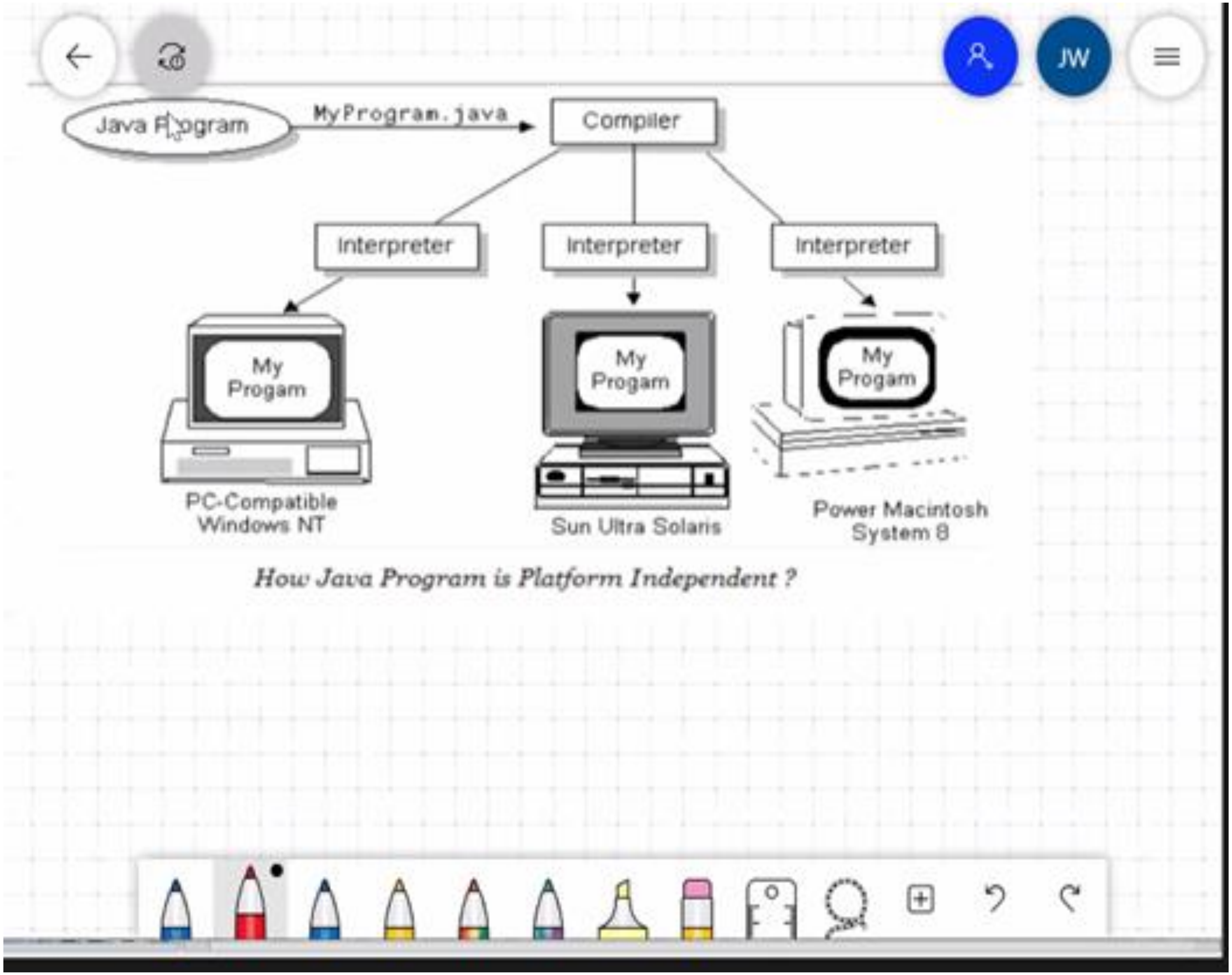
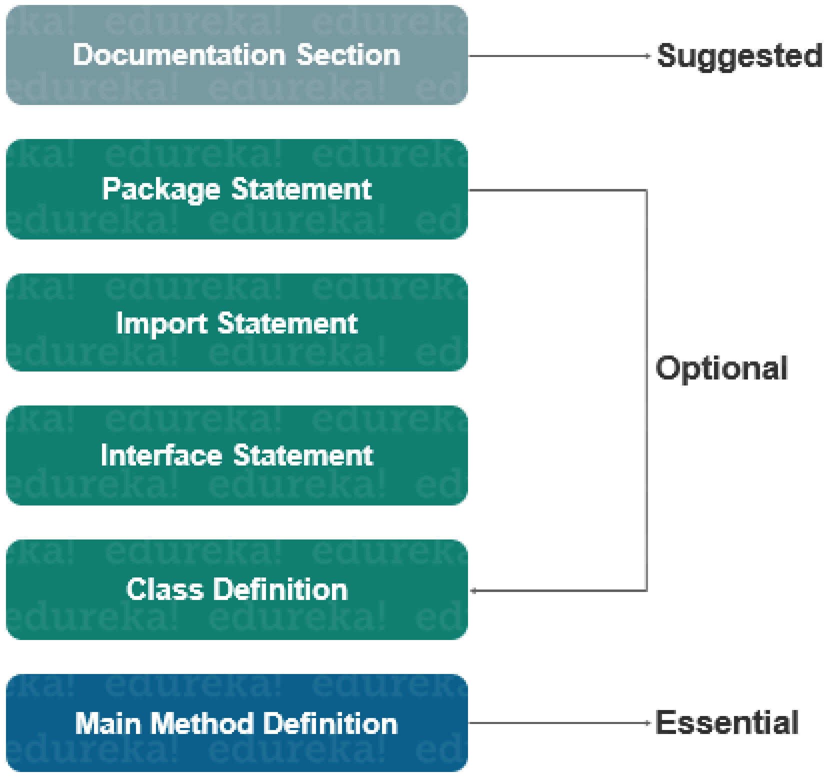
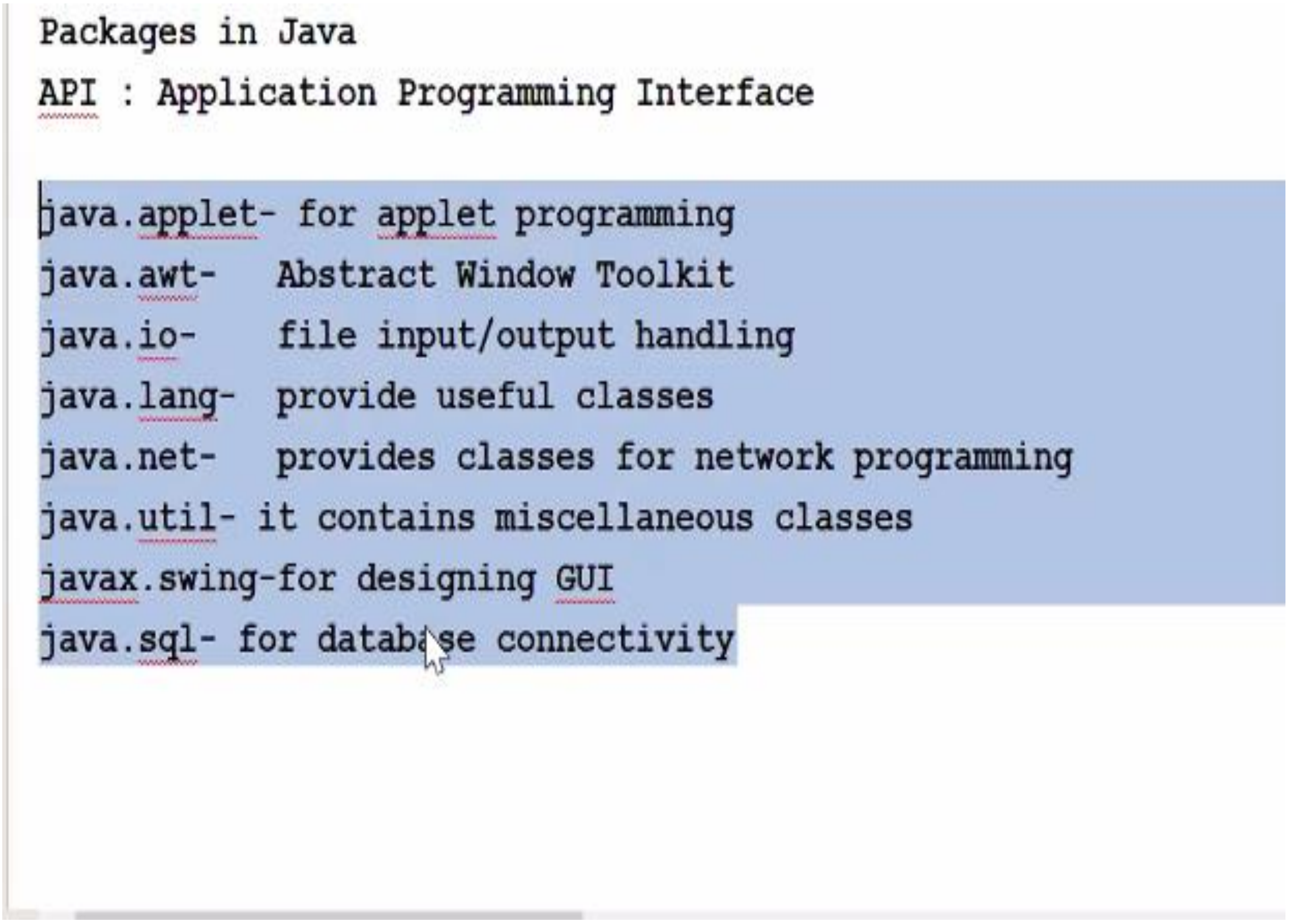


How Java Program is Platform Independent ?

Program Structure





Packages in Java

API : Application Programming Interface

java.applet- for applet programming

java.awt- Abstract Window Toolkit

java.io- file input/output handling

java.lang- provide useful classes

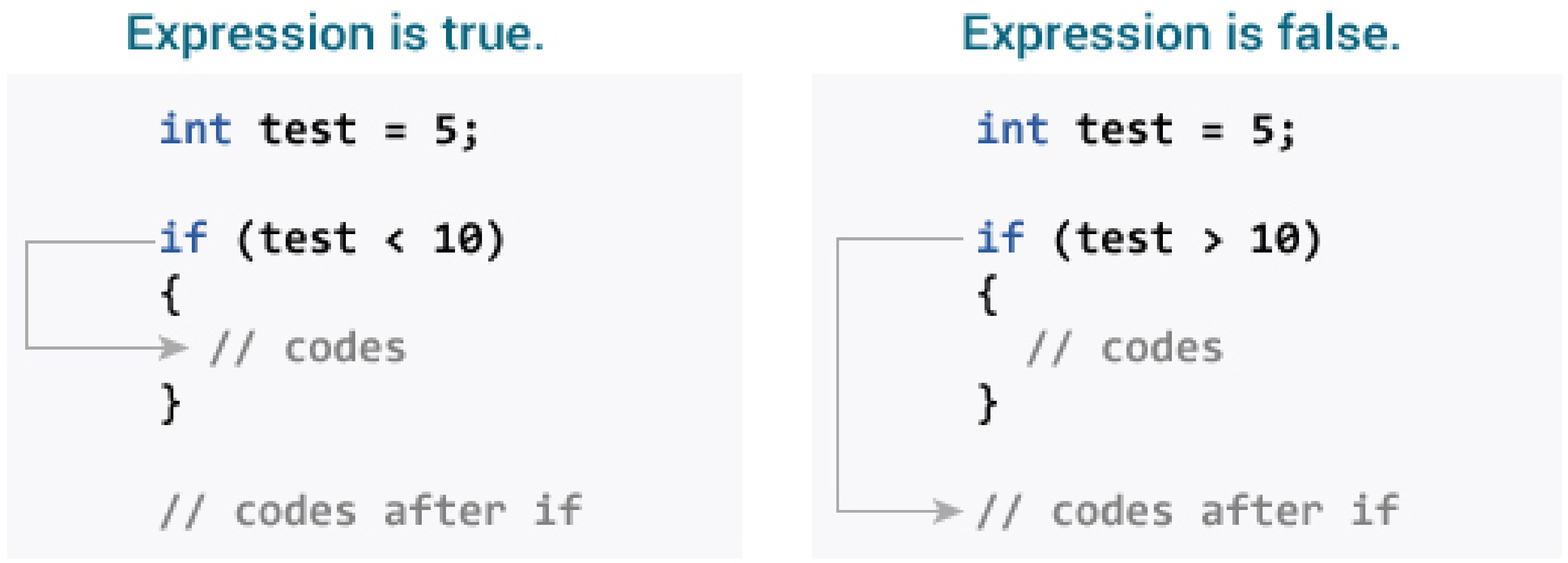
java.net- provides classes for network programming

java.util- it contains miscellaneous classes

javax.swing-for designing GUI

java.sql- for database connectivity

- if (expression)
- {
- // statements
- }



```
int test = 5;

if (test < 10)
{
  // codes
}

// codes after if
```

```
int test = 5;

if (test > 10)
{
  // codes
}

// codes after if
```

If - else

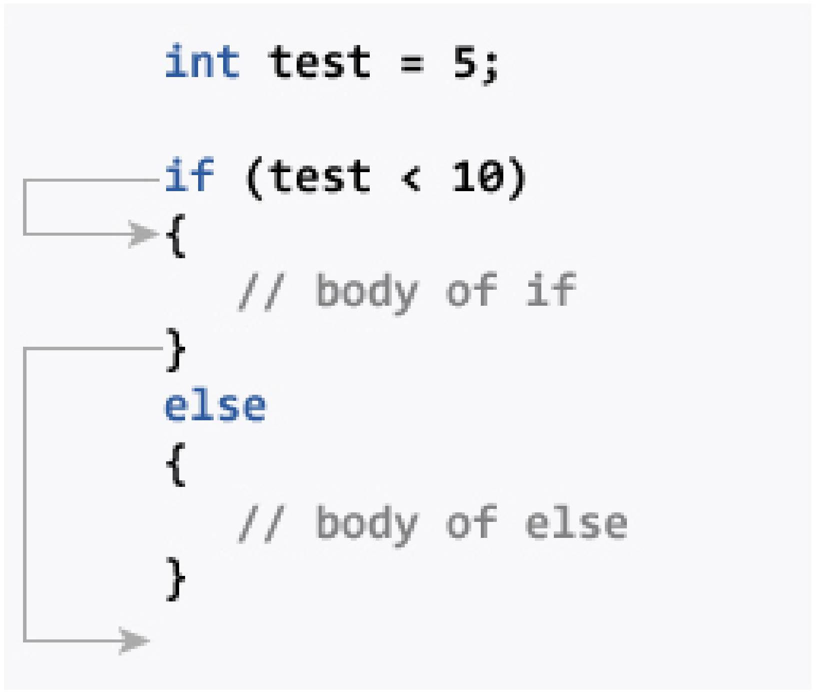
- class IfStatement
- {
- public static void main(String[] args)
- {
- int number = 10;
- {
- System.out.println("The number is positive."); }
- **Output**:
- The number is positive.

- class Main
- {
- public static void main(String[] args)
- {
- String language = "Java";

- if(language == "Java")
- {
- System.out.println("This is best programming language.");
- }
- }
- }

- if (expression)
- {
- // codes
- }
- else
- {
- // some other code
- }

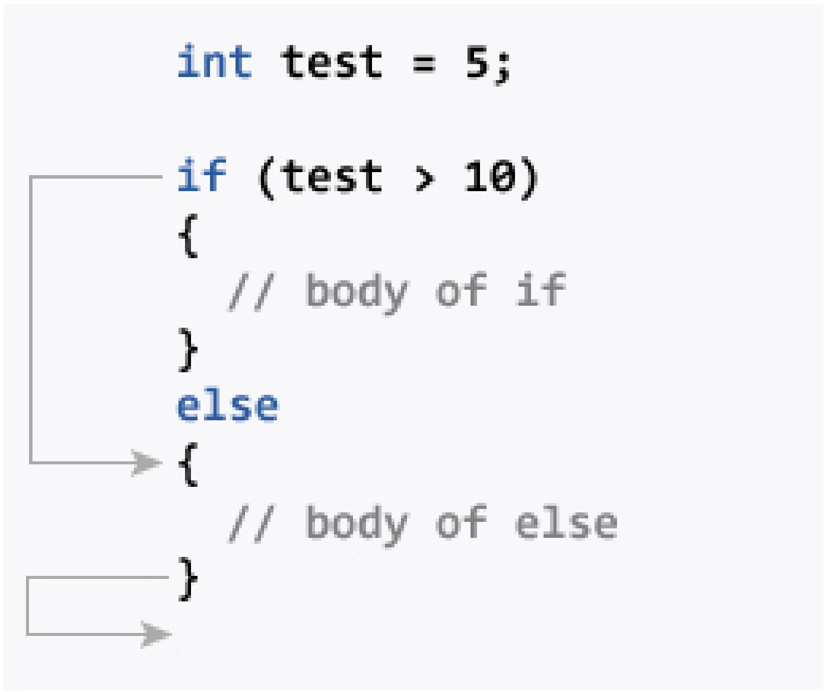
Expression is true.



```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

Expression is false.



```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

- class IfElse
- {
- public static void main(String[] args)
- {
- int number = 10;
- if (number > 0)
- {
- System.out.println("The number is positive.");
- }
- else
- {
- System.out.println("The number is not positive.");
- } } }
- **Output**:
- The number is positive.

switch

- switch (variable/expression)
- {
- case value1: // statements of case1
- break;
- case value2: // statements of case2
- break; .. ..
- default: // default statements
- }

Facts about switch

- Expression can only be **char, byte, short, int, Character, Byte, Short, Integer, String, or an enum type** other wise a **compile-time error** occurs.
- According to the specification followings are also must be true:
- No two of the case constant expressions associated with a switch statement may have the same value.
- No switch label is null.
- switch expression can't be float, double or boolean.
- boolean true false are meaningful using with if-else,
  e.g., if(true) then do.
- Floating point numbers (float, double) are not a good candiadtes for switch as exact comparison is often broken by rounding errors. e.g. 0.11 - 0.1 == 0.01 is false.
-

Condition Operator

- Conditional Operator (? :)
- variable x = (expression) ? value if true : value if false

Condition Operator

- Also known as the ternary operator
- **Consists of 3 operands**
- Evaluate Boolean expressions.
- One-liner replacement for if-else ,switch
- Same logic but less space in single line

- If else
- if (expression) //check true or false
- {
- number = 10;
- }
- else {
- number = -10;
- }

- Condition operator
- number = (expression) ? expressionTrue : expressionFalse

- class Operator
- {
- public static void main(String[] args)
- {
- Double number = -5.5;
- String result;
- result = (number > 0.0) ? "positive" : "not positive"; System.out.println(number + " is " + result);
- }
- }
- Output will be: -5.5 is not positive

```
• class Ternary
• {
•     public static void main(String[] args)
•     {
•         int n1 = 5, n2 = 10, max;
•         System.out.println("First num: " + n1);
•         System.out.println("Second num: " + n2);
•             max = (n1 > n2) ? n1 : n2;
•           System.out.println("Maximum is = " + max);
•     }
• }
```

- if (expression1)
- {
- result = 1;
- }
- else if (expression2)
- {
- result = 2;
- }
- else if (expression3)
- {
- result = 3;
- }
- else {
- result = 0;
- }

- class Main
- {
- public static void main(String[] args)
- {
- int week = 4;
- String day; // switch statement to check day

- switch (week)
- {
- case 1: day = "Sunday";
- break;
- case 2: day = "Monday";
- break;
- case 3: day = "Tuesday";
- break;
- case 4:
- day = "Wednesday";
- break;
- case 5: day = "Thursday";
- break;
- case 6: day = "Friday";
- break;
- case 7: day = "Saturday";
- break;
- default: day = "Invalid day";
- break;
- }
- System.out.println("The day is " + day); } }

Scanner

- A class in java.util package
- To obtain input of the primitive types
- int, double,char, Strings.
- To create an object of Scanner class,
- Pass predefine object System.in/(File),
- Scanner in = new Scanner(System.in);
-

To read character

- To read a single character, we use next().charAt(0).
- next() function returns the next token/word in the input as a string and charAt(0) function returns the first character in that string.
-

- public byte nextByte(): Scans next token as byte value
- public short nextShort(): Scans next token as short value
- public int nextInt(): Scans next token as integer value
- public long nextLong(): Scans next token as long value
- public float nextFloat(): Scans next token as float value
- public double nextDouble(): Scans next token as double value
- void close(): Scanner is closed
-

- public String nextLine() :
- public String next(): Returns the token before delimiter

- **next() Method:** The *next()* method in java is present in the Scanner class and is used to get the input from the user.
- This method can read the input only until a space(” “) is encountered.
-

- **nextLine() Method**:
- The *nextLine()* method in java is present in the Scanner class and is used to get the input from the user.
- In order to use this method, a Scanner object needs to be created.
- This method take input until the line change or new line ends input of getting ‘\n’ or press enter.
-

NEXT()	NEXTLINE()
It read input from the input device till the space character.	It read input from the input device till the line change.
It cannot read those words having space in it.	It can read those words having space in it.
It ends reading the input after getting space.	It ends reading the input after getting '\n' or press enter.
It places the cursor in the same line after reading the input.	It places the cursor in the next line after reading the input.
The escaping sequence of next() is space.	The escaping sequence of nextLine() is '\n'.
Syntax to scan input: Scanner.next()	Syntax to scan input: Scanner.nextLine()

Next()

```
• import java.util.Scanner;
• class Demo
• {
•     public static void main(String[] args)
•     {
•         // Creating the Scanner object
•         Scanner sc = new Scanner(System.in);
•
•         // Use of the next() method
•         String s = sc.next();
•         System.out.println(s);
•     }
  }
```

- i/p: hello world
- o/p : hello

nextLine()

- import java.util.Scanner;
- class Demo
- {
-     public static void main(String[] args)
-     {
-         // Creating the Scanner object
-         Scanner sc = new Scanner(System.in);
-
-         // Use of the next() method
-         String s = sc.nextLine();
-         System.out.println(s);
-     }

  }

3 different ways of taking input in java

- 1.Using Buffered Reader Class
- 2.Using Scanner class
- 3.Using Console