Java methods

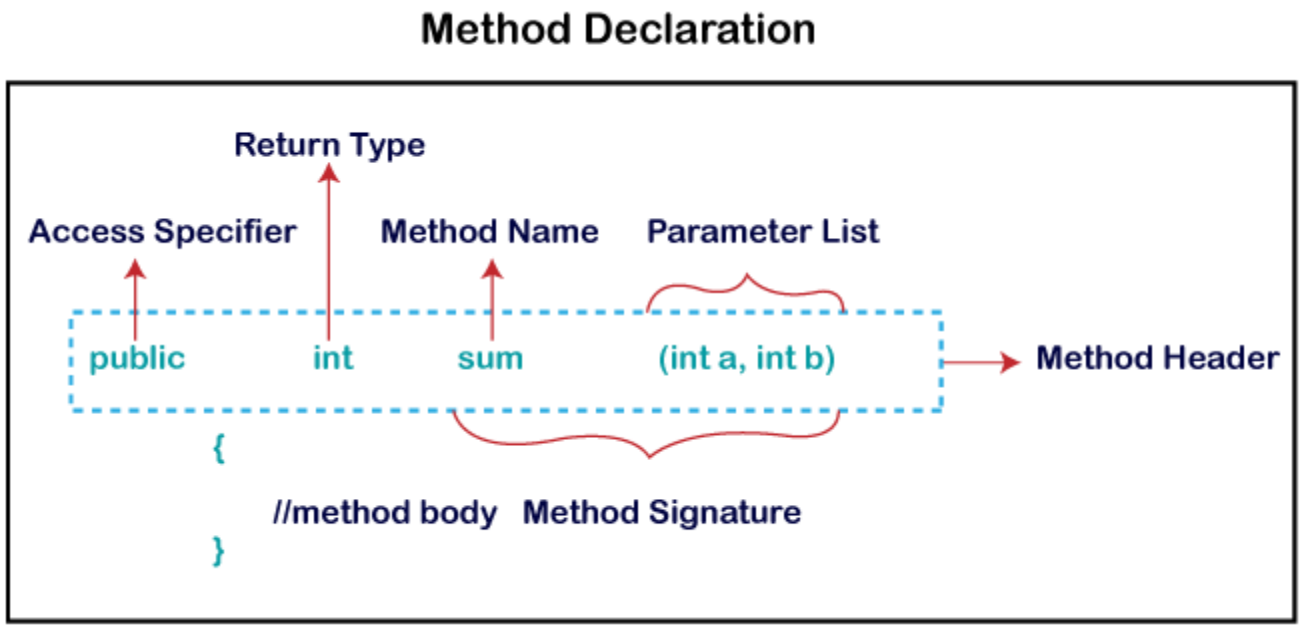
NAGARJUNA RAMINENI

What is a method in Java

- A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the **reusability** of code. We write a method once and use it many times. We do not require to write code again and again. It also provides the **easy modification** and **readability** of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

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**



Method Signature

- Every method has a **method signature**. It is a part of the method declaration. It includes the **method name** and **parameter list**.
- **Access Specifier:** Access specifier or modifier is the access type of the method. It specifies the visibility of the method. Java provides **four** types of access specifier:
  - **Public:** The method is accessible by all classes when we use public specifier in our application.
  - **Private:** When we use a private access specifier, the method is accessible only in the classes in which it is defined.
  - **Protected:** When we use protected access specifier, the method is accessible within the same package or subclasses in a different package.
  - **Default:** When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only.

Method Declaration…

- **Return Type:** Return type is a data type that the method returns. It may have a primitive data type, object, collection, void, etc. If the method does not return anything, we use void keyword.
- **Method Name:** It is a unique name that is used to define the name of a method. It must be corresponding to the functionality of the method. Suppose, if we are creating a method for subtraction of two numbers, the method name must be **subtraction().** A method is invoked by its name.
- **Parameter List:** It is the list of parameters separated by a comma and enclosed in the pair of parentheses. It contains the data type and variable name. If the method has no parameter, left the parentheses blank.
- **Method Body:** It is a part of the method declaration. It contains all the actions to be performed. It is enclosed within the pair of curly braces.

Naming a Method

- While defining a method, remember that the method name must be a **verb** and start with a **lowercase** letter. If the method name has more than two words, the first name must be a verb followed by adjective or noun. In the multi-word method name, the first letter of each word must be in **uppercase** except the first word. For example:
- **Single-word method name:** sum(), area()
- **Multi-word method name:** areaOfCircle(), stringComparision()
- It is also possible that a method has the same name as another method name in the same class, it is known as **method overloading**.

Types of Method

- There are two types of methods in Java:
  - Predefined Method
  - 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.
  - Each and every predefined method is defined inside a class. Such as **print()** method is defined in the **java.io.PrintStream** class. It prints the statement that we write inside the method. For example, **print("Java")**, it prints Java on the console.

Types of Method…

- 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.

```
//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");
}
```

Types of Method…

- 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.

```
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 the user
        int num=scan.nextInt();
        //method calling
        findEvenOdd(num);
}
```

Static Method

- A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. We can also create a static method by using the keyword **static** before the method name.
- The main advantage of a static method is that we can call it without creating an object. It can access static data members and also change the value of it. It is used to create an instance method. It is invoked by using the class name. The best example of a static method is the **main()** method.

```
static void show()
{
        System.out.println("It is an example of static method.");
}
```

```
public static void main(String[] args)
{
show();
}
```

Instance Method

- The method of the class is known as an **instance method**. It is a **non-static** method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class. Let's see an example of an instance method.

```
public class InstanceMethodExample
{
int s;
public static void main(String [] args)
{
//Creating an object of the class
InstanceMethodExample obj = new InstanceMethodExample();

//invoking instance method
System.out.println("The sum is: "+obj.add(12, 13));
}
}
```

```
//user-defined method because we have not used static keyword
public int add(int a, int b)
{
s = a+b;
//returning the sum
return s;
}
```

Instance Method

- There are two types of instance method:
  - **Accessor Method**
  - **Mutator Method**
- **Accessor Method:** The method(s) that reads the instance variable(s) is known as the accessor method. We can easily identify it because the method is prefixed with the word **get**. It is also known as **getters**. It returns the value of the private field. It is used to get the value of the private field.
- **Mutator Method:** The method(s) read the instance variable(s) and also modify the values. We can easily identify it because the method is prefixed with the word **set**. It is also known as **setters** or **modifiers**. It does not return anything. It accepts a parameter of the same data type that depends on the field. It is used to set the value of the private field.

Example of **Accessor & Mutator**

- Examples

Accessor

```
public int getId()
{
return Id;
}
```

Mutator

```
public void setRoll(int roll)
{
this.roll = roll;
}
```

Abstract Method

- The method that does not has method body is known as abstract method. In other words, without an implementation is known as abstract method. It always declares in the **abstract class**. It means the class itself must be abstract if it has abstract method. To create an abstract method, we use the keyword **abstract**.

  Syntax:

  ```
  abstract void method_name();
  ```

Abstract Method….

- Example:

```
abstract class Demo //abstract class
{
        //abstract method declaration
        abstract void display();
}
```

```
public class MyClass extends Demo{
        //method impelmentation
        void display() {
                System.out.println("Abstract method?");
        }
public static void main(String args[]) {
                //creating object of abstract class
                Demo obj = new MyClass();
                //invoking abstract method
                obj.display();
        }
}
```

Factory method

- It is a method that returns an object to the class to which it belongs. All static methods are factory methods.
  - For example, **NumberFormat obj = NumberFormat.getNumberInstance();**

```
double area=   755.0714285714286;
// Creating NumberFormat class object
NumberFormat obj=NumberFormat.getNumberInstance();
// store format info into obj
obj.setMaximumFractionDigits(2);
obj.setMinimumFractionDigits(7);
// apply the format to area value
String str=obj.format(area);
// displaying format to area value
System.out.println("Formatted area= "+str);
```

Recursion method

- Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method.
- It makes the code compact but complex to understand.

Syntax:

```
returntype methodname(){
//code to be executed
methodname();//calling same method
}
```

```
public class RecursionExample {
    static int factorial(int n){
        if (n == 1)
          return 1;
        else
          return(n * factorial(n-1));
    }

public static void main(String[] args) {
System.out.println("Factorial of 5 is: "+factorial(5));
}
}
```