

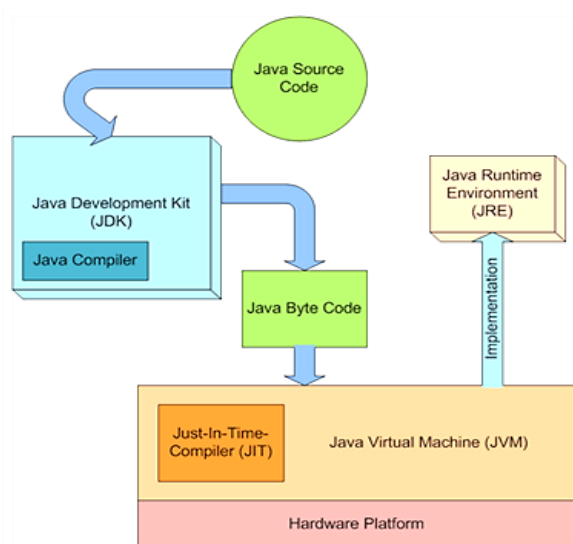
### UNIT-1 Introduction to java and elementary programming

Java language specification API, JDK and IDE, Creating, compiling and Executing a simple java program, Programming style, documentation and errors, Reading input from console, identifiers and variables, Assignment statements, Named constants and naming conventions, Data Types (Numeric, Boolean, Character, String) its Operations and Literals, Evaluating Expressions and operator Precedence, Types of Operators (Augmented assignment, Increment and Decrement, Logical), operator precedence and associativity, numeric type conversions.

#### 1. Define the terms- Java Language Specification, API, IDE

- **Java Language Specification**
- The Java language specification is a technical definition of the language that includes the semantics and syntax of the Java programming language.
- **API**
- Java Application Programming Interface (API) is the domain for the Java Developer Kit (JDK).
- An API contains classes, interfaces, packages, and their methods, fields, and constructors.
- All these built-in classes provide benefits to the programmer.
- API functions as an interface which provides the connection between the software as well as the programmer.
- API provides classes and packages which usually helps a programmer in minimizing the code.
- **IDE**
- An IDE (integrated development environment) is a software application containing all the features and tools required by software developers.
- Examples of IDEs include NetBeans, Eclipse, and IntelliJ.
- The programmer can use a Java development tool for rapidly developing the Java application programs.
- Compiling, editing, debugging, building, and online help are integrated into one graphical user interface.

#### 2. Explain JRE, JDK, JVM and JIT



## Unit 1: Introduction To Java And Elementary Programming

- **Java Development Kit (JDK)** is responsible for the development purpose. It contains various development tools like Java libraries, compilers, debuggers, bundling and deployment tools.  $JDK = JRE + \text{other development tools}$ .
- **Java Runtime Environment (JRE)** is an implementation of the JVM.  $JRE = JVM + \text{other class libraries}$ .
- **Java Virtual Machine (JVM)** is an abstract computing machine. JVM becomes an instance of JRE at runtime of a Java program. It is widely known as a runtime interpreter.
- **Just In Time Compiler (JIT)** is an integral part of the JVM. It runs after the program has started executing, on the fly. It has access to runtime information and makes optimizations of the code for better performance.



### 3. Explain role of JVM.

- JVM take care of creating of class file of java program and the same class file can be used in different operating system.
- JVM acts as an interface between your operating system and your program code.
- It also provide different security related features.
- JVM responsible for calling main method and running your program.
- It is also responsible for calling garbage collector for clean-up operations.

### 4. JVM is platform dependent. Justify.

- JVM is a virtual machine or a program that provides run-time environment in which java byte code can be executed.
- JVM needs to be installed on your machine depending on what platform do you have and what version of it.
- JVMs are available for many hardware and software platforms.
- JVM depends on the operating system – so if you are running Mac OS you will have a different JVM than if you are running Windows or some other operating system.
- JVM, JRE and JDK are platform dependent because configuration of each OS differs.

### 5. Define: 1) Byte code 2) Unicode

#### 1) Byte code

- Java bytecode is the result of the compilation of a Java program, an intermediate representation of that program which is machine independent.
- It is the job of the JVM to make the necessary resource calls to the processor in order to run the bytecode.

#### 2) Unicode

- Unicode is a 16-bit character encoding standard and is capable to represent almost every character of well-known languages of the world.
- As Java was developed for multilingual languages it adopted the unicode system. So lowest value is represented by \u0000 and highest value is represented by \uFFFF.

## Unit 1: Introduction To Java And Elementary Programming

### 6. Creating, compiling and Executing a simple java program

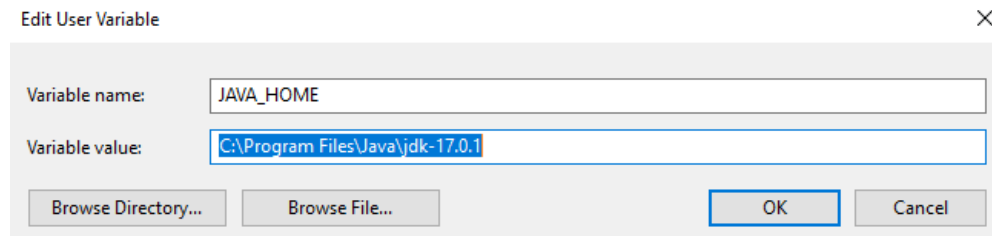
#### Setup JAVA environment

1. In the search field type in – advanced system settings

2. Set JAVA\_HOME Environment variable

In “System Properties window” click “Environment Variables...”

Under “System variables” click the “New...” button and enter JAVA\_HOME as “Variable name” and the path to your Java JDK directory under “Variable value”



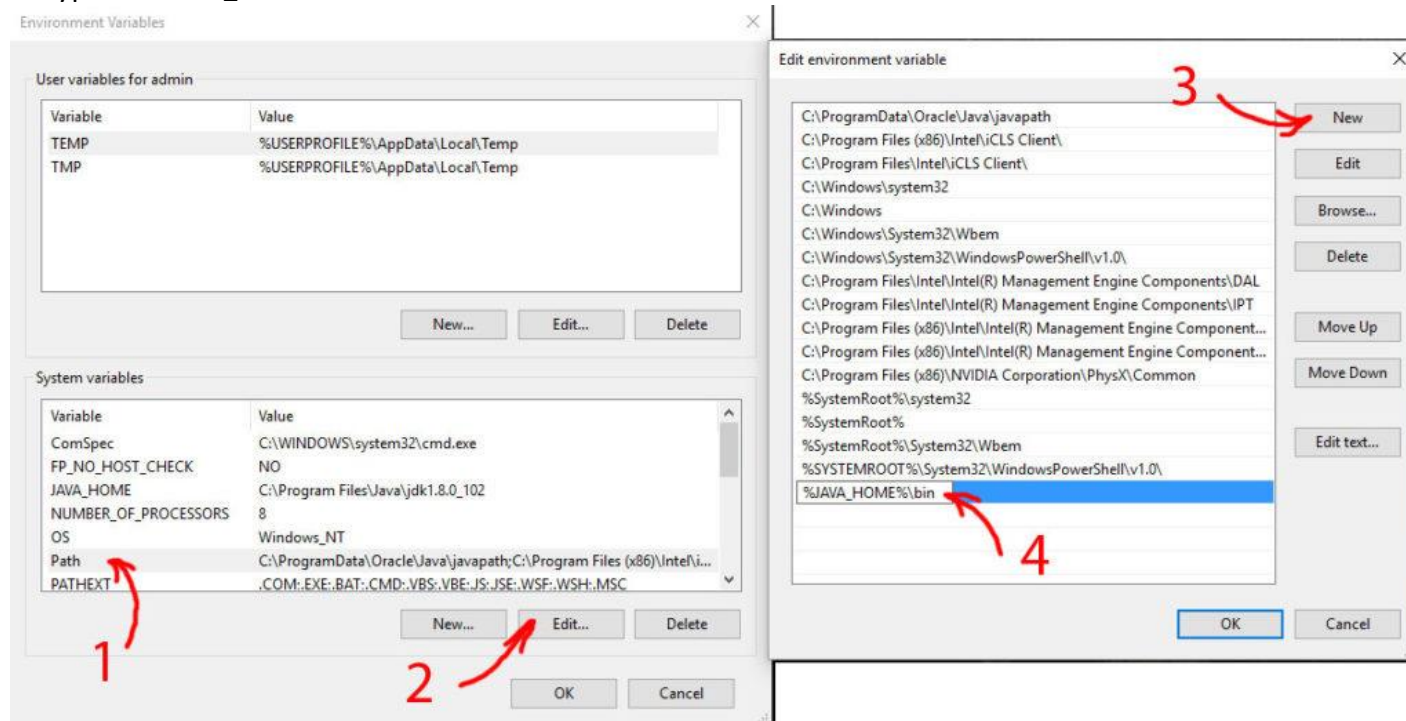
3. Update System PATH

1. In “Environment Variables” window under “System variables” select Path

2. Click on “Edit...”

3. In “Edit environment variable” window click “New”

4. Type in %JAVA\_HOME%\bin



4. Test your configuration

Open a new command prompt and type in:

`echo %JAVA_HOME%`

this will print out the directory JAVA\_HOME points to or empty line if the environment variable is not set correctly

Now type in:

`Javac --version`

## Unit 1: Introduction To Java And Elementary Programming

### Command Prompt

```
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\student>echo %JAVA_HOME%
C:\Program Files\Java\jdk-17.0.1

C:\Users\student>javac --version
javac 17.0.1

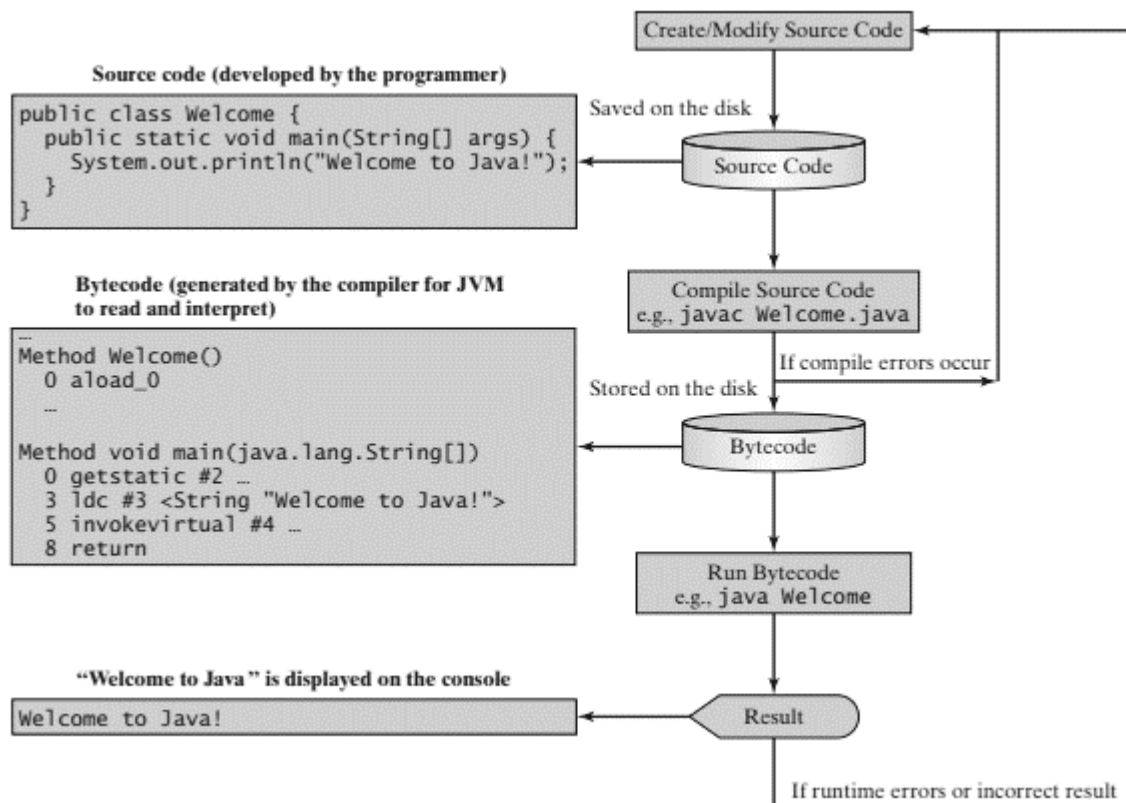
C:\Users\student>
```

### Steps for creating , compiling and executing a JAVA Program:

Simple Java Program:

```
public class FirstJavaProgram {
    public static void main(String[] args){
        System.out.println("This is my first program in java");
    } //End of main
} //End of FirstJavaProgram Class
```

**Output:** This is my first program in java



How to compile and run the above program

**Prerequisite:** You need to have JDK installed on your system. You have to set JDK path

**Step 1:** Open a text editor, like Notepad on windows. Copy the above program and paste it in the text editor.

## Unit 1: Introduction To Java And Elementary Programming

**Step 2:** Save the file as **FirstJavaProgram.java**. We should always name the file same as the public class name. In this program, the public class name is `FirstJavaProgram`, that's why the file name should be **FirstJavaProgram.java**.

**Step 3:** To compile the program. Open **command prompt (cmd) on Windows**.

To compile the program, type the following command and hit enter.

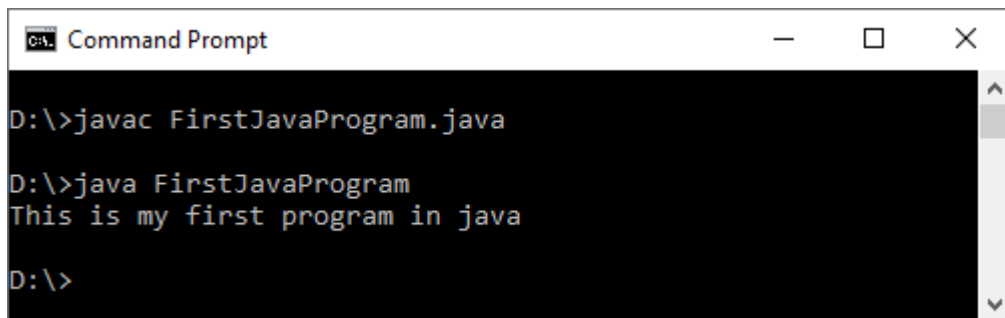
```
javac FirstJavaProgram.java
```

**Step 4:** After compilation the .java file gets translated into the .class file(byte code). Now we can run the program.

To run the program, type the following command and hit enter:

```
java FirstJavaProgram
```

Note that you should not append the .java extension to the file name while running the program.



```
Command Prompt

D:\>javac FirstJavaProgram.java

D:\>java FirstJavaProgram
This is my first program in java

D:\>
```

### 7. Reading input from console

Reading input from the console enables the program to accept input from the user.

Console input is not directly supported in Java, but you can use the `Scanner` class to create an object to read input from `System.in`, as follows:

```
Scanner input = new Scanner(System.in);
```

Above statement creates a `Scanner` object and assigns its reference to the variable `input`. An object may invoke its methods. To invoke a method on an object (variable `input`) is to ask the object to perform a task. You can invoke the `nextDouble()` method to read a double value as follows:

```
double radius = input.nextDouble();
```

#### Scanner class

#### **Example**

```
import java.util.Scanner;

public class Demo{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter String:");
        String mystr = sc.nextLine();
        System.out.println("The string is "+mystr);
        System.out.println("Enter Integer Value:");
        int val = sc.nextInt();
    }
}
```

## Unit 1: Introduction To Java And Elementary Programming

```
System.out.println("The integer is "+val);
System.out.println("Enter Float Value:");
Float float = sc.nextFloat();
System.out.println("The float value is"+float);
}
}
```

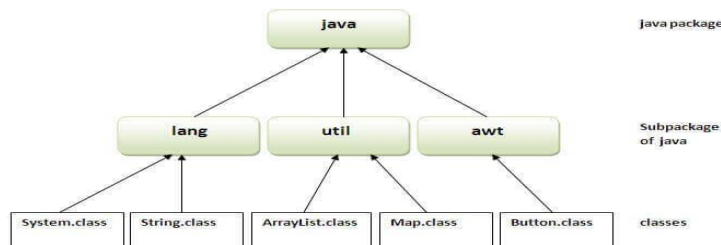
- A class named Demo contains the main function.
- An instance of the Scanner class is created and the 'nextLine' function is used to read every line of a string input.
- An integer value is defined and it is read from the standard input console using 'nextInt'.
- Similarly, 'nextFloat' function is used to read float type input from the standard input console.

### 8. What do you understand by package? Discuss benefits of package.

- A **Java package** is a collection of similar classes, interfaces, and sub-packages.
- In Java, packages are divided into two types: built-in packages and user-defined packages.
- There are numerous built-in packages available, including java, lang, awt, javax, swing, net, io, util, sql, and others.

#### Benefits of Java Package:

- Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- Java package provides access protection.
- Java package removes naming collision.



### 9. How package can be created in JAVA? Explain with suitable example.

- The **package** keyword is used to create a package in java.

```
//save as Simple.java
package mypack;
public class Simple{
    public static void main(String args[]){
        System.out.println("Welcome to package");
    }
}
```

#### How to compile java package

- If you are not using any IDE, you need to follow the syntax given below:

```
javac -d directory javafilename
```

For example:

```
javac -d . Simple.java
```

- The -d switch specifies the destination where to put the generated class file.
- You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc.
- If you want to keep the package within the same directory, you can use . (dot).

## Unit 1: Introduction To Java And Elementary Programming

### How to run java package program

- You need to use fully qualified name e.g. mypack.Simple etc to run the class.

To Compile: `javac -d . Simple.java`

To Run: `java mypack.Simple`

### **Output:**

Welcome to package

- The -d is a switch that tells the compiler where to put the class file i.e. it represents destination.
- The . represents the current folder.

### **10. Which package is automatically imported into every source file?**

- The package in Java is a collection of Java classes and interfaces.
- When we use the classes of a specific package, we must import the package in which those classes are defined.
- The fully qualified name, which includes the package name, is used by the class.
- We do not import any packages in the majority of the basic Java programmes.
- The question here is how Java programmes allow us to use classes defined in a specific package when we don't import any packages.
- JVM resolves this issue internally by importing the **java.lang** package by default.

#### java.lang Package

- By default, the Java compiler imports the java.lang package.
- It includes the fundamental classes required to create a basic Java programme.
- The most important classes are Object (the root of the class hierarchy) and Class (instances of which represent classes at run time).

### **11. List various built in package used in java. OR**

#### **List out all packages with short description**

- Java has already defined and included some packages in its software; these packages are known as built-in packages or predefined packages.
- These packages contain a large number of classes and interfaces that java programmers can use for a variety of purposes.
- Programmers can include these packages in their code and use the classes and interfaces provided by these packages.
- There are many built-in packages available in java. Some of the inbuilt packages in java are :

#### java.awt :

- Contains classes for creating user interfaces and for painting graphics and images.
- Classes like Button, Color, Event, Font, Graphics, Image etc are part of this package.

#### java.io :

- Provides classes for system input/output operations.
- Classes like BufferedReader, BufferedWriter, File, InputStream, OutputStream, PrintStream, Serializable etc are part of this package.

#### java.lang :

- Contains classes and interfaces that are fundamental to the design of Java programming language.
- Classes like String, StringBuffer, System, Math, Integer etc are part of this package.

#### java.net :

- Provides classes for implementing networking applications.



## Unit 1: Introduction To Java And Elementary Programming

- Classes like Authenticator, HttpCookie, Socket, URL, URLConnection, URLEncoder, URLDecoder etc are part of this package.

### java.sql :

- Provides the classes for accessing and processing data stored in a database.
- Classes like Connection, DriverManager, PreparedStatement, ResultSet, Statement etc are part of this package.

### java.util :

- Contains the collections framework, some internationalization support classes, properties, random number generation classes.
- Classes like ArrayList, LinkedList, HashMap, Calendar, Date, TimeZone etc are part of this package.

## 12. Method main is a public static method. Justify

- The main() method in Java programmes is the point at which the programme begins its execution, or simply the entry point of Java programmes.
- As a result, it is one of the most important Java methods, and a thorough understanding of it is essential.
- When a Java programme is executed, the Java compiler or JVM searches for the main method.
- For the JVM to recognise the main method as its entry point, its signature must be in a specific format.
- If we change the method signature, the programme compiles but does not run.
- The java.exe programme is used to run a Java programme.
- In turn, the Java.exe makes Java Native Interface (JNI) calls, which load the JVM.
- The command line is parsed by java.exe, a new String array is created, and the main() method is called.
- The main method is connected to a daemon thread, which is destroyed only when the Java programme terminates execution.
- **The most common syntax for the main() method is:**

```
// Java Program to demonstrate the  
// syntax of the main() function
```

```
class Demo {  
    public static void main(String[] args)  
    {  
        System.out.println("Hello");  
    }  
}
```

### Output:

```
Hello
```

- Explanation: Every word in the public static void main statement has got a meaning to the JVM.

### 1. Public

- It is an Access modifier, which specifies from where and who can access the method.
- Making the main() method public makes it globally available.
- It is made public so that JVM can invoke it from outside the class as it is not present in the current class.

```
// Java Program to demonstrate the  
// use of any other access modifier  
// other than public
```

```
class Demo {
```



## Unit 1: Introduction To Java And Elementary Programming

```
private static void main(String[] args)
{
    System.out.println("Hello");
}
```

Error: Main method not found in class,  
please define the main method as:  
public static void main(String[] args)  
or a JavaFX application class  
must extend javafx.application.Application

### 2. Static

- It is a keyword that is when associated with a method, making it a class-related method.
- The main() method is static so that JVM can invoke it without instantiating the class.
- This also saves the unnecessary wastage of memory which would have been used by the object declared only for calling the main() method by the JVM.

```
// Java Program to demonstrate the
// error occurred when we dont use the
// static keyword in the main() method
```

```
class Demo {
    public void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

Error: Main method is not static in class test,  
please define the main method as: public static  
void main(String[] args)

### 3. void

- It is a keyword and is used to specify that a method doesn't return anything.
- As the main() method doesn't return anything, its return type is void.
- As soon as the main() method terminates, the java program terminates too.
- Hence, it doesn't make any sense to return from the main() method as JVM can't do anything with the return value of it.

```
// Java Program to demonstrate the
// error occurred when we dont use the
// void return type in the main() method
```

```
class Demo {
    public static int main(String[] args)
    {
        System.out.println("Hello");
        return 1;
    }
}
```

Error: Main method not found in class,  
please define the main method as:  
public static void main(String[] args)  
or a JavaFX application class must  
extend javafx.application.Application

## Unit 1: Introduction To Java And Elementary Programming

### 4. main

- It is the name of the Java main method.
- It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

```
// Java Program to demonstrate the  
// error occurred when we name the  
// main() method as main.
```

```
class Demo {  
    public static void myMain(String[] args)  
    {  
        System.out.println("Hello");  
    }  
}
```

Error: Main method not found in class,  
please define the main method as:  
public static void main(String[] args)  
or a JavaFX application class  
must extend javafx.application.Application

### 5. String[] args

- It stores Java command-line arguments and is an array of type java.lang.String class.
- Here, the name of the String array is args but it is not fixed and the user can use any name in place of it.

```
// Java Program to demonstrate  
// the working of String[] args  
// in the main() method
```

```
class Demo {  
  
    // Command-Line Code ->  
    // javac Demo.java  
    // java Demo 1 2 3  
  
    public static void main(String[] args)  
    {  
        System.out.println(args[0]);  
        System.out.println(args[1]);  
        System.out.println(args[2]);  
    }  
}
```

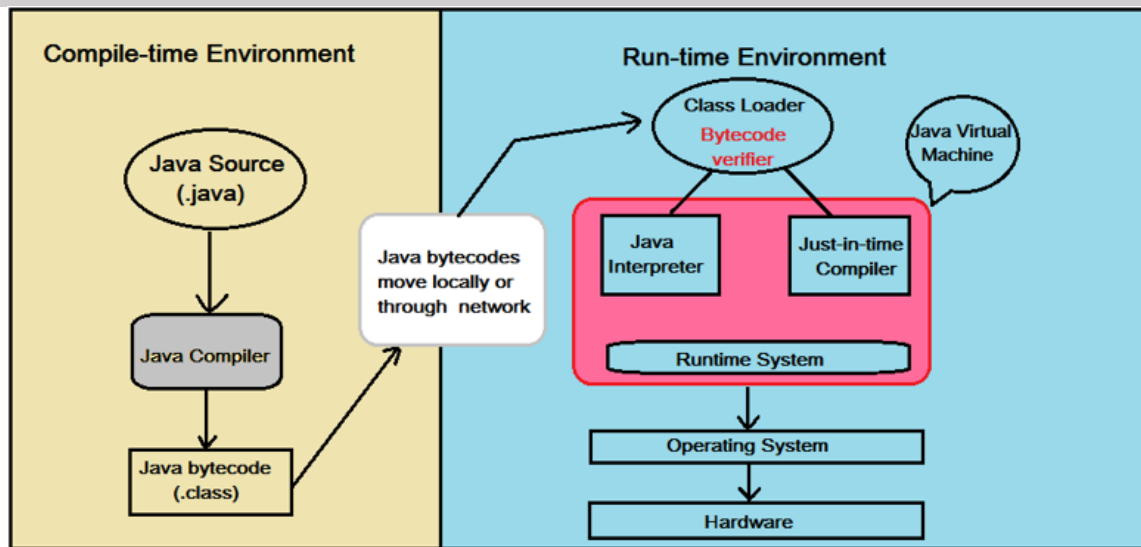
#### Output:

```
1  
2  
3
```

- Apart from the above-mentioned signature of main, you could use public static void main(String args[]) or public static void main(String... args) to call the main function in Java.
- The main method is called if its formal parameter matches that of an array of Strings.

## Unit 1: Introduction To Java And Elementary Programming

13. Java program is to be compiled first and then to be interpreted for execution. True or false? Justify your answer



Code written in Java is:

- First compiled to bytecode by a program called javac as shown in the left section of the image.
- Then, the Java runtime environment may compile and/or interpret the bytecode by using the Java Interpreter/JIT Compiler.

## 14. Explain features of JAVA

### Simple

- Java is very easy to learn, and its syntax is simple, clean and easy to understand after learning basic concepts of object oriented programming.
- Java has removed many complicated features like explicit pointers. There is an Automatic Garbage Collection in Java to remove unreferenced objects.

### Object Oriented

- Java is an object-oriented programming language because everything in Java is an object. It is a methodology for simple software development and maintenance by following some rules.
- Object-oriented means a software is organized as a combination of different objects.
- Basic concepts of OOPs are: Object, Class, Inheritance, Polymorphism, Abstraction, Encapsulation

### Platform Independent

- In Java, programs are compiled into byte code and that byte code is platform-independent.
- The byte code is executed by the Java Virtual Machine and the Java Virtual Machine is platform dependent.
- Java is platform-independent.
- Any machine to execute the byte code needs the Java Virtual Machine.

### Interpreted

- Java can be considered both a compiled and an interpreted language because its source code is first compiled into a byte-code.
- This byte-code runs on the Java Virtual Machine (JVM), which is usually a software-based interpreter.

## Unit 1: Introduction To Java And Elementary Programming

### Portable

- Java is portable because it facilitates you to carry the Java byte-code to any platform.
- It doesn't require any implementation.

### Architecture-neutral

- Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
- However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

### High performance

- Java is faster than other traditional interpreted programming languages because Java byte-code is "close" to native code.
- It is still a little bit slower than a compiled language (e.g., C++).
- Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

### Distributed

- Java is distributed because it facilitates users to create distributed applications in Java.
- RMI (Remote Method Invocation) and EJB (enterprise java bean) are used for creating distributed applications.
- This feature of Java makes us able to access files by calling the methods from any machine on the internet.

### Multithreaded

- A thread is like a separate program, executing concurrently.
- We can write Java programs that deal with many tasks at once by defining multiple threads.
- The main advantage of multi-threading is that it doesn't occupy memory for each thread.
- It shares a common memory area. Threads are important for multi-media, Web applications, etc.

### Dynamic

- Java is a dynamic language.
- It supports the dynamic loading of classes.
- It means classes are loaded on demand.
- It also supports functions from its native languages, i.e., C and C++.
- Java supports dynamic compilation and automatic memory management (garbage collection).

## 15. Why java is preferred as a programming language for Internet?

### Java was designed to be an object-oriented, cross-platform programming language.

Write java code once, and then you can use the code anywhere. It is one of the best programming language for developing applications that work on different operating systems.

### Java is a highly scalable language

It can maintain its performance while it's volume of data or requests increase. That is, performance holds steady for any individual user while the number of users increases.

## Unit 1: Introduction To Java And Elementary Programming

### Better Memory management

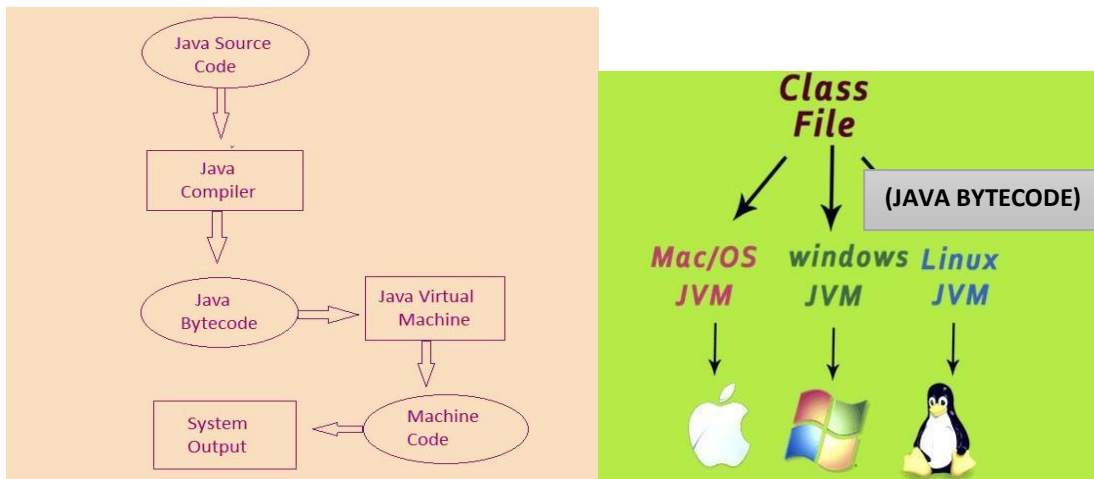
The Java language provides memory management by using a garbage collector. By using this, the programmer does not have to continuously monitor when objects are no longer being used and then delete them from memory.

### Multithreading Provides High Performance

- You can use Java to create both single-threaded and multithreaded applications. Developers can write programs using a single thread, which means it will only execute one task at a time. This type of application becomes very slow as the functions do not occur concurrently.
- For this reason, developers prefer to use multithreaded programs that allow them to split up tasks among different threads, which will then run in parallel.

### 16. What makes a Java platform independent language? Explain in detail.

The meaning of platform-independent is that the java compiled code (byte code) can run on all operating systems.



- Java source code(program) is compiled by the **java compiler** (javac)
- The **compiler** converts java code(.java file) into **bytecode** (.class file).
- **The bytecode generated by Compiler is platform independent** because it can be executed on multiple platforms (operating systems), i.e., Write Once and Run Anywhere (WORA).
- The bytecode of a java file will be same - no matter if the java file was compiled on Windows/Linux/Mac etc.
- Java Virtual Machine (JVM) converts the bytecode into machine specific code.
- **Java Compiler and JVM both are platform dependent.**  
(JVM is different across OS)
- **Note:** The working of a java compiler (javac),is same across the platforms, i.e to generate the same byte code even for different os, from the algorithmic perspective. However, the “javac” executables themselves are different across OS.

### 17. What are syntax errors (compile errors), runtime errors, and logic errors?

#### **Programming Errors**

Programming errors can be categorized into three types: syntax errors, runtime errors, and logic errors.

#### **Syntax Errors**

- Errors that are detected by the compiler are called syntax errors or compile errors.

## Unit 1: Introduction To Java And Elementary Programming

- **Program : ShowSyntaxErrors.java**

```
public class ShowSyntaxErrors {  
    public static main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

Error1: Invalid method declarations: return type required (void is missing at line3)

Error2: Unclosed string literal (string is not enclosed with double quotes in print statement at line 4)

Common Error 1: Missing Braces

Common Error 2: Missing Semicolons

Common Error 3: Missing Quotation Marks

Common Error 4: Misspelling Names

### Runtime Errors

- Runtime errors are errors that cause a program to terminate abnormally.
- For instance, if the program expects to read in a number, but instead the user enters a string, this causes data-type errors (input errors) to occur in the program.
- Another example of runtime errors is division by zero.

**Program: ShowRuntimeErrors.java**

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        System.out.println(1 / 0);  
    }  
}
```

### Logical Errors

- Logic errors occur when a program does not perform the way it was intended to.
- **Program : ShowLogicErrors.java**

```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.println("Celsius 35 is Fahrenheit degree ");  
        System.out.println((9 / 5) * 35 + 32);  
    }  
}
```

**Output:** Celsius 35 is Fahrenheit degree 67

- You will get Fahrenheit 67 degrees, which is wrong. It should be 95.0. In Java 9 / 5 is 1. To get the correct result, you need to use 9.0 / 5, which results in 1.8.

## 18. Write a note on Programming style and documentation [comments, conventions, indentation]

### Comments

- Comments are used to make the program more readable by adding the details of the code.
- It makes easy to maintain the code and to find the errors easily.
- The comments can be used to provide information or explanation about the variable, method, class, or any statement.

## Unit 1: Introduction To Java And Elementary Programming

### In Java there are three types of comments:

#### 1. Single – line comments.

Use inline comments whenever the meaning of the code is not immediately obvious from the text. For example, inline comments can be useful to summarize cases in a conditional expression as follows:

```
if (xPosition < xLeft)           // left of box
```

#### 2. Multi – line comments.

```
/* Let's declare and  
print variable in java. */  
int i=10;
```

#### 3. Documentation comments

Documentation comments are usually used to write large programs for a project or software application as it helps to create documentation API. These APIs are needed for reference, i.e., which classes, methods, arguments, etc., are used in the code.

To create documentation API, we need to use the javadoc tool. The documentation comments are placed between `/**` and `*/`.

```
/**  
 * This method calculates the summation of two integers.  
 * @param input1 This is the first parameter to sum() method  
 * @param input2 This is the second parameter to the sum() method.  
 */  
public int sum(int input1, int input2){  
    return input1 + input2;  
}
```

### Programming style

#### block styles used in java

A block is a group of statements surrounded by braces. There are two popular styles, next-line style and end-of-line style, as shown below.

```
public class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Block Styles");  
    }  
}
```

Next-line style

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Block Styles");  
    }  
}
```

End-of-line style

#### • Write self-documenting code.

1. Choose meaningful names for all variables, parameters, classes, and methods. Use complete words instead of abbreviations where practical.
2. Use named constants instead of numbers throughout your code. Rule of thumb: if a constant is used more than once, give it a name. [final double PI=3.14;]
3. Avoid repetitive code by writing a method and calling it multiple times.



## Unit 1: Introduction To Java And Elementary Programming

- **Follow standard formatting conventions.**

variables and method names

the first letter is lower case, with the first letter of each subsequent word capitalized  
for example: `int rateOfInterest = 18;` **OR** `float radius=1.0;`

class names

the start of each word is capitalized

for example, `public class AreaOfCircle {`

constants

the entire word is capitalized

for example `public final double PI = 3.14159;`

Note: Do not choose class names that are already used in the Java library. For example, since the System class is defined in Java, you should not name your class System.

- **Indentation:**

braces

the open brace ({} ) must match the corresponding close brace ({} )

code inside braces, conditional statements and loops

indent one level (about 3 spaces) for each level of curly braces ({} )

continued lines

when a statement continues across two or more lines, indent the second and remaining lines an equal amount past the start of the first line of the statement.

Example of proper indentation technique:

```
balance = balance - requested amount;  
System.out.println("Withdrawal of " + requestedAmount +  
    " successful, leaving " +  
    balance + ".");
```

- **Order of variables and methods:**

- Constants: with public constants before private ones
- class variables (static variables) immediately after the constants, listing the public variables first
- instance variables should follow the class variables, with the public ones first
- constructors should precede all other methods
- public accessors should immediately follow the constructors
- other methods, public and private can be listed in any logical order, with related methods near each other
- local variables should be declared as close as possible to their point of use.  
for example, a variable used as a counter in a for loop can be declared as follows:  
`for (int i = 0; i < 10; i++)`

### 19. Identifiers And Variables

**What is variable? How can we define variable in java? Also list rules for valid variable names.**

*Identifiers are the names that identify the elements such as classes, methods, and variables in a program.*

All identifiers must obey the following rules:

## Unit 1: Introduction To Java And Elementary Programming

- An identifier is a sequence of characters that consists of letters, digits, underscores (`_`), and dollar signs (`$`).
- An identifier must start with a letter, an underscore (`_`), or a dollar sign (`$`). It cannot start with a digit.
- An identifier cannot be a reserved word. (keywords)
- An identifier cannot be **true**, **false**, or **null**.
- An identifier can be of any length.

For example, **\$2**, **ComputeArea**, **area**, **radius**, and **printArea** are legal identifiers, Whereas **2A** and **d+4** are not because they do not follow the rules.

Since Java is case sensitive, **area**, **Area**, and **AREA** are all different identifiers.

**Variables** are used to store values that may be changed in the program.

The *variable declaration* tells the compiler to allocate appropriate memory space for the variable based on its data type. A variable must be declared before it can be assigned a value.

The syntax for declaring a variable is

```
datatype variableName;
```

Here are some examples of variable declarations:

```
int count; // Declare count to be an integer variable
double radius; // Declare radius to be a double variable
double interestRate; // Declare interestRate to be a double variable
```

You can also use a shorthand form to declare and initialize variables of the same type together. For example,

```
int i = 1, j = 2;
```

To improve readability, Java allows you to use underscores between two digits in a number literal. For example, the following literals are correct.

```
long ssn = 232_45_4519;
long creditCardNumber = 2324_4545_4519_3415L;
```

However, `45_` or `_45` is incorrect. The underscore must be placed between two digits.

### 20. Assignment statements

An assignment statement designates a value for a variable. An assignment statement can be used as an expression in Java.

The syntax for assignment statements is as follows:

```
variable = expression;
```

For example, consider the following code:

```
int y = 1; // Assign 1 to variable y
double radius = 1.0; // Assign 1.0 to variable radius
int x = 5 * (3 / 2); // Assign the value of the expression to x
x = y + 1; // Assign the addition of y and 1 to x
double area = radius * radius * 3.14159; // Compute area
x = x + 1; // A variable can be used at both sides
```

For example, the following statement is correct:

```
System.out.println(x = 1);
```

which is equivalent to

## Unit 1: Introduction To Java And Elementary Programming

```
x = 1;
System.out.println(x);
```

If a value is assigned to multiple variables, you can use this syntax:

```
i = j = k = 1;
```

### 21. Named constants and naming conventions

A named constant is an identifier that represents a permanent value. A named constant, or simply constant, represents permanent data that never changes.

Here is the syntax for declaring a constant  $\pi$ :

```
final datatype CONSTANTNAME = value;
final double PI = 3.14159;
```

A constant must be declared and initialized in the same statement. The word `final` is a Java keyword for declaring a constant.

Program : `ComputeAreaWithConstant.java`

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConstant {
    public static void main(String[] args) {
        final double PI = 3.14159; // Declare a constant

        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * PI;

        // Display result
        System.out.println("The area for the circle of radius " +
            radius + " is " + area);
    }
}
```

#### Output:

```
Enter a number for radius: 10
The area for the circle of radius 10.0 is 314.159
```

There are three benefits of using constants:

- (1) You don't have to repeatedly type the same value
- (2) You need to change it only in a single location
- (3) A descriptive name for a constant makes the program easy to read.

### 22. Explain Data Types in detail with example. OR What are the four integer types supported by Java

Java defines 8 primitive data types : byte, short, int, long, char, float, double and boolean.

They are divided into the following categories:

- Integers

## Unit 1: Introduction To Java And Elementary Programming

- Floating Point Numbers
- Characters
- Boolean Type

### Integers

These are of four types: byte, short, int, long. It is important to note that these are signed positive and negative values. Signed integers are stored in a computer using 2's complement. It consist both negative and positive values but in different formats like (-1 to -128) or (0 to +127). An unsigned integer can hold a larger positive value, and no negative value like (0 to 255). Unlike C++ there is no unsigned integer in Java.

- byte:
  - Byte data type is an 8-bit signed two's complement integer.
  - Wrapper Class: Byte
  - Minimum value: -128 ( $-2^7$ )
  - Maximum value: 127 ( $2^7 - 1$ )
  - Default value: 0
  - Example: `byte a = 10 , byte b = -50;`
- short:
  - Short data type is a 16-bit signed two's complement integer.
  - Wrapper Class: Short
  - Minimum value: -32,768 ( $-2^{15}$ )
  - Maximum value: 32,767 ( $2^{15} - 1$ )
  - Default value: 0.
  - Example: `short s = 20, short r = -100;`
- int:
  - int data type is a 32-bit signed two's complement integer. It is generally used as the default data type for integral values unless there is a concern about memory.
  - Wrapper Class: Integer
  - Minimum value: ( $-2^{31}$ )
  - Maximum value: ( $2^{31} - 1$ )
  - The default value: 0.
  - Example: `int a = 50000, int b = -20`
- long:
  - Long data type is a 64-bit signed two's complement integer.
  - Wrapper Class: Long
  - Minimum value: ( $-2^{63}$ )
  - Maximum value: ( $2^{63} - 1$ )
  - Default value: 0L.
  - Example: `long x = 200000L, long y = -500000L;`

By default all integer type variable is "int". So `long num=600851475143` will give an error. But it can be specified as long by appending the suffix L (or l)

## Unit 1: Introduction To Java And Elementary Programming

### Floating- Point

These are also called real numbers and are used for expressions involving fractional precision. These are of two types: float, double. Float is actually avoided in case of precise data such as currency or research data.

- float:
  - float data type is a single-precision 32-bit IEEE 754 floating point.
  - Wrapper Class: Float
  - Float is mainly used to save memory in large arrays of floating point numbers.
  - Default value: 0.0f.
  - Example: `float f = 14.5f;`

The default data type of floating-point number is double. So float f = 24.5 will introduce an error. However, we can append the suffix F (or f) to designate the data type as float.

- double:
  - double data type is a double-precision 64-bit IEEE 754 floating point. This data type is generally the default choice. This data type should never be used for precise values, such as currency.
  - Wrapper Class: Double
  - This data type is generally used as the default data type for decimal values.
  - Default value: 0.0d.
  - Example: `double d1 = 163.400778;`

### Character

We use this data type to store characters. This is not the same as the char in C/C++. Java uses a UNICODE, internationally accepted character set. Char in Java is 16-bits long while that in C/C++ is 8-bits.

- Wrapper Class: Character
- Minimum value: `'\u0000'` (or 0).
- Maximum value: `'\uffff'` (or 65,535).
- Default value: null (`'\u0000'`).
- Example: `char letterB = 'b';`

### Boolean

This is used for storing logical values. A boolean type can have a value of either true or false. This type is generally returned by relational operators.

- There are only two possible values: true and false.
- Wrapper Class: Boolean
- This data type is used for simple flags that track true/false conditions.
- Default value is false.
- Example: `boolean b = true,`  
`boolean b1 = 1`(this is not possible in java and give incompatible type error)

## Unit 1: Introduction To Java And Elementary Programming

### String

String is not a primitive data type, but it lets you store multiple character data types in an array and has many methods that can be used. It is used quite commonly when the user types in data and you have to manipulate it.

*A string is a sequence of characters.*

For example, the following code declares message to be a string with the value "Welcome to Java".

```
String message = "Welcome to Java";
```

#### Simple Methods for String Objects

Method	Description
length()	Returns the number of characters in this string.
charAt(index)	Returns the character at the specified index from this string.
concat(s1)	Returns a new string that concatenates this string with string s1.
toUpperCase()	Returns a new string with all letters in uppercase.
toLowerCase()	Returns a new string with all letters in lowercase
trim()	Returns a new string with whitespace characters trimmed on both sides.

You can use the length() method to return the number of characters in a string. For example, the following code

```
String message = "Welcome to Java";  
System.out.println("The length of " + message + " is " + message.length());
```

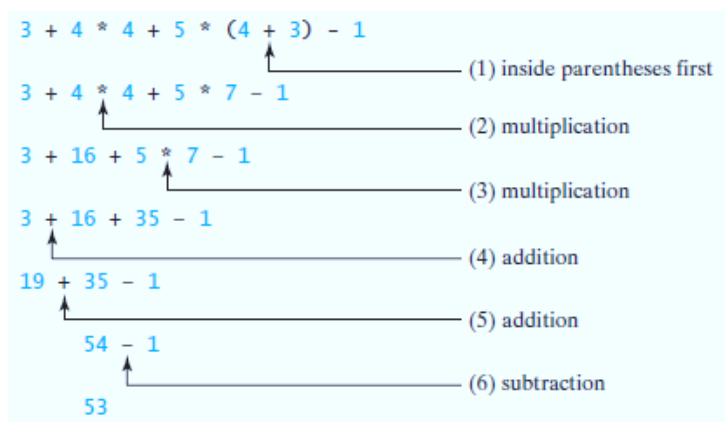
displays

```
The length of Welcome to Java is 15
```

### 23. Evaluating Expressions , Operator Precedence and Associativity

Operator precedence determines the order in which the operators in an expression are evaluated.

Evaluate the expression  $3 + 4 * 4 + 5 * (4 + 3) - 1$



## Unit 1: Introduction To Java And Elementary Programming

The table below lists the associativity & precedence of operators in Java; higher it appears in the table, the higher its precedence.

Java Operator Precedence and Associativity		
Operators	Precedence	Associativity
postfix increment and decrement	++ --	left to right
prefix increment and decrement, and unary	++ -- + - ~ !	right to left
multiplicative	* / %	left to right
additive	+ -	left to right
shift	<< >> >>>	left to right
relational	< > <= >= instanceof	left to right
equality	== !=	left to right
bitwise AND	&	left to right
bitwise exclusive OR	^	left to right
bitwise inclusive OR		left to right
logical AND	&&	left to right
logical OR		left to right
ternary	? :	right to left
assignment	= += -= *= /= %= &= ^=  = <<= >>= >>>=	right to left

### Example: Operator Precedence

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

        int a = 10, b = 3, c = 1, result;
        result = a-++c-++b;

        System.out.println(result);
    }
}
```

**Output: 4**

### Example: Associativity of Operators in Java

If an expression has two operators with similar precedence, the expression is evaluated according to its associativity (either left to right, or right to left). Let's take an example.

```
a = b = c;
```

Here, the value of `c` is assigned to variable `b`. Then the value of `b` is assigned of variable `a`. Why? It's because the associativity of `=` operator is from right to left.

## 24. Operators

**Explain Types of Operators (Augmented assignment, Increment and Decrement, Logical) OR Explain short circuited operators and shift operators.**

Operators in Java can be classified into 5 types:

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators
- Unary Operators
- Bitwise Operators & Shift Operators



## Unit 1: Introduction To Java And Elementary Programming

Unary	postfix	<i>expr++ expr--</i>	<b>int a=10,b=10;</b> System.out.println(a++ + ++a);//10+12=22 System.out.println(b++ + b++);//10+11=21  <b>~5 =&gt; -6 and ~( -5) =&gt; 4</b> <b>If int a = true then !a becomes false</b>
	prefix	<i>++expr --expr +expr -expr ~ !</i>	
Arithmetic	multiplicative	<i>* / %</i>	System.out.println(10*10/5+3-1*4/2);  Prints 21
	additive	<i>+ -</i>	
Shift	shift	<i>&lt;&lt;, &gt;&gt;</i>  <i>&gt;&gt;&gt;</i> (unsigned right shift) explained below	System.out.println(10<<3);//10*2^3=10*8=80 System.out.println(20<<2);//20*2^2=20*4=80 System.out.println(10>>2);//10/2^2=10/4=2 System.out.println(20>>3);//20/2^3=20/8=2
Relational	comparison	<i>&lt;, &gt;, &lt;=, &gt;=</i>	
	equality	<i>==, !=</i>	
Bitwise	bitwise AND	<i>&amp;</i>	<b>Int a=10,b=5,c=20;</b> System.out.println(a<b && a++ < c); //false && true = false System.out.println(a); //10 because second condition is not checked System.out.println(a<b & a++ < c); //false && true = false System.out.println(a); //11 because second condition is checked  The logical operator doesn't check the second condition if the first condition is false. It checks the second condition only if the first one is true.  The bitwise operator always checks both conditions whether first condition is true or false.
	bitwise exclusive OR	<i>^</i>	
	bitwise inclusive OR	<i> </i>	
Logical	logical AND	<i>&amp;&amp;</i>	 The logical operator doesn't check the second condition if the first condition is false. It checks the second condition only if the first one is true.  The bitwise operator always checks both conditions whether first condition is true or false.
	logical OR	<i>  </i>	
Ternary	ternary	<i>? :</i>	<b>int min =(2&lt;5) ? 2 : 5;                      // prints 2</b>
Assignment	assignment	<i>=, +=, -=, *=, /=, %= &amp;=, ^=,  =, &lt;=, &gt;= &gt;&gt;=</i>	<b>int b=20;</b>  <b>b-=4;    //b=b-4 (b=20-4)</b>

### Bitwise complement (~) Unary Example

change 0 to 1 and 1 to 0.

Int a = 5 ;

a = 5 = 0 1 0 1 (In Binary)

Bitwise Compliment Operation of 5

**~ 0 1 0 1**

**1 0 1 0 = 10 (In decimal)**

Output : 10

	Positive				Negative (2's complement)			
Number	128	64	32	16	8	4	2	1
Binary	0	0	0	0	0	1	0	1
~ 5	1	1	1	1	1	0	1	0
1's com.	0	0	0	0	0	1	0	1
Addition							+	1
2's com.	0	0	0	0	0	1	1	0

Output : -6

( - var ) - 1 = complement.  
(- 5) - 1 = - 6

## Unit 1: Introduction To Java And Elementary Programming

**~5 MSB =1** denotes a negative number adding 4 and 2 in 2's complement gives 6

Final output is -6

[**Positive number:** Increase number 5 by 1 gives 6 and changing the sign gives -6]

**~(-5) =>** decrease number 5 by 1 gives +4 [**negative number** becomes positive]

### Unsigned right shift example

//For positive number, >> and >>> works same

```
System.out.println(20>>2);    // 5
```

```
System.out.println(20>>>2);   // 5
```

//For negative number, >>> changes parity bit (MSB) to 0 instead of 1

```
System.out.println(-8>>1);    // -4
```

```
System.out.println(-8>>>1);   // 2147483644
```

Binary of -8 => 1111 1111 1111 1111 1111 1111 1111 1000

-8>>>1 => 0111 1111 1111 1111 1111 1111 1111 1000

### **NOTE**

in java negative numbers are stored using 2's complement => -8>>1 becomes -4 newly filled bit depends on left most bit

signed right shift >> (filled bit 0 if number is +ve and 1 if no. is -ve)

unsigned right shift >>> (filled bit 0 always)

Unlike unsigned Right Shift, there is no "<<<" operator in Java

**25. What is Type Casting? Explain widening and narrowing type casting.**

**OR**

**Explain type-conversion in java OR Explain Numeric type conversions.**

The process of converting a value from one data type to another is known as type conversion in Java.

Type conversion is also known as type casting in java or simply 'casting'.

If two data types are compatible with each other, Java will perform such conversion automatically or implicitly for you.

We can easily convert a primitive data type into another primitive data type by using type casting.

Similarly, it is also possible to convert a non-primitive data type (referenced data type) into another non-primitive data type by using type casting.

But we cannot convert a primitive data type into an advanced (referenced) data type by using type casting. For this case, we will have to use methods of Java Wrapper classes.

In Java, there are two types of casting:

**Widening Casting** (automatically) - converting a smaller type to a larger type size  
byte -> short -> char -> int -> long -> float -> double

## Unit 1: Introduction To Java And Elementary Programming

### Ex:1

```
int myInt = 9;
double myDouble = myInt; // Automatic casting: int to double

System.out.println(myInt);    // Outputs 9
System.out.println(myDouble); // Outputs 9.0
```

### Ex:2

```
char ch = 'C';
int i = ch;
System.out.println(i); //prints 97
```

**Narrowing Casting (manually)** - converting a larger type to a smaller size type  
double -> float -> long -> int -> char -> short -> byte

### Ex:1

```
double myDouble = 9.78d;
int myInt = (int) myDouble; // Manual casting: double to int

System.out.println(myDouble); // Outputs 9.78
System.out.println(myInt);    // Outputs 9
```

### Ex:2

```
int i = 97;
char ch = (char) i;
System.out.println("Character value of the given integer: "+ch); //prints C
```