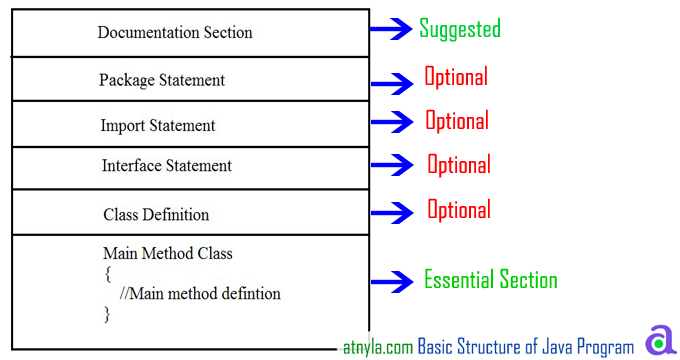
**2.6 Structure of Java Program :-**

A Java program consists of different sections. Some of them are mandatory but some are optional. The optional section can be excluded from the program depending upon the requirements of the programmer.



**Documentation Section**

It includes the comments that improve the readability of the program. A comment is a non-executable statement that helps to read and understand a program especially when your programs get more complex. It is simply a message that exists only for the programmer and is ignored by the compiler. A good program should include comments that describe the purpose of the program, author name, date and time of program creation. This section is optional and comments may appear anywhere in the program.   
Java programming language supports three types of comments.

Single line (or end-of line) comment:

It starts with a double slash symbol (//) and terminates at the end of the current line. The compiler ignores everything from// to the end of the line. For example:

// Calculate sum of two numbers

Multiline Comment:

Java programmer can use C/C++ comment style that begins with delimiter /\* and ends with \*/. All the text written between the delimiter is ignored by the compiler. This style of comments can be used on part of a line, a whole line or more commonly to define multi-line comment. For example.

/\*calculate sum of two numbers

and it is a multiline comment \*/

Nested Comment is not possible:

Comments cannot be nested. In other words, you cannot comment a line that already includes traditional comment. For example:

/\* x = y /\* initial value \*/ + z; \*/ **is** wrong.

Documentation comments:

This comment style is new in Java. Such comments begin with delimiter /\*\* and end with \*/. The compiler also ignores this type of comments just like it ignores comments that use / \* and \*/. The main purpose of this type of comment is to automatically generate program documentation. The java doc tool reads these comments and uses them to prepare your program's documentation in HTML format. For example.

/\*\* The text enclosed here will be part of program documentation \*/

**Package Statement**

A package is a collection of classes, interfaces and sub-packages. A sub package contains collection of classes, interfaces and sub-sub packages etc.

java.lang.\*; package is imported by default and this package is known as default package. It must appear as the first statement in the source code file before any class or interface declaration. This statement is optional.

**Example**Suppose you write the following package declaration as the first statement in the source code file.

**package** institute;

This statement declares that all classes and interfaces defined in this source file are part of the institute package. Only one package declaration can appear in the source file.

**Import statements**

Java contains many predefined classes that are stored into packages. In order to refer these standard predefined classes in your program, you need to use fully qualified name (i.e. Packagename.className). But this is a very tedious task as one need to retype the package path name along with the classname. So a better alternative is to use an import statement.

An import statement is used for referring classes that are declared in other packages. The import statement is written after a package statement but before any class definition. You can import a specific class or all the classes of the package.

**Example**If you want to import Date class of java.util package using import statement then write.

**import** java.util.Date;

This statement allows the programmer to use the simple classname Date rather than fully qualified classname java.util.Date in the code.

Unlike package statement, you can specify more than one import statement in your program.

**Example**

Import java.util.Date; /\* imports only the Date class in java.util package \*/

**import** java.applet.\*; // imports all the classes in java applet

**Interface Section**

In the interface section, we specify the interfaces. An interface is similar to a class but contains only constants and method declarations. Interfaces cannot be instantiated. They can only be implemented by classes or extended by other interfaces. It is an optional section and is used when we wish to implement multiple inheritance feature in the program.

**interface** stack

{

**void** push(**int** item); // Insert item into stack

**int** pop(); // Delete an item from stack

}

**Class Definition:**

Java program may contain multiple class definition. Classes are primary feature of Java program. The classes are used to map real world problems.

**class** Addition

{

**void** **add**(String args[])

{

**int** a=2, b=3, c;

c=a+b;

System.**out**.println(c);

}

}

**Main Method Class Section:**

The Class section describes the information about user-defined classes present in the program. A class is a collection of fields (data variables) and methods that operate on the fields. Every program in Java consists of at least one class, the one that contains the main method. The main () method which is from where the execution of program actually starts and follow the statements in the order specified.

The main method can create objects, evaluate expressions, and invoke other methods and much more. On reaching the end of main, the program terminates and control passes back to the operating system.

The class section is mandatory.

After discussing the structure of programs in Java, we shall now discuss a program that displays a string Hello Java on the screen.

// Program to display message on the screen

**class** HelloJava

{

**public** **static** **void** main(String args[])

{

System.**out**.println("Hello Java");

}

}

Class

Class is keyword used for developing user defined data type and every java program must start with a concept of class.

ClassName

"ClassName" represent a java valid variable name treated as a name of the class each and every class name in java is treated as user-defined data type.

Data member

Data member represents either instance or static they will be selected based on the name of the class.

User-defined methods

User-defined methods represents either instance or static they are meant for performing the operations either once or each and every time.

main() method

Each and every java program starts execution from the main() method. And hence main() method is known as program driver.

void

Since main() method of java is not returning any value and hence its return type must be void.

Static

Since main() method of java executes only once throughout the java program execution and hence its nature must be static.

public

Since main() method must be accessed by every java programmer and hence whose access specifier must be public.

String args[]

Each and every main() method of java must take array of objects of String.

Block of statements

Block of statements represents set of executable statements which are in term calling user-defined methods are containing business-logic.

Class

Class is keyword used for developing user defined data type and every java program must start with a concept of class.