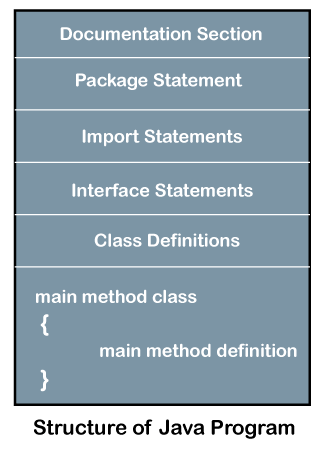
Structure of Java Program

Java is an [object-oriented programming](https://www.javatpoint.com/java-oops-concepts), **platform independent,** and **secure** programming language that makes it popular. Using the Java programming language, we can develop a wide variety of applications. So, before diving in depth, it is necessary to understand the **basic structure of Java program** in detail. In this section, we have discussed the basic **structure of a Java program**. At the end of this section, you will able to develop the [Hello world Java program](https://www.javatpoint.com/simple-program-of-java), easily.



Let's see which elements are included in the structure of a [Java program](https://www.javatpoint.com/java-programs). A typical structure of a [Java](https://www.javatpoint.com/java-tutorial) program contains the following elements:

* Documentation Section
* Package Declaration
* Import Statements
* Interface Section
* Class Definition
* Class Variables and Variables
* Main Method Class
* Methods and Behaviors

Documentation Section

The documentation section is an important section but optional for a Java program. It includes **basic information** about a Java program. The information includes the **author's name, date of creation, version, program name, company name,** and **description** of the program. It improves the readability of the program. Whatever we write in the documentation section, the Java compiler ignores the statements during the execution of the program. To write the statements in the documentation section, we use **comments**. The comments may be **single-line, multi-line,** and **documentation** comments.

* **Single-line Comment:** It starts with a pair of forwarding slash **(//)**. For example:

1. //First Java Program

* **Multi-line Comment:** It starts with a **/\*** and ends with **\*/.** We write between these two symbols. For example:

1. /\*It is an example of
2. multiline comment\*/

* **Documentation Comment:** It starts with the delimiter **(/\*\*)** and ends with **\*/**. For example:

1. /\*\*It is an example of documentation comment\*/

Package Declaration

The package declaration is optional. It is placed just after the documentation section. In this section, we declare the **package name** in which the class is placed. Note that there can be **only one package** statement in a Java program. It must be defined before any class and interface declaration. It is necessary because a Java class can be placed in different packages and directories based on the module they are used. For all these classes package belongs to a single parent directory. We use the keyword **package** to declare the package name. For example:

1. **package** Student ; //where Student is the package name
2. **package** com.Student; //where com is the root directory and javatpoint is the subdirectory

Import Statements

The package contains the many predefined classes and interfaces. If we want to use any class of a particular package, we need to import that class. The import statement represents the class stored in the other package. We use the **import** keyword to import the class. It is written before the class declaration and after the package statement. We use the import statement in two ways, either import a specific class or import all classes of a particular package. In a Java program, we can use multiple import statements. For example:

1. **import** java.util.Scanner; //it imports the Scanner class only
2. **import** java.util.\*; //it imports all the class of the java.util package

Interface Section

It is an optional section. We can create an **interface** in this section if required. We use the **interface** keyword to create an interface. An [interface](https://www.javatpoint.com/interface-in-java) is a slightly different from the class. It contains only **constants** and **method** declarations. Another difference is that it cannot be instantiated. We can use interface in classes by using the **implements** keyword. An interface can also be used with other interfaces by using the **extends** keyword. For example:

1. **interface** car
2. {
3. **void** start();
4. **void** stop();
5. }

Class Definition

In this section, we define the class. It is **vital** part of a Java program. Without the [class](https://www.javatpoint.com/object-and-class-in-java), we cannot create any Java program. A Java program may conation more than one class definition. We use the **class** keyword to define the class. The class is a blueprint of a Java program. It contains information about user-defined methods, variables, and constants. Every Java program has at least one class that contains the main() method. For example:

1. **class** Student //class definition
2. {
3. }

Class Variables and Constants

In this section, we define [variables](https://www.javatpoint.com/java-variables) and **constants** that are to be used later in the program. In a Java program, the variables and constants are defined just after the class definition. The variables and constants store values of the parameters. It is used during the execution of the program. We can also decide and define the scope of variables by using the modifiers. It defines the life of the variables. For example:

1. **class** Student //class definition
2. {
3. String sname;  //variable
4. **int** id;
5. **double** percentage;
6. }

Main Method Class

In this section, we define the **main() method.** It is essential for all Java programs. Because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class. It must be inside the class. Inside the main method, we create objects and call the methods. We use the following statement to define the main() method:

1. **public** **static** **void** main(String args[])
2. {
3. }

For example:

1. **public** **class** Student //class definition
2. {
3. **public** **static** **void** main(String args[])
4. {
5. //statements
6. }
7. }

You can read more about the Java main() method [here](https://www.javatpoint.com/java-main-method).

Methods and behavior

In this section, we define the functionality of the program by using the [methods](https://www.javatpoint.com/method-in-java). The methods are the set of instructions that we want to perform. These instructions execute at runtime and perform the specified task. For example:

1. **public** **class** Demo //class definition
2. {
3. **public** **static** **void** main(String args[])
4. {
5. **void** display()
6. {
7. System.out.println("Welcome to KKWIEER ");
8. }
9. //statements
10. }
11. }