — Question 2

Write a Java program to categorize a person based on age:

- $\geq 60 \rightarrow$ Senior Citizen
- $\geq 18 \rightarrow$ Adult
- Else $\rightarrow$ Minor

`else--if` Ladder — Question 2

Write a Java program to categorize a person based on age:

- $\geq 60 \rightarrow$ Senior Citizen
- $\geq 18 \rightarrow$ Adult
- Else $\rightarrow$ Minor

Test with: `age = 25`

else--if Ladder — Solution 2

```
public class AgeCategory {
 public static void main(String[] args) {

  int age = 25;

  if (age >= 60) {
   System.out.println("Senior Citizen");
  } else if (age >= 18) {
   System.out.println("Adult");
  } else {
   System.out.println("Minor");
  }
 }
}
```

`else--if` Ladder — Question 3

Write a Java program to classify electricity usage:

- $\leq 100 \rightarrow$ Low Usage
- $\leq 300 \rightarrow$ Medium Usage
- $> 300 \rightarrow$ High Usage

`else--if` Ladder — Question 3

Write a Java program to classify electricity usage:

- $\leq 100 \rightarrow$ Low Usage
- $\leq 300 \rightarrow$ Medium Usage
- $> 300 \rightarrow$ High Usage

Test with: `units = 280`

else--if Ladder — Solution 3

```
public class ElectricityUsage {
 public static void main(String[] args) {

  int units = 280;

  if (units <= 100) {
   System.out.println("Low Usage");
  } else if (units <= 300) {
   System.out.println("Medium Usage");
  } else {
   System.out.println("High Usage");
  }
 }
}
```

else--if Ladder — Question 4

Write a Java program to determine income tax slab:

- $\leq$ 2,50,000 $\rightarrow$ No Tax
- $\leq$ 5,00,000 $\rightarrow$ 5% Tax
- $\leq$ 10,00,000 $\rightarrow$ 20% Tax
- $>$ 10,00,000 $\rightarrow$ 30% Tax

`else--if` Ladder — Question 4

Write a Java program to determine income tax slab:

- $\leq$ 2,50,000 $\rightarrow$ No Tax
- $\leq$ 5,00,000 $\rightarrow$ 5% Tax
- $\leq$ 10,00,000 $\rightarrow$ 20% Tax
- $>$ 10,00,000 $\rightarrow$ 30% Tax

Test with: `income = 6,50,000`

else--if Ladder — Solution 4

```
public class TaxSlab {
 public static void main(String[] args) {

  int income = 650000;

  if (income <= 250000) {
   System.out.println("No Tax");
  } else if (income <= 500000) {
   System.out.println("Tax Slab: 5%");
  } else if (income <= 1000000) {
   System.out.println("Tax Slab: 20%");
  } else {
   System.out.println("Tax Slab: 30%");
  }
 }
}
```

else--if Ladder — Question 5

Write a Java program to identify a character type:

- Digit → "Digit"
- Uppercase letter → "Uppercase"
- Lowercase letter → "Lowercase"
- Else → "Special Character"

`else--if` Ladder — Question 5

Write a Java program to identify a character type:

- Digit → `"Digit"`
- Uppercase letter → `"Uppercase"`
- Lowercase letter → `"Lowercase"`
- Else → `"Special Character"`

Test with: `ch = 'A'`

else--if Ladder — Solution 5

```
public class CharacterType {
 public static void main(String[] args) {

  char ch = 'A';

  if (ch >= '0' && ch <= '9') {
   System.out.println("Digit");
  } else if (ch >= 'A' && ch <= 'Z') {
   System.out.println("Uppercase");
  } else if (ch >= 'a' && ch <= 'z') {
   System.out.println("Lowercase");
  } else {
   System.out.println("Special Character");
  }
 }
}
```

`switch` Statement — What is it?

The `switch` statement is a decision-making control structure used to execute **one block of code** from many choices.

`switch` Statement — What is it?

The `switch` statement is a decision-making control structure used to execute **one block of code** from many choices.

In simple words:

- A value is checked once
- It is compared with multiple cases
- The matching case executes

`switch` Statement — What is it?

The `switch` statement is a decision-making control structure used to execute **one block of code** from many choices.

In simple words:

- A value is checked once
- It is compared with multiple cases
- The matching case executes

Why use `switch`?

- Cleaner than long `else--if` ladders
- Best for menu-driven programs
- Easy to read and maintain

`switch` — Real-Life Analogy

Think of a **TV remote**:

`switch` — Real-Life Analogy

Think of a **TV remote**:

- Press 1 $\rightarrow$ News channel
- Press 2 $\rightarrow$ Sports channel
- Press 3 $\rightarrow$ Movie channel

`switch` — Real-Life Analogy

Think of a **TV remote**:

- Press 1 $\rightarrow$ News channel
- Press 2 $\rightarrow$ Sports channel
- Press 3 $\rightarrow$ Movie channel

Only one button works at a time.

`switch` — Real-Life Analogy

Think of a **TV remote**:

- Press 1 $\rightarrow$ News channel
- Press 2 $\rightarrow$ Sports channel
- Press 3 $\rightarrow$ Movie channel

Only one button works at a time.
Similarly in Java:

- One value
- Many options
- Only one case executes

switch vs if--else

- Use `if--else` when:
  - Conditions involve ranges (marks > 50)
  - Logical operators are needed

switch vs if--else

- Use `if--else` when:
  - Conditions involve ranges (marks > 50)
  - Logical operators are needed
- Use `switch` when:
  - You compare one value with many fixed options
  - Menu-driven programs
  - Cleaner multi-choice logic

switch — Syntax

```
switch (expression) {
 case value1:
 // statements
 break;
 case value2:
 // statements
 break;
 default:
 // statements
}
```

switch — Syntax

```
switch (expression) {
 case value1:
 // statements
 break;
 case value2:
 // statements
 break;
 default:
 // statements
}
```

Important Parts:

- `expression` → value to be checked
- `case` → possible values
- `break` → stops execution
- `default` → runs if no case matches

How `switch` Works

1. Expression is evaluated
2. Control jumps to matching `case`
3. Code runs until `break`
4. If no match $\rightarrow$ `default` executes

How `switch` Works

1. Expression is evaluated
2. Control jumps to matching `case`
3. Code runs until `break`
4. If no match $\rightarrow$ `default` executes

Golden Rule:

- Only one case should execute
- Always use `break`

Why `break` is Important

Without `break`, Java follows **fall-through behavior**.

Why `break` is Important

Without `break`, Java follows **fall-through behavior**.
Meaning:

- After matching a case,
- It continues executing the next cases too

Why `break` is Important

Without `break`, Java follows **fall-through behavior**.

Meaning:

- After matching a case,
- It continues executing the next cases too

Hence:

- Use `break` to stop execution
- Avoid unwanted outputs

Valid Data Types in `switch`

In Java, `switch` works with:

Valid Data Types in `switch`

In Java, `switch` works with:

- `int`, `byte`, `short`, `char`
- `String` (Java 7+)
- `enum`

Valid Data Types in `switch`

In Java, `switch` works with:

- `int`, `byte`, `short`, `char`
- `String` (Java 7+)
- `enum`

Not allowed:

- `float`, `double`
- `boolean`

When NOT to use `switch`

Do NOT use `switch` when:

When NOT to use `switch`

Do NOT use `switch` when:

- Conditions involve ranges
  - Example: marks > 50

When NOT to use `switch`

Do NOT use `switch` when:

- Conditions involve ranges
  - Example: marks $>$ 50
- Conditions use logical operators
  - Example: age $>$ 18 AND hasID

When NOT to use `switch`

Do NOT use `switch` when:

- Conditions involve ranges
  - Example: marks $>$ 50
- Conditions use logical operators
  - Example: age $>$ 18 AND hasID
- You need complex expressions

When NOT to use `switch`

Do NOT use `switch` when:

- Conditions involve ranges
  - Example: marks $>$ 50
- Conditions use logical operators
  - Example: age $>$ 18 AND hasID
- You need complex expressions

Use `if--else` instead in these cases.

Common Mistakes in `switch`

- Forgetting `break`

Common Mistakes in `switch`

- Forgetting `break`
- Using duplicate `case` values

Common Mistakes in `switch`

- Forgetting `break`
- Using duplicate `case` values
- Using invalid data types

Common Mistakes in `switch`

- Forgetting `break`
- Using duplicate `case` values
- Using invalid data types
- Expecting range conditions to work

Common Mistakes in `switch`

- Forgetting `break`
- Using duplicate `case` values
- Using invalid data types
- Expecting range conditions to work
- Forgetting `default`

Common Mistakes in `switch`

- Forgetting `break`
- Using duplicate `case` values
- Using invalid data types
- Expecting range conditions to work
- Forgetting `default`

Tip: *Use `switch` only for exact-value matching.*

Q1 (Fall-Through)

Code:

```
int x = 2;

switch (x) {
 case 1:
 System.out.println("One");
 case 2:
 System.out.println("Two");
 case 3:
 System.out.println("Three");
}
```

Q1 (Fall-Through)

Code:

```
int x = 2;

switch (x) {
 case 1:
 System.out.println("One");
 case 2:
 System.out.println("Two");
 case 3:
 System.out.println("Three");
}
```

Output:

- Two
- Three

Q1 (Fall-Through)

Code:

```
int x = 2;

switch (x) {
 case 1:
 System.out.println("One");
 case 2:
 System.out.println("Two");
 case 3:
 System.out.println("Three");
}
```

Output:

- Two
- Three

Reason: No `break` → execution continues to next cases.

Q2 (No Default)

Code:

```
int day = 9;

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

Q2 (No Default)

Code:

```
int day = 9;

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

Output:

- No output

Q2 (No Default)

Code:

```
int day = 9;

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

Output:

- No output

Reason: No matching case and no `default`.

Q3 (Duplicate Case)

Code:

```
int x = 1;

switch (x) {
 case 1:
 System.out.println("One");
 break;
 case 1:
 System.out.println("Again One");
 break;
}
```

Q3 (Duplicate Case)

Code:

```
int x = 1;

switch (x) {
 case 1:
 System.out.println("One");
 break;
 case 1:
 System.out.println("Again One");
 break;
}
```

Answer:

- Compile-time error

Q3 (Duplicate Case)

Code:

```
int x = 1;

switch (x) {
 case 1:
 System.out.println("One");
 break;
 case 1:
 System.out.println("Again One");
 break;
}
```

Answer:

- Compile-time error

Reason: Duplicate case labels are not allowed.

Q4 (Break Scope)

Code:

```
int x = 2;
switch (x) {
 case 2:
 if (x == 2) {
  System.out.println("Matched");
  break;
 }
 System.out.println("After If");
 default:
 System.out.println("Default");
}
```

Q4 (Break Scope)

Code:

```
int x = 2;
switch (x) {
 case 2:
 if (x == 2) {
  System.out.println("Matched");
  break;
 }
 System.out.println("After If");
 default:
 System.out.println("Default");
}
```

Output:

- Matched

Q4 (Break Scope)

Code:

```
int x = 2;
switch (x) {
 case 2:
 if (x == 2) {
  System.out.println("Matched");
  break;
 }
 System.out.println("After If");
 default:
 System.out.println("Default");
}
```

Output:

- Matched

Reason: `break` exits the entire `switch`, not just the `if`.

Q5 (Char Confusion)

Code:

```
char ch = 'A';
switch (ch) {
 case 65:
 System.out.println("ASCII A");
 break;
 case 'A':
 System.out.println("Character A");
 break;
}
```

Q5 (Char Confusion)

Code:

```
char ch = 'A';
switch (ch) {
 case 65:
 System.out.println("ASCII A");
 break;
 case 'A':
 System.out.println("Character A");
 break;
}
```

Answer:

- Compile-time error

Q5 (Char Confusion)

Code:

```
char ch = 'A';
switch (ch) {
 case 65:
 System.out.println("ASCII A");
 break;
 case 'A':
 System.out.println("Character A");
 break;
}
```

Answer:

- Compile-time error

Reason:

- 'A' has ASCII value 65
- Both case 65 and case 'A' are duplicates

Q6 (String Sensitivity)

Code:

```
String lang = "java";

switch (lang) {
 case "Java":
 System.out.println("Matched Java");
 break;
 case "java":
 System.out.println("Matched java");
 break;
}
```

Q6 (String Sensitivity)

Code:

```
String lang = "java";

switch (lang) {
 case "Java":
 System.out.println("Matched Java");
 break;
 case "java":
 System.out.println("Matched java");
 break;
}
```

Output:

- Matched java

Q6 (String Sensitivity)

Code:

```
String lang = "java";

switch (lang) {
 case "Java":
 System.out.println("Matched Java");
 break;
 case "java":
 System.out.println("Matched java");
 break;
}
```

Output:

- Matched java

Reason: `switch` with String is **case-sensitive**.

Q7 (Expression in Switch)

Code:

```
int x = 2;

switch (x + 1) {
 case 2:
 System.out.println("Two");
 break;
 case 3:
 System.out.println("Three");
 break;
}
```

Q7 (Expression in Switch)

Code:

```
int x = 2;

switch (x + 1) {
 case 2:
 System.out.println("Two");
 break;
 case 3:
 System.out.println("Three");
 break;
}
```

Output:

- Three

Q7 (Expression in Switch)

Code:

```
int x = 2;

switch (x + 1) {
 case 2:
 System.out.println("Two");
 break;
 case 3:
 System.out.println("Three");
 break;
}
```

Output:

- Three

Reason: `x + 1 = 3` $\rightarrow$ case 3 matches.

Thank You!

Stay Connected

Premanand S

Email: premanand.s@vit.ac.in
Phone: +91-7358679961

LinkedIn: linkedin.com/in/premsanand
Instagram: instagram.com/premsanand
WhatsApp Channel: anandsDataX

Google Scholar: Google Scholar Profile
GitHub: github.com/anandprems