**Suvarna Ballid Date:9/08/2023**

**Email:Suvarnaballid726@gmail.com**

**1). What is Difference Between Else if ladder and Switch?**

**1.1). Else if Ladder Statement**

**1.2). syntax of else if ladder**

**1.3). flowchart for else if ladder**

**1.4). PROGRAM For Else if ladder**

**1.5). Features of An else-if-ladder**

**1.6). Switch case statement**

**1.7). Syntax of switch case**

**1.8). Flowchart for switch case**

**1.9). PROGRAM for switch case**

**1.10). Features of Switch case**

**2)Nested Conditions**

**Nested if and Nested if Else**

**2.1)Is Nested Simple If is permited or not?**

**2.2).Is Nested If Else permited or not?**

**2.3).Is Nested If Else is permited?**

1.1).Else If Ladder:

else if statement can be defined as a control statement which controls the statement(s) to be executed on the basis of some conditions. Whenever the else if statement is used, the complier or interpreter initially checks the condition whether it is true or false and if the condition is found to be true then, the corresponding statements are executed. If the condition is found to be false, it continues checking the next else if statement until the condition comes to be true or the control comes to the end of the else if ladder.

1.2).The **syntax of else if ladder** can be represented as:

if( condition-1)

statement-1;

else if (condition-2)

statement-2;

else if (condition-3)

statement-3;

else if (condition-4)

statement-4;

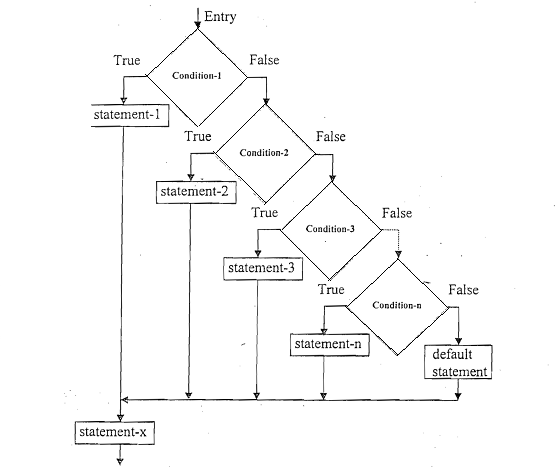
else if (condition-n)

statement-n;

else

default statement;

**1.3).flowchart for else if ladder**

[](https://www.codewithc.com/wp-content/uploads/2014/07/if.png)

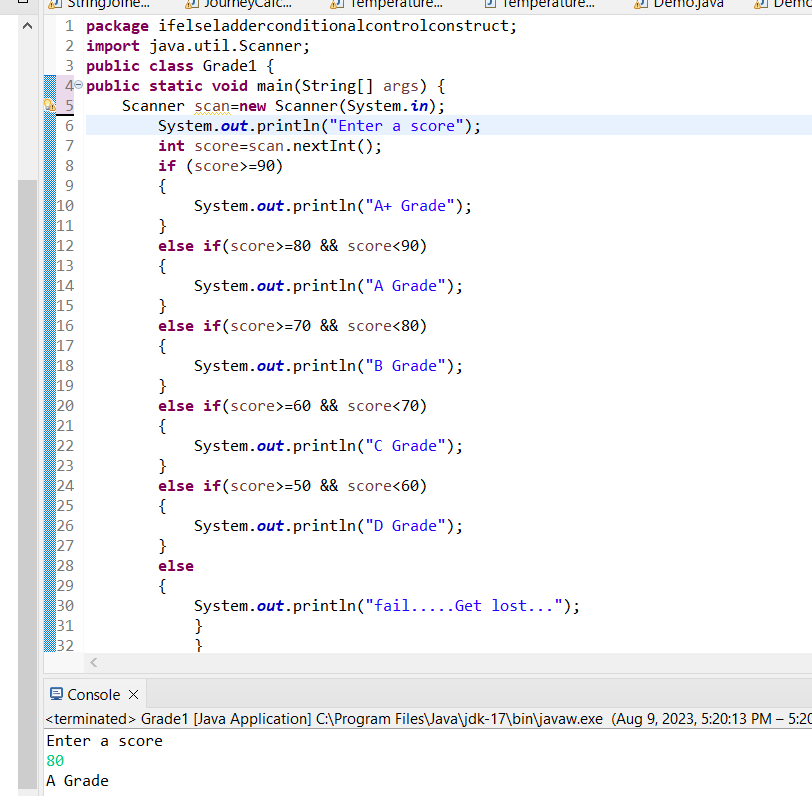
The “else-if ladder,” also known as the “if-else-if-else” chain, is another control flow construct used in programming languages to make multiple conditional checks in sequence. It allows you to evaluate a series of conditions, and if any of them are true, the corresponding code block is executed. If found to be false, it continues to check the next else if ladder statement until the condition comes to be true or the control comes to end the else if ladder statement.

In an else-if ladder, the program evaluates each condition in order. If a condition is true, the corresponding block of code executes, and the program exits the entire if-else structure. If none of the conditions are true, the code under the else block will be executed.

The else-if ladder is useful when you have multiple conditions to evaluate, and the order of evaluation matters. Each condition is checked in sequence, and the first true condition encountered determines the execution path.

If we have multiple blocks of statement and only one block of statement should be Executed at any given point of time, then we have to make use of the else if ladder conditional control construct.

**1.4).PROGRAM for else if ladder:**



1.5).Features of An else-if-ladder

* It evaluates an expression and then, the code is selected based on the true value of evaluated expression.
* Each else if has its own expression or condition to be evaluated.
* The variable [data](https://www.codewithc.com/data-visualization-java-project/) type used in the expression of else if is either integer or character.
* The decision making of the else if is dependent on zero or non-zero basis**.**
* Zero and non-zero basis is where the else if ladder depends for decision making.
* Either integer or character is the variable data type used in the expression of else if ladder.
* Each else if has its own expression or condition to be evaluated.
* Else if ladder evaluates an expression and then, the code is selected based on the true value of evaluated expression.

**1.6).Switch Case statement:**

The switch case statement is similar to the else-if ladder as it provides multiple branching or multi-conditional processing. But, the basic difference between switch case and else if ladder is that the switch case statement tests the value of variable or expression against a series of different cases or values, until a match is found. Then, the block of code with in the match case is executed. If there are no matches found, the optional default case is executed.

1.7).The **syntax of switch case** can be represented as:

switch(expression)

{

case constant-1

block-1;

break;

case constant-2

block-2;

break;

case constant-3

block-3;

break;

case constant-n

block-n;

break;

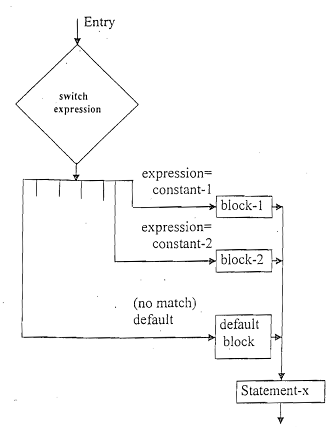
default:

default\_block;

}

statement-x;

**1.8).Flowchart for switch case**

[](https://www.codewithc.com/wp-content/uploads/2014/07/sw.png)

The “switch case” statement is a control flow mechanism found in many programming languages, including C, C++, Java, and others. It is particularly useful when you have multiple possible values for a single variable and want to execute different blocks of code based on the value of that variable. The syntax of a switch case looks like this:

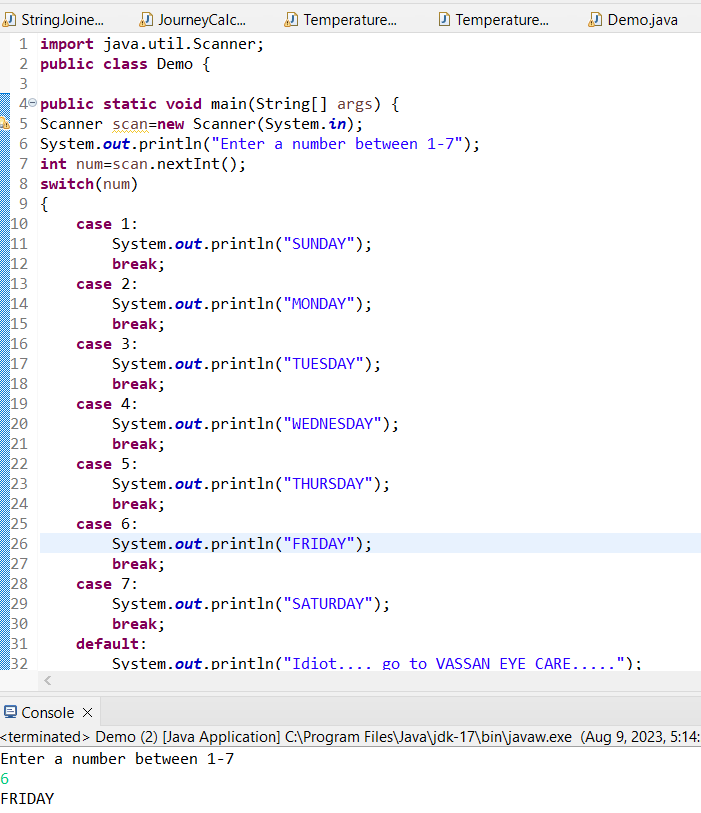
Here, the “variable” is the one whose value you want to check, and the “case” statements represent the possible values it can take. If “variable” matches a particular case, the corresponding code block is executed until it encounters a “break” statement or the end of the switch block.

The “default” case is optional and serves as the default code block to execute when none of the specified cases match the value of the variable.

Which one to use depends on the specific situation. Use a switch case when dealing with a single variable being compared to multiple constant values. On the other hand, use an else-if ladder when dealing with multiple independent conditions, and their order of evaluation is important.

If we have multiple options to choose and at any point in time only one block should be Executed then we have to make use of Switch condition control construct.

**1.9). PROGRAM:**



1.10).Features of Switch Case

* The switch case takes decision on the basis of equality.
* Each case has a break statement.
* Integer is the only data type that can be used in switch expression.
* Each switch case will always refer back to the original expression.
* The switch statement evaluates the value of an expression and a block code is selected on the basis of that evaluated expression.

**Difference Between Switch case and else if ladder**

So, after going through the two control statements in brief, here are the main difference between switch case and else if ladder:

1. In *else if ladder*, the control goes through the every else if statement until it finds true value of the statement or it comes to the end of the else if ladder. In case of *switch case*, as per the value of the switch, the control jumps to the corresponding case.
2. The switch case is more compact than lot of nested else if. So, switch is considered to be more readable.
3. The use of break statement in switch is essential but there is no need of use of break in else if ladder.
4. The variable data type that can be used in expression of switch is integer only where as in else if ladder accepts integer type as well as character.
5. Another difference between switch case and else if ladder is that the *switch* statement is considered to be less flexible than the else if ladder, because it allows only testing of a single expression against a list of discrete values.
6. Since the compiler is capable of optimizing the switch statement, they are generally considered to be more efficient. Each case in switch statement is independent of the previous one. In case of else if ladder, the code needs to be processed in the order determined by the programmer.
7. *Switch case statement* work on the basis of equality operator whereas *else if ladder* works on the basis of true false( zero/non-zero) basis.
8. In *else if ladder*, the control goes through the every else if statement until it finds true value of the statement or it comes to the end of the else if ladder. In case of *switch case*, as per the value of the switch, the control jumps to the corresponding case.
9. The switch case is more compact than lot of nested else if. So, switch is considered to be more readable.
10. The use of break statement in switch is essential but there is no need of use of break in else if ladder.
11. The variable data type that can be used in expression of switch is integer only where as in else if ladder accepts integer type as well as character.
12. Another difference between switch case and else if ladder is that the *switch* statement is considered to be less flexible than the else if ladder, because it allows only testing of a single expression against a list of discrete values.
13. Since the compiler is capable of optimizing the switch statement, they are generally considered to be more efficient. Each case in switch statement is independent of the previous one. In case of else if ladder, the code needs to be processed in the order determined by the programmer.
14. *Switch case statement* work on the basis of equality operator whereas *else if ladder* works on the basis of true false( zero/non-zero) basis.

If Else ladder Vs. Switch case statement in tabular form

|  |  |  |
| --- | --- | --- |
| **BASIS OF COMPARISON** | **ELSE IF LADDER** | **SWITCH CASE** |
| **The control** | In else if ladder, the control runs through the every else if statement until it arrives at the true value of the statement or until it comes to the end of the else if ladder. | In else if ladder, the control runs through the every else if statement until it arrives at the true value of the statement or until it comes to the end of the else if ladder. |
| **Working** | Else if ladder statement works on the basis of true false (zero/non-zero) basis. | Switch case statement work on the basis of equality operator. |
| **Use of Break Statement** | In switch, the use of break statement is mandatory and very important. | In else if ladder, the use of break statement is not very essential. |
| **Variable Data** | Integer is the only variable data type that can be in expression of switch. | Either integer or character is the variable data type used in the expression of else if ladder. |
| **Processing Of Codes** | In the case of else if ladder, the code needs to be processed in the order determined by the programmer. | In switch case, it is possible to optimize the switch statement, because of their efficiency. Each case in switch statement does not depend on the previous one. |
| **Flexibility** | Else if statement is not flexible because it does not give room for testing of a single expression against a list of discrete values. | Switch case statement is flexible because it gives room for testing of a single expression against a list of discrete values. |
| **Usage** | Else if ladder is used when there is multiple conditions are to be tested. | Switch case is used when there is only one condition and multiple values of the same are to be tested. |
| **Values** | Values are based on constraint. | Values are based on user choice. |

**Nested Simple If and Nested If Else**

**2.1).Is Nested Simple If is permited or not?**

Yes, it is permitted. But the inner if condition will not execute. Here is an example.

PROGRAM:

package Practice;

import java.util.\*;

public class NestedSimpleIf {

public static void main(String[] args) {

Scanner scan = new Scanner(System.***in***);

System.***out***.println("Enter the time");

int i=scan.nextInt();

if (i>9)

{

System.***out***.println("You are late,so you are not allowed into the school");

if (i<9)

{

System.***out***.println("Your allowed into the school");

}

scan.close();

}

}

}

OUTPUT:

Enter the time

10

You are late,so you are not allowed into the school

Enter the time

8

**2.2).Is Nested If Else permited or not?**

Yes, it is permited.

PROGRAM:

package Practice;

import java.util.\*;

public class NestedIfElse {

public static void main(String[] args) {

Scanner scan = new Scanner(System.***in***);

System.***out***.println("Enter Marks");

int i = scan.nextInt();

if(i<35)

{

System.***out***.println("You are not eligible for next class");

System.***out***.println("Enter a number");

int j = scan.nextInt();

if(j%2==0)

{

System.***out***.println("Even number");

}

else

{

System.***out***.println("Odd number");

}

}

else

{

System.***out***.println("You are eligible for next class");

}

scan.close();

}

}

OUTPUT:

Enter Marks

33

You are not eligible for next class

Enter a number

66

Even number

Enter Marks

27

You are not eligible for next class

Enter a number

13

Odd number

Enter Marks

72

You are eligible for next class

**2.3).Is Nested If Else is permited?**

Yes, it is permited.

PROGRAM:

package Practice;

import java.util.Scanner;

public class NestedIfElse2 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.***in***);

System.***out***.println("Enter Marks");

int i = scan.nextInt();

if(i<35)

{

System.***out***.println("You are not eligible for next class");

}

else

{

System.***out***.println("You are eligible for next class");

System.***out***.println("Enter a number");

int j = scan.nextInt();

if(j%2==0)

{

System.***out***.println("Even number");

}

else

{

System.***out***.println("Odd number");

}

}

scan.close();

}

}

OUTPUT:

Enter Marks

27

You are not eligible for next class

Enter Marks

40

You are eligible for next class

Enter a number

22

Even number

Enter Marks

45

You are eligible for next class

Enter a number

43

Odd number