OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

JAVA'S SELECTION STATEMENTS

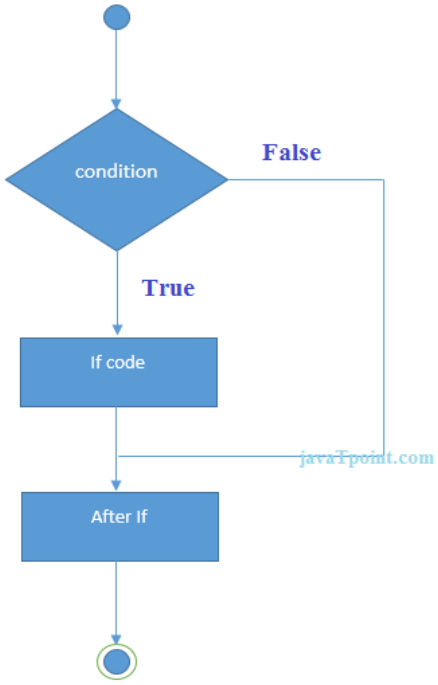
- if
- if-else
- if-else-if ladder
- nested if
- switch

IF STATEMENT

- The if Statement : The Java if statement tests the condition.
- It executes the if block if condition is true.

- Syntax:

```
if(condition) {
        // statement;
}
```



- Example:

```
if(num < 100)
  System.out.println("less than 100");
```

EXAMPLE

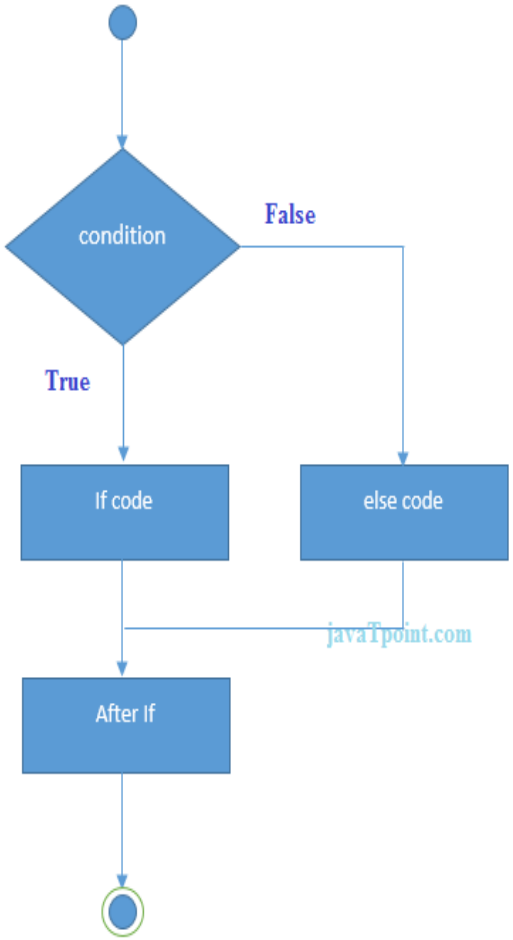
//Java Program to demonstrate the use of if statement.

```
public class IfExample {
        public static void main(String[] args) {
            //defining an 'age' variable
            int age=20;
            //checking the age
            if(age>18){
                System.out.print("Eligible for voting");
            } } }
```

IF ELSE STATEMENT

- The Java if-else statement also tests the condition.
- It executes the if block if condition is true otherwise else block is executed.
- Syntax:

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



EXAMPLE

```
//It is a program of odd and even number.


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

                int number=13;

                //Check if the number is divisible by 2 or not

                  if(number%2==0){ System.out.println("even
                  number");
                }

                else {

                        System.out.println("odd number");
                } } }
```

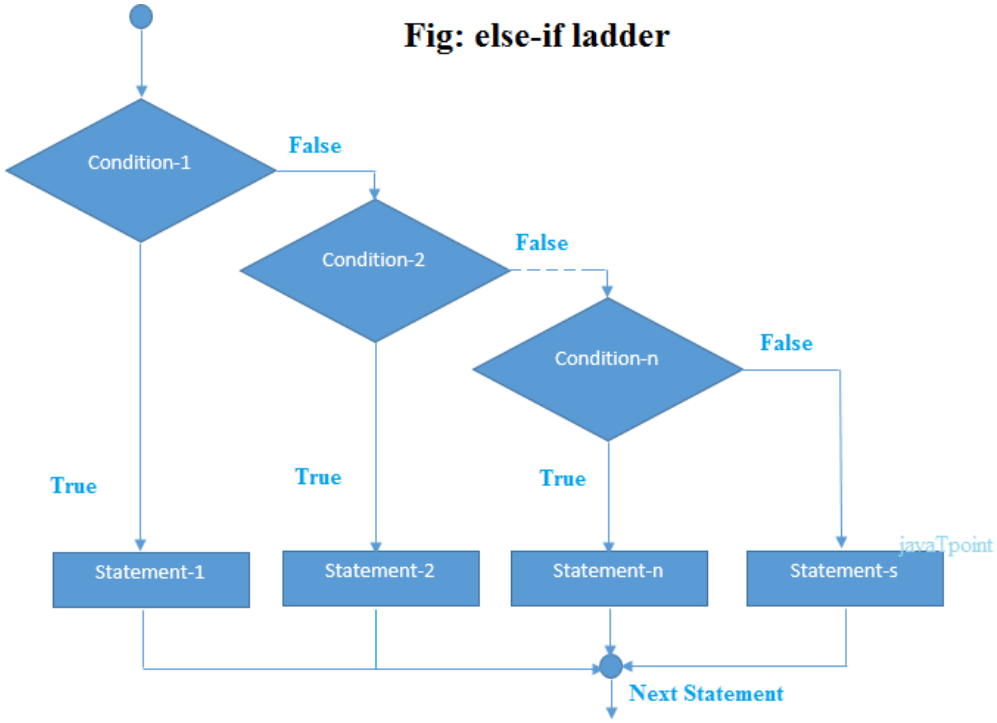
EXAMPLE

```
//A year is leap, if it is divisible by 4 and 400. But, not by 100.

public class LeapYearEx {
  public static void main(String[] args) {
        int year=2020;
        if(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)) {
            System.out.println("LEAP YEAR");
        }
        else {
            System.out.println("COMMON YEAR");
        } } }
```

IF ELSE IF LADDER

- The if-else-if ladder statement executes one condition from multiple statements.
- Syntax:

```
if(condition1){
   //code to be executed if condition1 is true
}else if(condition2){
   //code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
   ...
else{
   //code to be executed if all the conditions are false }
```



EXAMPLE

```
//Find out given no is +ve or –ve or zero

public class PositiveNegativeExample {
public static void main(String[] args) {
  int number=-13;

  if(number>0){
            System.out.println("POSITIVE");
     }else if(number<0){ System.out.println("NEGATIVE");
  }else{
            System.out.println("ZERO");
}
}
}
```

EXAMPLE

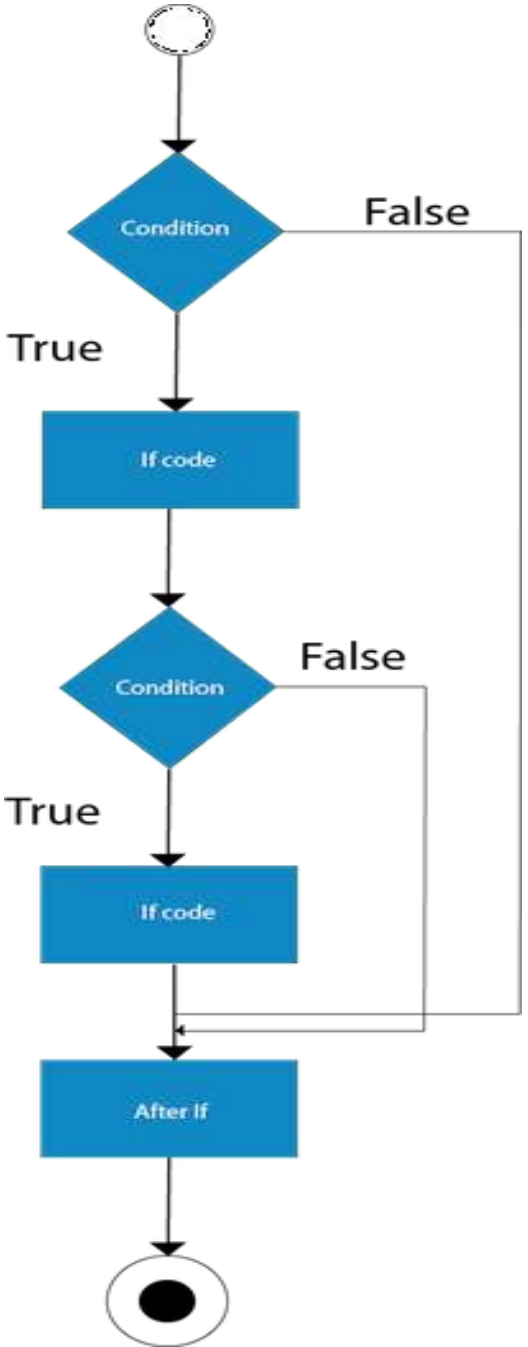
```
//It is a program of grading system for fail, D grade, C grade, B grade, A grade and A+.

public class IfElseIfExample {
public static void main(String[] args) {
        int marks=65;
        if(marks<35){
            System.out.println("fail");
        }
            else if(marks>=35 && marks<60){ System.out.println("D grade");
        }
        else if(marks>=60 && marks<70){
            System.out.println("C grade");
        }
        else if(marks>=70 && marks<80){
            System.out.println("B grade");
        }
        else if(marks>=80 && marks<90){
            System.out.println("A grade");
           }else if(marks>=90 && marks<100){ System.out.println("A+ grade");
        }else{
            System.out.println("Invalid!");
        }}}
```

NESTED IF

- The nested if statement represents the if block within another if block.
- Here, the inner if block condition executes only when outer if block condition is true.

- Syntax:

```
if(condition)
{
   //code to be executed
        if(condition){
           //code to be executed
        }
}
```



EXAMPLE

```
//Java Program to demonstrate the use of Nested If Statement.


public class JavaNestedIfExample {

public static void main(String[] args) {
 int age=20;

 int weight=80;

 if(age>=18){

            if(weight>50){
                        System.out.println("You are eligible to donate blood");}

            else{

                        System.out.println("You are not eligible to donate blood");}  }

 else{

            System.out.println("Age must be greater than 18");
}}}
```

SWITCH STATEMENT

- The Java switch statement executes one statement from multiple conditions.
- It is like if-else-if ladder statement.
- The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long.
- There can be one or N number of case values for a switch expression.
- The case value must be of switch expression type only. The case value must be *literal or constant*. It doesn't allow variables.
- The case values must be *unique*. In case of duplicate value, it renders compile-time error.
- Each case statement can have a *break statement* which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.
- The case value can have a *default label* which is optional.

SWITCH

```
switch(expression){
case value1:
        //code to be executed;
        break; //optional
case value2:
        //code to be executed;
        break; //optional
        ......

default:
        code to be executed if a
}
```

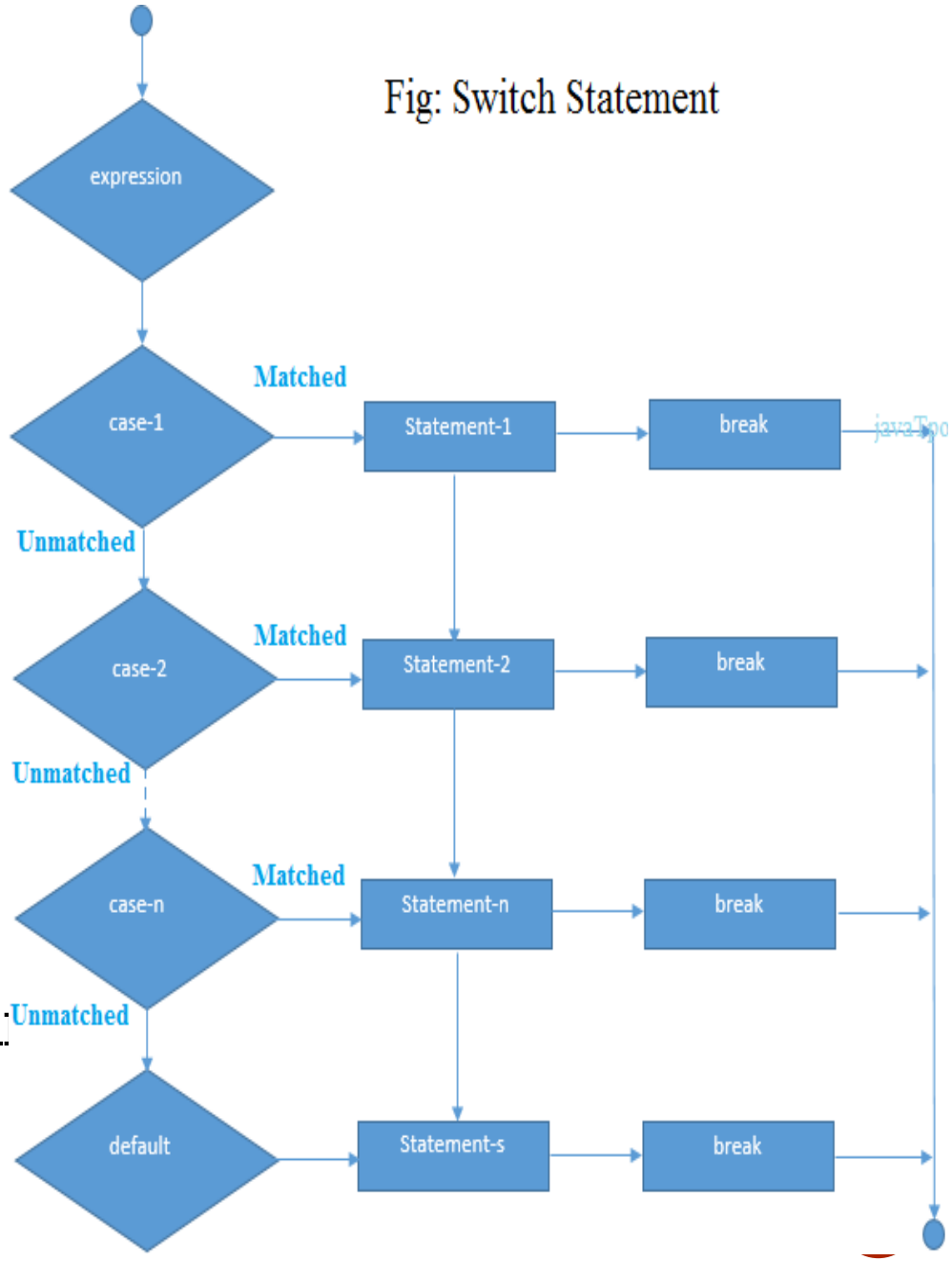


Fig: Switch Statement

EXAMPLE

```
public class SwitchExample {
public static void main(String[] args) {
    int number=2;

  //Switch expression
  switch(number){
                                        //Case statements
                case 1: System.out.println(“ONE");
                        break;
                case 2: System.out.println(“TWO");
                        break;
                case 3: System.out.println(“THREE");
                        break;
                                        //Default case statement
                default:System.out.println("Not in 1, 2 or 3");
  }}}
```

```
//Name of the month

public class SwitchMonthExample {
public static void main(String[]
args) {
   //Specifying month number
   int month=7;
   String monthString="";
   //Switch statement
   switch(month){
  //case statements within the switch block

   case 1: monthString="1 -
   January";
            break;
   case 2: monthString="2 -
   February";
            break;
   case 3: monthString="3 - March";
            break;
   case 4: monthString="4 - April";
            break;
   case 5: monthString="5 - May";
            break;
   case 6: monthString="6 - June";
            break;
   case 7: monthString="7 - July";
            break;
   case 8: monthString="8 - August";
            break;
   case 9: monthString="9 - September";
            break;
   case 10: monthString="10 - October";
            break;
   case 11: monthString="11 -
   November";
            break;
   case 12: monthString="12 -
   December";
   default: System.out.println("Invalid
   Month!");
   }
   //Printing month of the given number
   System.out.println(monthString);
}
}
```

SWITCH W/O BREAK

It executes all statements after the first match, if a break statement is not present.

```
//without break statement
public class SwitchExample2 {
 public static void main(String[] args) {
     int num=2;
     //switch expression with int value
     switch(num){
     //switch cases without break statements
case 1: System.out.println("1");
case 2: System.out.println("2");
case 3: System.out.println("3");
default:System.out.println("Not in 1, 2 or 3");
}}}
```

Output:

2
3
Not in 1, 2 or 3

EXAMPLE

```
// In a switch, break statements are optional. class MissingBreak {
public static void main(String args[]) {
      for(int i=0; i<12; i++)
            switch(i) {
                  case 0:
                  case 1:
                  case 2:
                  case 3:
                  case 4:
                        System.out.println("i is less than 5"); break;
                  case 5:
                  case 6:
                  case 7:
                  case 8:
                  case 9:
                        System.out.println("i is less than 10"); break;
                  default:
                        System.out.println("i is 10 or more");
            }}}
```

NESTED SWITCH STATEMENT

- We can use switch statement inside other switch statement in Java.
- It is known as nested switch statement.
- Syntax:

```
switch(expression1){
        case value1:
        //code to be executed;
        break;
        case value2:
                switch(expression2){
                        case value21:
                        //code to be executed;
                        break;
                        case value22:
                        break;
                        default:
                        //default code to be executed ; }
        break;
        default:

      //code to be executed if all cases are not matched; }
```

ASSIGNMENT:

You are searching for a department in a university and you're asked to select a school from a choice of three schools namely:

School of Arts
School of Business
School of Engineering

Having selected a school you are again provided with a list of departments that fall under the department namely:

School of Arts
- Department of finearts
- Department of sketcharts

School of Business
- Department of Commerce
- Department of purchasing

School of Engineering
- CSE
- ECE

The initial choices for Computer Science, Business and Engineering schools would be inside as a set of switch statements. Then various departments would then be listed within inner switch statements beneath their respective schools.