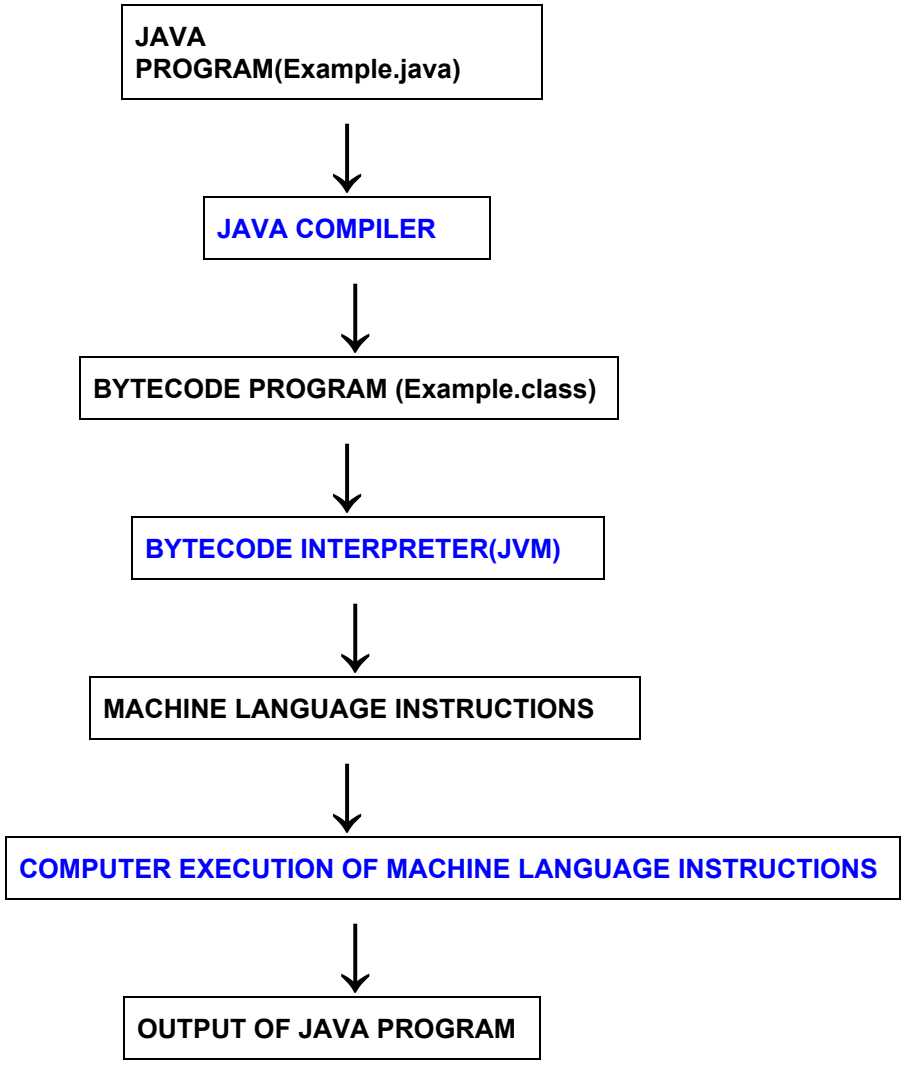
Java Bytecode

The Java compiler does not translate the program into the machine language for your particular computer.It translates into bytecode. Bytecode is not the machine language for any particular computer. Instead it is a machine language for a hypothetical computer known as virtual machine. Translating a program written in bytecode into a machine language program for an actual computer is quite easy.The program that does this translation is a kind of interpreter called the Java Virtual Machine or JVM. The JVM translates and runs the bytecode.

2 Steps for running a java program:

a. Compile - Converts the java program into bytecode
b. Run - The bytecode is converted into machine language instructions of the computer and executed.



IDE for Java : We can use any IDE (Integrated development environment) like BlueJ, Eclipse, JGrasp and NetBeans for writing, compiling and running a Java program.

Java program can also be written using simple text editors like Notepad (in windows) and TextEdit(in Mac). To then compile the java program we can use the free Java system distributed by Sun Microsystems for Windows, Linux, and Solaris.

To compile use the following command on command prompt:

```
javac   example.java
```

- Note - Example can be replaced by any program name

To run a Java program, use the following command:

```
java example
```

- Note that there is no extension used . We only us the name of the class

Java program should be saved with the name of the class which consists the void main()

Automatic Type Conversion: When two operands of different data types are encountered in the same expression, the variable of lower data type is automatically converted to the data type of variable with higher data type, and then the expression is calculated.

For Example:

```
int a=98;
float b=5;
System.out.println(a/3.0);  //converts a temporarily to float type, since 3.0 is of float
                            // type
System.out.println(a/b); //converts a temporarily to float type, since b is of float type,
                         //and gives the result 19.6
```

Of all the types used in the expression , it is the one that appears rightmost in the following list:

byte → short→ int → long → float → double

Type Casting: Type casting refers to the data type conversions specified by the programmer, as opposed to the automatic type conversions. This can be done when the compiler does not do the conversions automatically.Type casting can be done to higher or lower data type.

For Example: System.out.println((float)12/5); // displays 2.4 , since 12 is converted to //float type

Compound Statement: More than one statement to be executed in one go, is known as a compound statement.

Relational Operators: Relational operators are used to relate two values/variables. The relational operators present in Java are:

Operator	Usage	Example	Explanation
<	Less than	A<B	A is less than B
>	Greater than	A>B	A is greater than B
<=	Less than or equal to	A<=B	A is less than or equal to B
>=	Greater than or equal to	A>=B	A is greater than or equal to B
==	Equal to	A==B	A is equal to B
!=	Not equal to	A!=B	A is not equal to B

Compound Condition: It is a combination of two conditions using logical operator &&(And) or ||(Or).

Logical Operators: Two conditions can be logically combined with the help of logical operators. The logical operators present in Java are:

Operator		Usage	Example
&&	**AND**	The compound condition evaluates to true, if both the conditions in the compound condition evaluate to true.	((a>b) && (a>c))
||	**OR**	The compound condition evaluates to true, if any or both the conditions in the compound condition evaluate to true.	((a>b) || (a>c))
!	**NOT**	It negates the condition. That is: ● If the condition evaluates to true, it makes it false. ● If the condition evaluates to false, it makes it true.	!(a>b)

Conditional Constructs: They help us to execute a set of statements based on a condition. There are two conditional constructs present in Java.

- **if…else construct**
- **switch…case construct**

Note: If there is only one statement associated with a case, then the curly braces are optional.
Note: e*lse* is always optional with a particular *if*.
Different syntaxes that can be used with if…else construct:

Syntax	Example

if(<condition>)
  statement;
	if(a>b)
	  System.out.println(a);

if(<condition>)
{
  statement 1;
  statement 2;
  .
  .
  .
  statement n;
}
	if(a>b)
	{
	System.out.println(”diff=” + (a-b));
	System.out.println(”quotient=” + (a/b));
	}

```
if(<condition>)
  statement;
else
  statement;
```

```
if(a>b)
   System.out.println(a);
else
   System.out.println(b);
```

```
if(<condition>)
{
        statement 1;
        statement 2;
        .
        .
        .
        statement n;
}
else
{
        statement 1;
        statement 2;
        .
        .
        .
        statement n;
}
```

```
if(a>b)
{
  System.out.println("diff=" + (a-b));
  System.out.println("quotient=" + (a/b));
}
else
{
  System.out.println("diff=" + (b-a));
  System.out.println("quotient=" + (b/a));
}
```

Nested if…else

a) if…else Ladder *(The last else is optional)*

```
if(<condition1>)
{
        statement 1;
        .
        .
}
else if(<condition2>)
{
```

```
if((a>=b)&& (a>=c))
   System.out.println("largest=" + a);
else if((b>=a)&& (b>=c))
   System.out.println("largest=" + b);
else
   System.out.println("largest=" + c);
```

```
        statement 1;
        .
        .
}
.
.
else
{
        statement 1;
        .
}
```

b)

```
if(<condition1>) //one
    if(<condition2>) //two
    .
    .
    .
      if(<condition n>) //n
      {
            statement 1;
            .
            .
      }
      else //n
      {
            statement 1;
            .
            .
      }
    else //two
    {
          statement 1;
          .
          .
    }
else //one
{
        statement 1;
        .
        .
}
```

This can be nested for any number of conditions.

```
if(per>=50) //one
  if(per>=60) //two
    if(per>=70) //three
      if(per>=80) //four
        if(per>=90)//five
          grade='A';
        else //five
          grade='B';
      else //four
        grade='C';
    else //three
      grade='D';
  else //two
    grade='E';
else //one
  grade='f';
```

Backslash character constants: Java allows certain non-graphic characters i.e. characters that cannot be typed directly from the keyboard. The following are some backslash character constants:

Backslash character constant	Usage
\n	Escape sequence for new line character
\t	Escape sequence for tab(eight spaces)
\"	Escape sequence for Double quote
\'	Escape sequence for Single quote
****	Escape sequence for Backslash character
\r	Escape sequence for carriage return. Go to beginning of the current line

These are called **escape sequences,** since the backslash causes an escape from the normal way characters are interpreted. They can be used in both character and string constants.

For eg.: `System.out.println(`"Run\nspot" + "\n" + "run" +'\n');