**jump Statements**

Jump statements are used to **change the normal flow of execution** in loops or methods.

**1. break**

* Used to **exit a loop or switch** immediately.
* Control jumps out of the loop/switch block.

public class BreakExample {

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

if (i == 5) {

break; // exit loop when i=5

}

System.out.println(i);

}

}

}

**2. continue**

* Used to **skip the current iteration** and move to the **next iteration** of the loop.

Example:

public class ContinueExample {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

if (i == 3) {

continue; // skip 3

}

System.out.println(i);

}

}

}

**Java Methods**

A **method** is a block of code that performs a specific task.

* Methods help us **reuse code** (write once, use many times).
* They improve **readability, modularity, and maintainability**.

**Method Syntax**

returnType methodName(parameters) {

// body of method

return value; // (if returnType is not void)

}

* **returnType** → Type of value method returns (int, String, void, etc.).
* **methodName** → Name of method (should follow camelCase).
* **parameters** → Input values (optional).
* **body** → Code that runs when method is called.

Types of Methods

**(A) Predefined Methods**

* Already defined in Java libraries.

public class PredefinedExample {

public static void main(String[] args) {

String name = "Hello Java";

System.out.println(name.length()); // length() is predefined method

}

}

**(B) User-Defined Methods**

Created by programmers.

Example 1: No return, no parameters

public class Example1 {

static void greet() {

System.out.println("Hello, Welcome to Java!");

}

public static void main(String[] args) {

greet(); // method call

}

}

Example 2: With return type and parameters

public class Example2 {

static int add(int a, int b) {

return a + b;

}

public static void main(String[] args) {

int sum = add(10, 20);

System.out.println("Sum = " + sum);

}

}

**3. Method Overloading**

* Same method name, different parameter lists.
* Decided at **compile time**.

public class OverloadExample {

static int add(int a, int b) {

return a + b;

}

static double add(double a, double b) {

return a + b;

}

public static void main(String[] args) {

System.out.println(add(5, 10)); // calls int method

System.out.println(add(5.5, 10.5)); // calls double method

}

}  
  
**4. Static vs Non-Static Methods**

* **Static methods** → belong to class, can be called without object.
* **Non-static methods** → require object to call.

public class StaticVsNonStatic {

static void staticMethod() {

System.out.println("I am static");

}

void nonStaticMethod() {

System.out.println("I am non-static");

}

public static void main(String[] args) {

staticMethod(); // called directly

StaticVsNonStatic obj = new StaticVsNonStatic();

obj.nonStaticMethod(); // called using object

}

}