**JAVA NAMING CONVENTIONS**

Java naming convention is not a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method, etc.

But, it is not forced to follow. So, it is known as convention not rule. These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

Advantage of naming conventions in java

By using standard Java naming conventions, you make your code easier to read for yourself and other programmers. Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

**The following are the key rules that must be followed by every identifier:**

The name must not contain any white spaces.

The name should not start with special characters like & (ampersand), $ (dollar), \_ (underscore).

Let's see some other rules that should be followed by identifiers.

**Class**

It should start with the uppercase letter.

It should be a noun such as Color, Button, System, Thread, etc.

Use appropriate words, instead of acronyms.

**Example: -**

public class Employee

{

//code snippet

}

**Method**

It should start with lowercase letter.

It should be a verb such as main(), print(), println().

If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

**Example:-**

class Employee

{

//method

void draw()

{

//code snippet

}

}

**Variable**

It should start with a lowercase letter such as id, name.

It should not start with the special characters like & (ampersand), $ (dollar), \_ (underscore).

If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName.

Avoid using one-character variables such as x, y, z.

Example :-

class Employee

{

//variable

int id;

//code snippet

}

**Constant**

It should be in uppercase letters such as RED, YELLOW.

If the name contains multiple words, it should be separated by an underscore(\_) such as MAX\_PRIORITY.

It may contain digits but not as the first letter.

**Example :-**

class Employee

{

//constant

static final int MIN\_AGE = 18;

//code snippet

}

**CamelCase in java naming conventions**

Java follows camel-case syntax for naming the class, interface, method, and variable.

If the name is combined with two words, the second word will start with uppercase letter always such as actionPerformed(), firstName, ActionEvent, ActionListener, etc.

**OBJECTS AND CLASSES IN JAVA**

In object-oriented programming technique, we design a program using objects and classes.

What is an object in Java

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.

An object has three characteristics:

**State**: represents the data (value) of an object.

**Behavior**: represents the behavior (functionality) of an object such as deposit, withdraw, etc.

**Identity**: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

**What is a class in Java**

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity.

**A class in Java can contain:**

Fields

Methods

Constructors

Blocks

**Syntax to declare a class:**

class <class\_name>{

field;

method;

}

**new keyword in Java :**

The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

**Object and Class Example: main within the class**

In this example, we have created a Student class which has two data members id and name. We are creating the object of the Student class by new keyword and printing the object's value.

Here, we are creating a main() method inside the class.

//Defining a Student class.

class Student{

//defining fields

int id;//field or data member or instance variable

String name;

//creating main method inside the Student class

public static void main(String args[]){

//Creating an object or instance

Student s1=new Student();//creating an object of Student

//Printing values of the object

System.out.println(s1.id);//accessing member through reference variable

System.out.println(s1.name);

}

}

**Object and Class Example: main outside the class**

//Java Program to demonstrate having the main method in

//another class

//Creating Student class.

class Student{

int id;

String name;

}

//Creating another class TestStudent1 which contains the main method

class TestStudent1{

public static void main(String args[]){

Student s1=new Student();

System.out.println(s1.id);

System.out.println(s1.name);

}

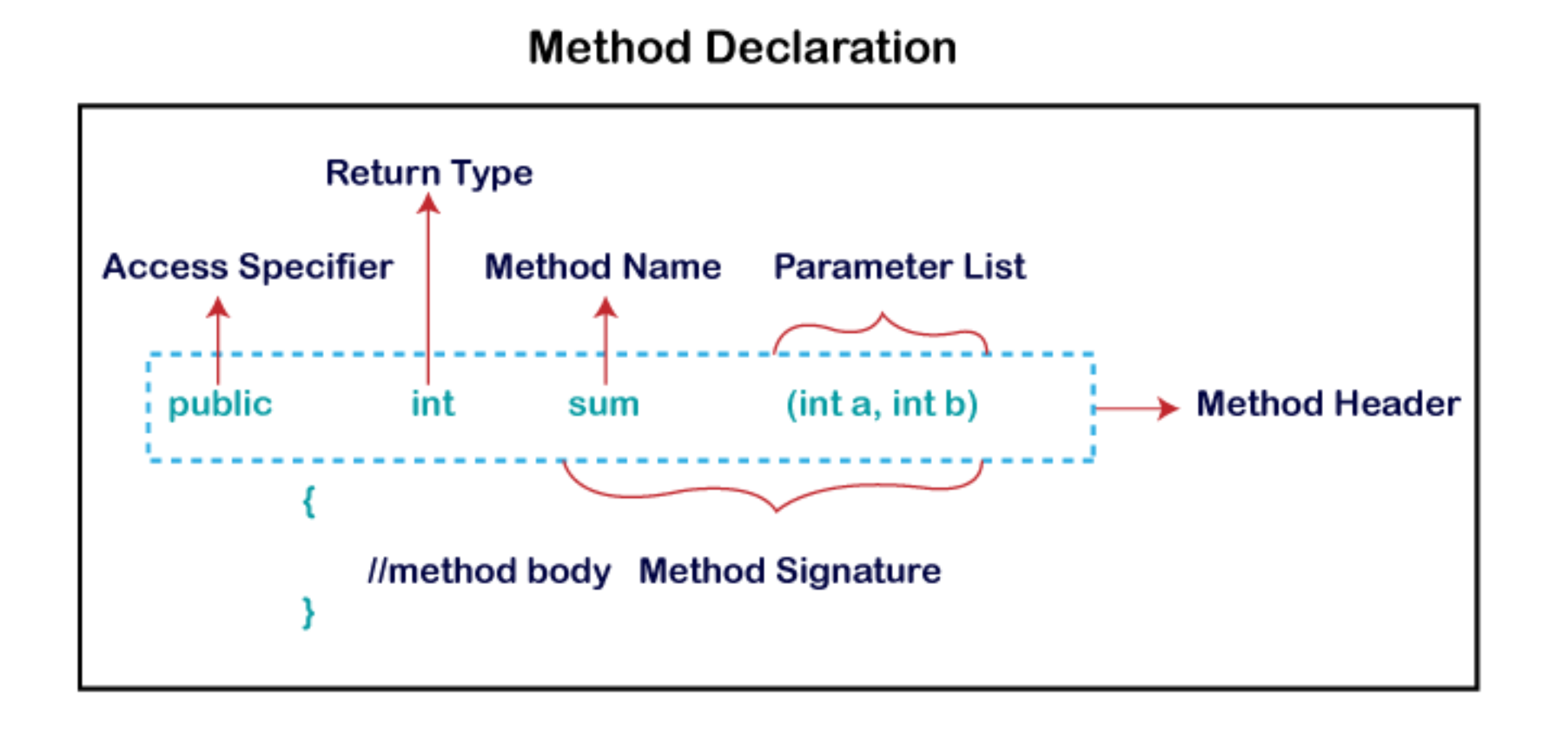
}

**METHOD IN JAVA**

In general, a method is a way to perform some task. Similarly, the method in Java is a collection of instructions that performs a specific task. It provides the reusability of code. We can also easily modify code using methods. In this section, we will learn what is a method in Java, types of methods, method declaration, and how to call a method in Java.

Method Declaration

The method declaration provides information about method attributes, such as visibility, return-type, name, and arguments. It has six components that are known as method header, as we have shown in the following figure.



**Types of Method**

There are two types of methods in Java:

1. Predefined Method
2. User-defined Method

**Predefined Method**

In Java, predefined methods are the method that is already defined in the Java class libraries is known as predefined methods. It is also known as the standard library method or built-in method. We can directly use these methods just by calling them in the program at any point. Some pre-defined methods are **length(), equals(), compareTo(), sqrt(), etc.** When we call any of the predefined methods in our program, a series of codes related to the corresponding method runs in the background that is already stored in the library.

public class Demo

{

public static void main(String[] args)

{

// using the max() method of Math class

System.out.print("The maximum number is: " + Math.max(9,7));

}

}

**User-defined Method**

The method written by the user or programmer is known as a user-defined method. These methods are modified according to the requirement.

**How to Create a User-defined Method**

Let's create a user defined method that checks the number is even or odd. First, we will define the method.

public static void findEvenOdd(int num)

{

//method body

if(num%2==0)

System.out.println(num+" is even");

else

System.out.println(num+" is odd");

}

**How to Call or Invoke a User-defined Method**

Once we have defined a method, it should be called. The calling of a method in a program is simple. When we call or invoke a user-defined method, the program control transfer to the called method.

import java.util.Scanner;

public class EvenOdd

{

public static void main (String args[])

{

//creating Scanner class object

Scanner scan=new Scanner(System.in);

System.out.print("Enter the number: ");

//reading value from user

int num=scan.nextInt();

//method calling

findEvenOdd(num);

}

//user defined method

public static void findEvenOdd(int num)

{

//method body

if(num%2==0)

System.out.println(num+" is even");

else

System.out.println(num+" is odd");

}

}