**Types of user defined function**

In Java, user-defined functions are known as methods. These methods can be categorized based on various criteria, such as their purpose, scope, and behaviour . Here are some common types of user-defined methods in Java:

**1. Based on Functionality**

* **Helper Methods**: These methods perform a specific task and are often used by other methods within the class.

public class HelperExample {

private int add(int a, int b) {

return a + b;

}

}

* **Business Logic Methods**: These methods implement the core logic of the application.

public class BusinessLogicExample {

public double calculateDiscount(double price, double discountRate) {

return price \* (1 - discountRate);

}

}

**2. Based on Return Type**

* **Void Methods**: These methods do not return any value.

public void printMessage(String message) {

System.out.println(message);

}

* **Non-Void Methods**: These methods return a value of a specified type.

public int add(int a, int b) {

return a + b;

}

**3. Based on Parameters**

* **Parameterized Methods**: These methods take parameters as input.

public int multiply(int a, int b) {

return a \* b;

}

* **Parameterless Methods**: These methods do not take any parameters.

public void greet() {

System.out.println("Hello, World!");

}

**4. Based on Scope**

* **Instance Methods**: These methods belong to an instance of a class and require an object of the class to be invoked.

public class InstanceMethodExample {

public void display() {

System.out.println("Instance method");

}

}

* **Static Methods**: These methods belong to the class rather than any object instance and can be called without creating an instance of the class.

public class StaticMethodExample {

public static void display() {

System.out.println("Static method");

}

}

**5. Based on Access Modifiers**

* **Public Methods**: These methods can be accessed from any other class.

public void publicMethod() {

System.out.println("Public method");

}

* **Private Methods**: These methods can be accessed only within the class they are declared.

private void privateMethod() {

System.out.println("Private method");

}

* **Protected Methods**: These methods can be accessed within the same package and by subclasses.

protected void protectedMethod() {

System.out.println("Protected method");

}

* **Default (Package-Private) Methods**: These methods can be accessed only within the same package.

void defaultMethod() {

System.out.println("Default method");

}

**6. Based on Use Case**

* **Constructor Methods**: Special methods used to initialize objects. They have the same name as the class and do not have a return type.

public class ConstructorExample {

public ConstructorExample() {

System.out.println("Constructor method");

}

}

* **Getter Methods (Accessors)**: These methods are used to retrieve the value of a private field.

public class GetterExample {

private int value;

public int getValue() {

return value;

}

}

* **Setter Methods (Mutators)**: These methods are used to set the value of a private field.

public class SetterExample {

private int value;

public void setValue(int value) {

this.value = value;

}

}

* **Utility Methods**: Commonly used static methods in utility classes.

public class MathUtils {

public static int add(int a, int b) {

return a + b;

}

}

* **Recursive Methods**: Methods that call themselves to solve a problem.

public int factorial(int n) {

if (n == 0) return 1;

return n \* factorial(n - 1);

}