If-else and switch Statement

Talent Accelerator
powered by

Agenda

- If statement
- If-else statement
- Logical/boolean operations Ternary operator
- Nested if/else structure
- Logical Operators
- Switch multiple-selection structure

zalando

If statement

Talent Accelerator powered by START

What is “If”?

if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statements is executed otherwise not.

Syntax of if Statement

```
if(condition)
{
//Statements to execute if
//condition is true
}
```

zalando

Talent Accelerator
powered by START

If Structure

```
public class StudentGrade {
public static void main(String[] args) {
 int grade = 65;  // You can change this value to test different grades

if (grade >= 60) {
System.out.println("Passed");
    } else {
 System.out.println("Failed");
    }
  }
}
```

zalando

If Structure

Here's the breakdown of the Java code:

- int grade = 65;: This sets the student's grade. You can change the value to test the program.
- if (grade >= 60): This checks if the student's grade is greater than or equal to 60.
- System.out.println("Passed");: If the condition is true, this statement is executed, and "Passed" is printed.
- System.out.println("Failed");: If the condition is false, this statement is executed, and "Failed" is printed.

If Structure

- Flowchart
    - Graphical representation of an algorithm
    - Special purpose symbols connected by arrows(flowlines)
    - Rectangle symbol(action symbol)
        - Any type of action
    - Oval symbol
        - Beginning or end of a program, or a section of code(circles)
- Single-entry/single exit control structure
    - Connect exit point of one to entry point of the next
    - Control structure stacking
- Selection Structure
    - Choose among alternative courses of action
    - Pseudo code example:
        - *“If student`s grade is greater than or equal to 60 Print “Passed””*
    - If the condition is true:
        - Print statement executed, program continues to next statement

Talent Accelerator
powered by START

If Structure

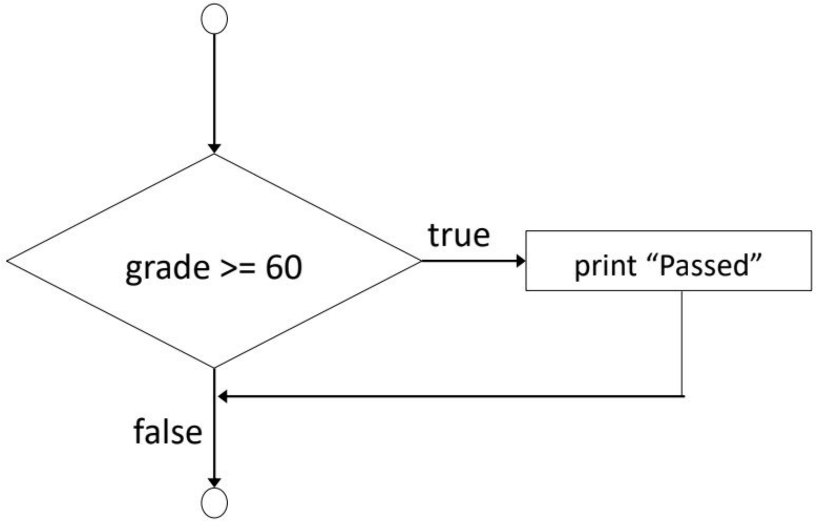
  - If the condition is false
    - Print statement ignored, program continue
- Translation into Java
  - *"If student`s grade is greater than or equal to 60 Print "Passed""*
  -

```
if (grade >= 60) {
System.out.println("Passed");
```

- Diamond symbol(decision symbol)
  - Indicates decision is to be made
  - Contains an expression that can be true or false
- If structure
  - Single entry / single exit

zalando

If Structure FlowChart



A decision can be made on any expression.

zero - **`false`**

nonzero - **`true`**

Example:

`3 - 4` is **`true`**

If-else statement

What is “If-else”?

If- else together represents the set of Conditional statements in Java that are executed according to the condition which is true.

Syntax of if-else Statement

```
if (condition)
{
  // Executes this block if
   // condition is true
}
else
{
   // Executes this block if
   // condition is false
}
```

Talent Accelerator
powered by START

If-else Structure

```
int grade = 85; // Replace with the actual grade
if (grade >= 90) {
 System.out.println("A");
} else if (grade >= 80) {
System.out.println("B");
} else if (grade >= 70) {
System.out.println("C");
 } else if (grade >= 60)
{ System.out.println("D");
} else {
 System.out.println("F");
 }
```

zalando

If-else Structure

Here's the breakdown of the Java code:

- int grade = 85; This line declares an integer variable named grade and initializes it with a value of 85. This is the student's grade, and you can change this value to test the program.
- if (grade >= 90): This if statement checks if the variable grade is greater than or equal to 90. If this condition is true, the code block inside the if statement will be executed.
- System.out.println("A");: This line will execute if the preceding if condition is true. It will print "A" to the console, indicating that the student received an 'A'.
- else if (grade >= 80): IIf the previous if condition was not met, then this else if condition is checked. It tests whether grade is greater than or equal to 80 and less than 90.
- The same logic applies to the rest of the else if and else blocks for grades C, D, and F. Each block contains a condition to check, and an action to perform if the condition is true.

If-else Structure

- If
  - Performs action if condition is true
- If-else
  - Different actions if condition is true or false
- Pseudocode

  -
    ```
    If student`s grade is greater than or equal to 60
     Print "Passed""
    else
    Print "failed"
    ```

- Java code:

```
if (grade >= 60) {
System.out.println("Passed");
    } else {
 System.out.println("Failed");
    }
```

Talent Accelerator powered by START

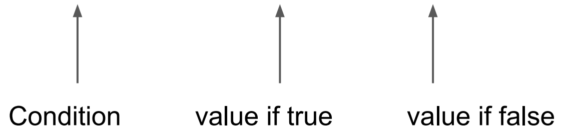
zalando

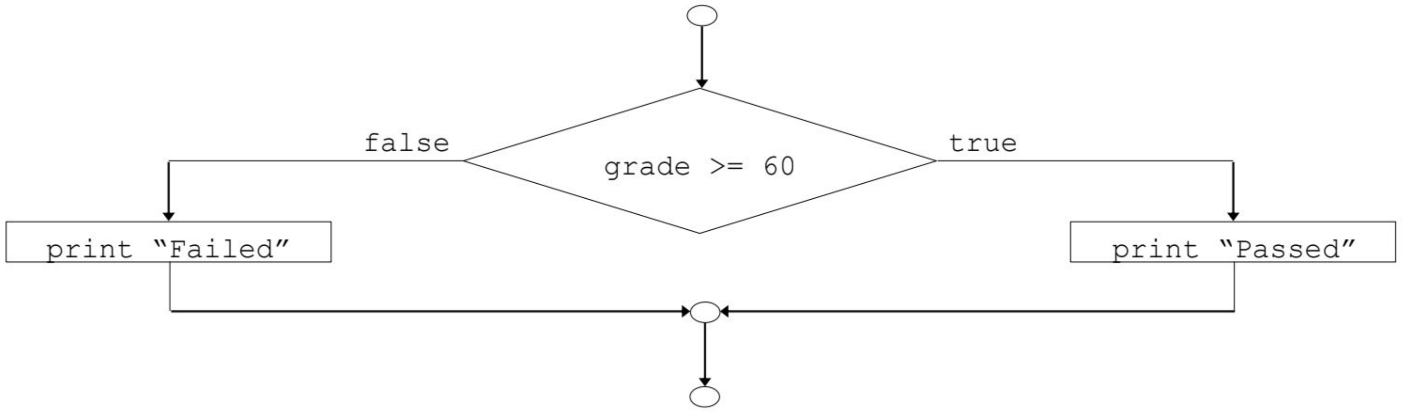
Logical/boolean operations Ternary operator

Talent Accelerator powered by START

Logical/boolean operations Ternary operator

- Ternary conditional operator (? :)
  - Three arguments (condition, value if true , value if false)
- Code could be written:
  - `String result = (grade >= 60) ? "Passed" : "Failed";`





zalando

Nested if/else structure

Talent Accelerator
powered by START

What is “ nested If-else”?

A nested if-else statement is an if-else statement contained inside another if or else block. In other words, it's an if-else construct within the code block of another if or else construct. This allows for more complex decision-making pathways within a program.

Syntax of nested if-else Statement

```
if (condition1) {

// code to be executed if condition1 is true

 } else {

 if (condition2) {

 // code to be executed if condition1 is false and
condition2 is true

 } else {

// code to be executed if all conditions are false } }
```

zalando

Nested If-else Structure

```
int grade = 85; // Replace with the actual grade

if (grade >= 90) {
    System.out.println("A");
} else {
    if (grade >= 80) {
        System.out.println("B");
    } else {
        if (grade >= 70) {
            System.out.println("C");
        } else {
            if (grade >= 60) {
                System.out.println("D");
            } else {
                System.out.println("F");
            }
        }
    }
}
```

Nested If-else Structure

Here's the breakdown of the Java code:

- Certainly! Here's a breakdown of the Java code:
-
- 1. `int grade = 85;`: This line declares an integer variable named `grade` and initializes it with a value of 85. This represents the grade you're testing. You can replace 85 with any other grade you want to test.
-
- 2. `if (grade >= 90)`: This is the outermost `if` statement that checks if `grade` is greater than or equal to 90.
-   - `System.out.println("A");`: If the condition is true, this line will print "A" to the console.
-
- 3. `else`: If the first `if` condition is false (grade is less than 90), the program will enter this `else` block and check the following nested `if`-`else` statements:
-   - `if (grade >= 80)`: Checks if `grade` is greater than or equal to 80 but less than 90.

- - `System.out.println("B");`: Prints "B" if the above condition is true.
- - `else`: If the grade is less than 80, it moves to another nested `if`-`else` statement:
- - `if (grade >= 70)`: Checks if `grade` is greater than or equal to 70 but less than 80.
- - `System.out.println("C");`: Prints "C" if true.
- - `else`: If the grade is less than 70, another nested `if`-`else` statement:
- - `if (grade >= 60)`: Checks if `grade` is greater than or equal to 60 but less than 70.
- - `System.out.println("D");`: Prints "D" if true.
- - `else`: If all other conditions were false (grade is less than 60).
- - `System.out.println("F");`: Prints "F".

The nested `if`-`else` structure ensures that only one of the conditions will be true, executing its corresponding code block and skipping the others. It's like a chain of checks that goes from top to bottom, stopping at the first true condition it finds.

Nested If-else Structure

- Nested if-else structures
  - One inside another,test for multiple cases
  - Once condition met, other statements skipped

```
if student's grade is greater than or equal to 90
        Print "A"
else
        if student's grade is greater than or equal to 80
                Print "B"
        else
        if student's grade is greater than or equal to 70
                        Print "C"
                else
                if student's grade is greater than or equal to 60
                                Print "D"
                        else
                                        Print "F"
```

Break 10 minutes

Logical Operators

Logical operators

- Used as condition in loops, if statements
- && (Logical AND)
  - True if both conditions are true
  - if (gender == 1 && age >= 65) {
    seniorFemales++;
- || (Logical OR)
  - True if either of condition is true
  - if (semesterAverage >= 90 || finalExam >= 90) {
    System.out.println("Student grade is A");
    }
- !(Logical NOT)
  - Returns true when its condition is false & vice versa

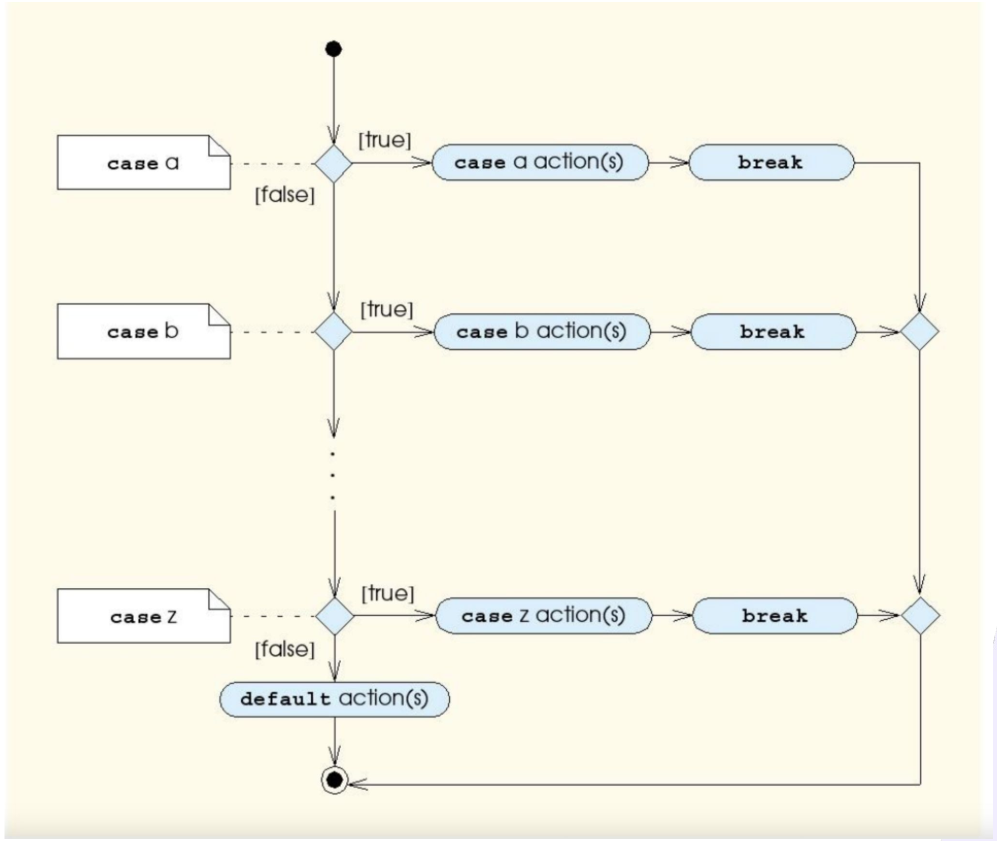
Switch multiple-selection structure

Switch Statement

switch provides a better alternative than if-else-if when the execution follows several branches depending on the value of an expression.
General form:

```
 switch (expression) {
case value1: statement1; break;
case value2: statement2; break;
 case value3: statement3; break;
 ...
default: statement;
}
```

Switch Structure



Switch Code example

```
// Import Scanner class for reading input
import java.util.Scanner;

// Define the class "Remark_According"
public class Remark_According {
    // Main method to execute the program
    public static void main(String args[]) {
        // Create a Scanner object to read input
        Scanner input = new Scanner(System.in);
        // Declare variable to store grade
        char g;
        // Print the options for grades to the console
        System.out.println("A, B, C, D or F");
        // Prompt user to enter a grade
        System.out.print("Please Select Grade :");
        g = input.next().charAt(0);  // Read the first character of the user's input
        // Display the selected grade
        System.out.println("Grade : " + g);


        // Convert grade to uppercase to make the switch case-insensitive
        char grade = Character.toUpperCase(g);
        // Switch statement to evaluate the grade
switch (grade) {
      case 'A':
        System.out.println("Remark : Excellent!");
        break;  // Exit switch statement
      case 'B':
        System.out.println("Remark : Well Done");
        break;  // Exit switch statement
      case 'C':
        System.out.println("Remark : Very Good");
        break;  // Exit switch statement
      case 'D':
        System.out.println("Remark : Good");
        break;  // Exit switch statement
      case 'F':
        System.out.println("Remark : Fail");
        break;  // Exit switch statement
      default:
        System.out.println("Invalid Grade");  // For any character not in 'A', 'B', 'C', 'D
        break;  // Optional, exits switch statement
    }
    // Close the scanner to prevent resource leak
    input.close(); }}
```

Break causes **switch** to end and the program continues with the first statement after **switch** structure

Notice the **default** statement , which catches all other cases

Output:

```
A, B, C, D or F
Please Select Grade :A
Grade : A
Remark : Excellent!
```

Nested Switch Statement

A switch statement can be nested within another switch statement:

```
switch(count) {
              case 1:
                          switch(target) {
                                        case
0:System.out.println("target is zero");
                                       break;
                                       case
1:System.out.println("target is one");
                                       break;


          }
                          break;
             case 2: …
}
```

Since, every switch statement defines its own block, no conflict arises between the case constants in the inner and outer switch statements.

Comparing switch and if

Two main differences:

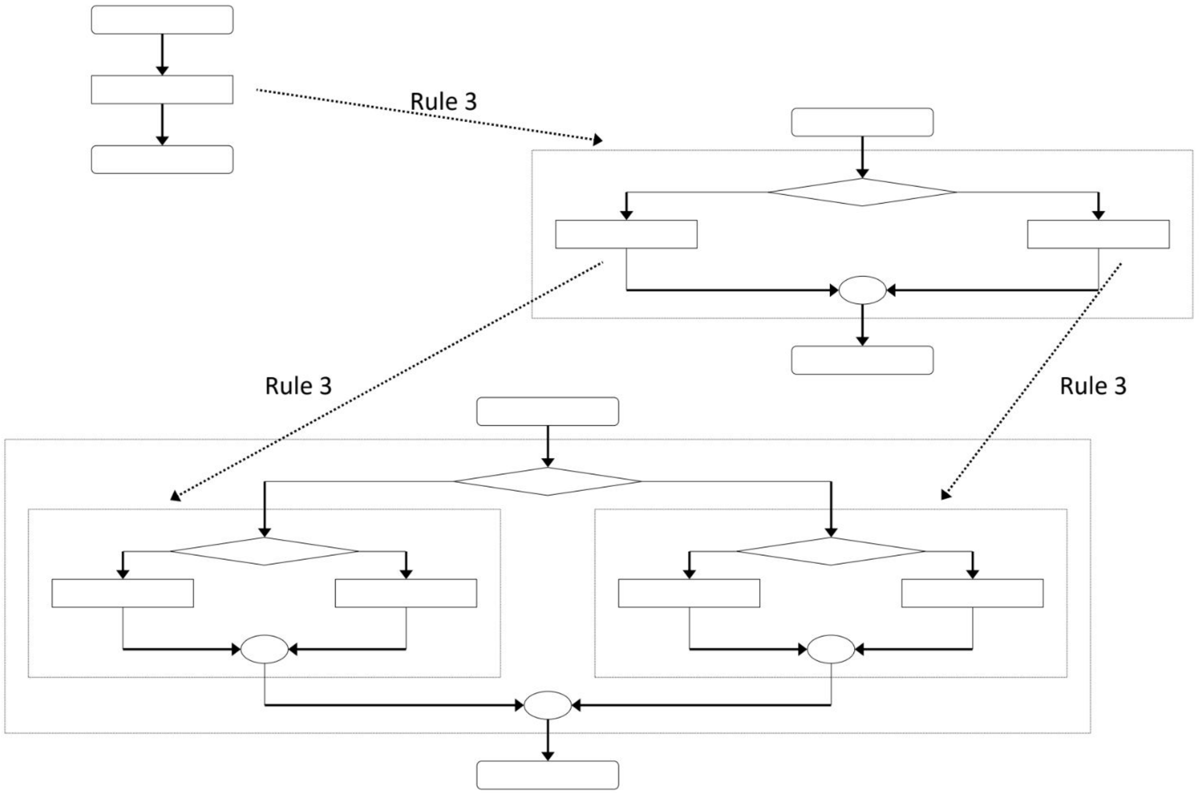
1) switch can only test for equality, while if can evaluate any kind of boolean expression

2) Java creates a “jump table” for switch expressions, so a switch statement is usually more efficient than a set of nested if statements

Structured-Programming Summary

1. **Begin with the "simplest flowchart"**: Starting with a straightforward design makes it easier to understand what the program is intended to do. You can then progressively add more complexity as needed.
2. **Any rectangle (action) can be replaced by two rectangles (actions) in sequence:** In Java, you can always replace a single statement with multiple statements. For example, you can replace System.out.println("Hello, World!"); with two statements like String message = "Hello, World!"; System.out.println(message);.
3. **Any rectangle (action) can be replaced by any control structure (sequence, if, if/else, switch, while, do/while or for)**: Java supports all these control structures, and you can use them to replace a single action in your code to handle more complex logic.
4. **Rules 2 and 3 can be applied in any order and multiple times**: You can start by applying one rule and then the other, and you can apply each rule multiple times to gradually build up the complexity of your program.
5. **Sequence**: Statements in Java are executed in sequence by default. Each line is executed one after the other in the order they appear in the code.
6. **Selection**: Java provides if, if/else, and switch statements for selection. You can certainly replace any switch or if/else construct with a series of if statements, though that may not always be the most readable or efficient way to do it.
7. **Repetition**: Java supports while, do/while, and for loops for repetition. It is true that any for or do/while loop can be rewritten using a while loop, though again, readability and efficiency may vary.

Representation of Rule 3 (replacing any rectangle with a control structure)



Questions?

Thank you for your attention!