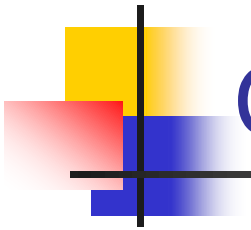
CÔNG NGHỆ JAVA

Trường: Đại học Giao thông vận Tải

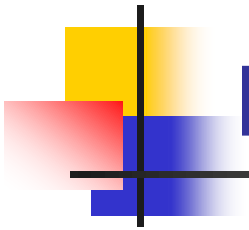
Khoa: Công nghệ thông tin



CÔNG NGHỆ JAVA

Chương 2:

Cơ sở của ngôn ngữ Java



Nội dung

- Nội dung
  - Các khái niệm cơ bản
    - Chú thích
    - Hằng, biến và kiểu dữ liệu
    - Các phép toán với kiểu dữ liệu
    - Vào ra dữ liệu và cấu trúc chương trình đơn giản trong Java
  - Cấu trúc điều khiển
    - Cấu trúc điều khiển rẽ nhánh
    - Cấu trúc điều khiển lặp
  - Mở rộng
    - Mảng/String
  - Khai báo phương thức trong Java

Chương 2:

1. Các khái niệm cơ bản

Các khái niệm cơ bản

- **Chú thích**
  - Dùng để giải thích cho mỗi đoạn chương trình.
  - Dùng để hỗ trợ xây dựng các tài liệu thông qua Javadoc.
  - Các chú thích sẽ được trình biên dịch bỏ qua, không ảnh hưởng đến hiệu năng của chương trình.
  - Các cách thức để chú thích:
    - Chú thích theo từng dòng với //
    - Chú thích theo một khối

```
/*
*/
```

    - Chú thích để hỗ trợ sinh tài liệu tự động

```
/**
*/
```

  - Trong Java, để sinh tài liệu tự động, chúng ta còn cần dùng đến các thẻ như @version, @param, etc…

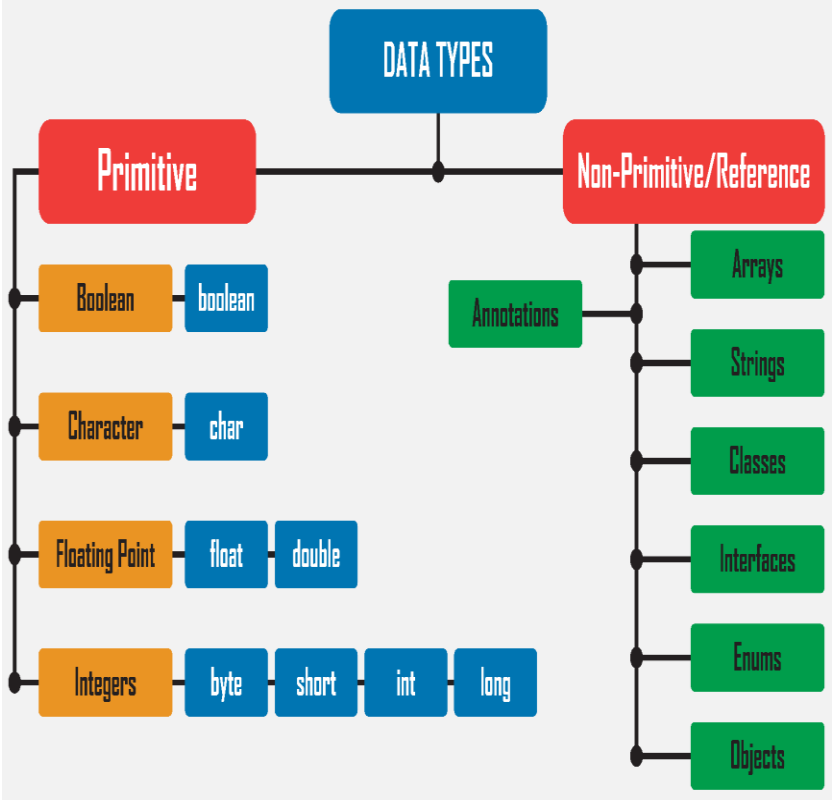
Các khái niệm cơ bản

- **Từ khóa**
  - Là các từ dành riêng được quy ước bởi ngôn ngữ lập trình.
  - Mỗi từ khóa đảm nhiệm một chức năng cụ thể.
  - Các từ khóa không được dùng để khai báo biến, hằng, đối tượng. Trong ngôn ngữ lập trình Java, có khoảng 50 từ khóa thường xuyên được sử dụng.
    - Từ khóa định nghĩa kiểu dữ liệu: *byte, short, int, long, float, double, char, boolean, void.*
    - Từ khóa cho cấu trúc điều khiển: *if, else, switch, case, default, for, while, do.*
    - Từ khóa điều khiển: *break, continue, return.*
    - Từ khóa miêu tả đặc tính: *private, public, protected, final, static, abstract, synchronized, volatile, …*
    - Từ khóa liên quan đến đối tượng: *class, interface, new, extends, implements, instanceof, this, super.*
    - Các từ khóa kiểm soát lỗi: *try, catch, finally, throw, throws.*
    - Các từ khóa khác: *import, package, true, false, …*

Các khái niệm cơ bản

- **Các kiểu dữ liệu cơ bản trong Java**

Kiểu	Kích thước	Trị mặc định	Giá trị	Ví dụ
byte	1 byte	0	[-128; 127]	byte x = 56;
short	2 byte	0	[-32768; 32767]	short x = 1000;
int	4 byte	0	[-2,147,483,648; 2,147,483,647]	int x = 150000;
long	8 byte	0	[-9,223,372,036,854,775,808 (2^{63}); 9,223,372,036,854,775,807 ($2^{63}-1$)]	long x = 653535535632l;
float	4 byte	0.0	$1.4E^{-45}$; $3.4028235E^{38}$	float x = 5.421f
double	8 byte	0.0	4.9E-34; 1.797693134862315 7E+308	double x = 5.421d
char	2 byte		0; 65,535	char c = 'b';
boolean	1 bit	false	true (1), false (0)	boolean b = true;

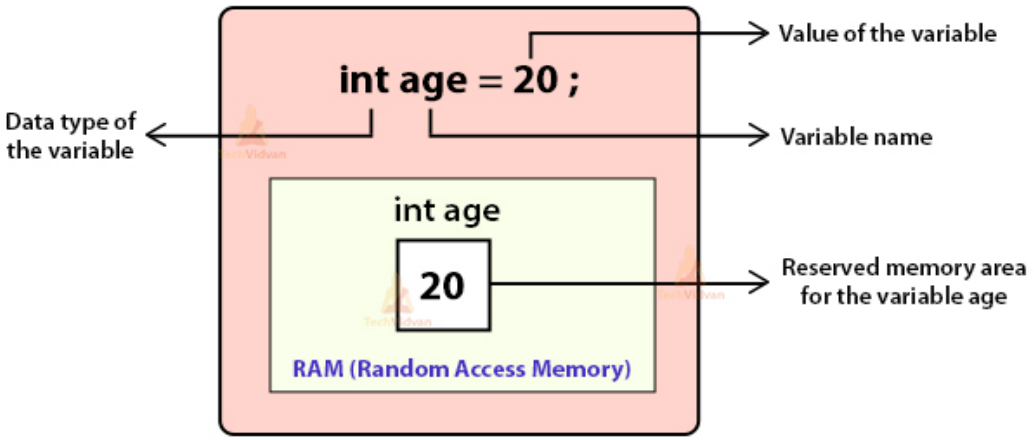


Các khái niệm cơ bản

- **Toán tử trong Java**
  - Java cung cấp các toán tử đối với các kiểu dữ liệu cơ bản.
  - Các toán tử được thực hiện trên các kiểu dữ liệu tương ứng. Java sẽ ép kiểu dữ liệu khi thực hiện các toán tử với các dữ liệu khác nhau.
  - Các phép toán phức tạp được Java cung cấp thông qua thư viện Math: *abs, sqrt, sin, cos, tan, etc…*
  - Toán tử số học: *+, -, *, /, %, ++, --*
  - Toán tử quan hệ: >, <, >=, <=, !=, ==
  - Toán tử logic: &&, ||, !
  - Toán tử bit: ~, &, |, ^, >>, <<, &=, |=, ^=, >>=, <<=
  - Toán tử gán: =, +=, -=, *=, /=, %=
  - Biểu thức ghép: condition ? true_value : false_value;

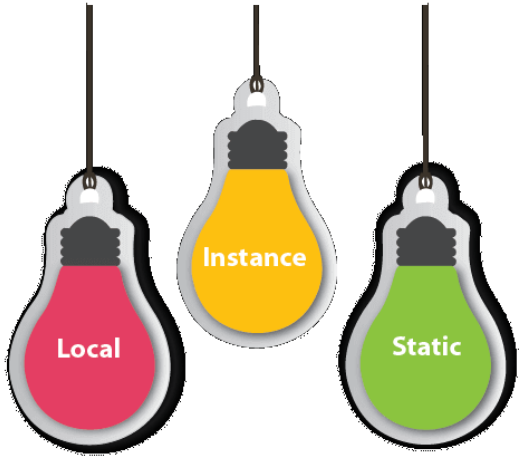
Các khái niệm cơ bản

- **Biến**
  - Là tên của một vùng nhớ được cấp phát để lưu trữ dữ liệu trong chương trình. Giá trị của biến thay đổi tùy theo chương trình.
  - Được truy cập thông qua tên biến nhằm thay thế cho việc truy cập trực tiếp thông qua địa chỉ vùng nhớ.
  - Cách khai báo biến trong Java
    - *data_type variable_name*
    - *data_type variable_name = default_value*
    - *i.e.*
      - *int myNum = 15;*
      - *boolean myBool;*



Các khái niệm cơ bản

- **Biến**
  - Trong Java cung cấp 3 kiểu biến
    - Biến cục bộ (local variable)
    - Biến thực thể (instance variable)
    - Biến tĩnh (static variable)
    - Ngoài ra, biến tham số [parameter variable]
  - Phạm vi hoạt động




```
package example;

public class MyClass {
    public int my_property;

    public void myMethod(int i) {
        String s = "Hello";
        if (true) {
            int j = 5;
            System.out.print(j);
        }
        System.out.print(s);
        System.out.print(i);
        System.out.print(j); //Lỗi biên dịch
    }
}
```

biến thực thể [thuộc tính của đối tượng]

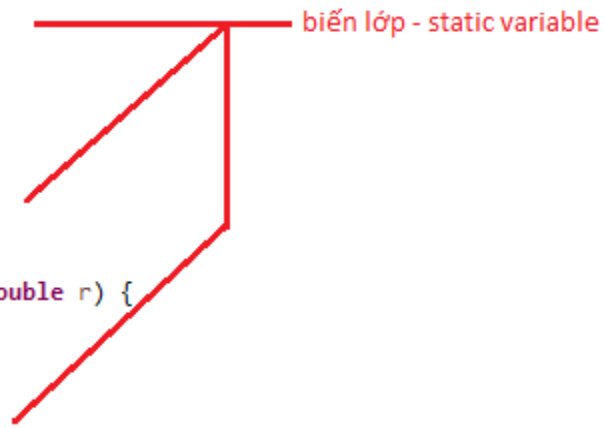
biến cục bộ

tham số - biến hình thức

Các khái niệm cơ bản

- **Biến**
  - Phạm vi hoạt động của static variables
    - Biến static được khai báo trong một class với từ khóa "static", phía bên ngoài các phương thức, constructor và block.
    - Sẽ chỉ có duy nhất một bản sao của các biến static được tạo ra và được dùng chung cho các đối tượng sinh ra.

```
package example;

public class Circle {
    private double x;
    private double y;
    private double r;
    private static int NUM_CIRCLES = 0;

    public Circle() {
        this.x = 0.0;
        this.y = 0.0;
        this.r = 0.0;
        Circle.NUM_CIRCLES++;
    }

    public Circle(double x, double y, double r) {
        this.x = x;
        this.y = y;
        this.r = r;
        Circle.NUM_CIRCLES++;
    }
}
```



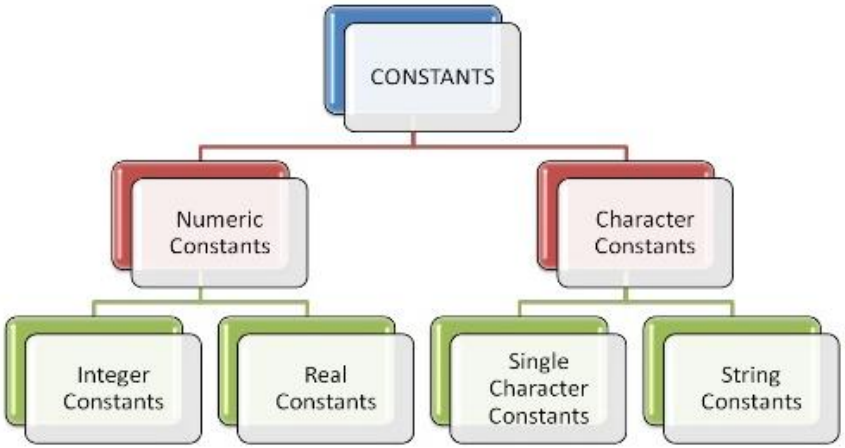
Các khái niệm cơ bản

- **Biến**
  - Quy tắc đặt tên biến
    - Tên biến phân biệt hoa thường.
    - Có thể chứa ký tự, số, và một số các ký tự đặc biệt: *_, $*
    - Không bắt đầu bằng số và không được đặt tên trùng từ khóa.
    - Không chứa khoảng trắng, ký hiệu toán học.
    - Khuyến nghị:
      - Đặt tên biến có ý nghĩa ➔ giúp cho việc đọc và debug.
      - Bắt đầu bằng chữ thường. i.e. *dateOfBirth, address, description, placeOfBirth*
      - Biến tạm trong phạm vi hẹp có thể đặt: *a, i, j, …*
      - Tránh kết hợp nhiều ngôn ngữ.
    - Java naming conventions:

Name	Convention
Class Name	Should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread, ArrayList, StringBuffer etc.
Interface Name	Should start with uppercase letter and be an adjective e.g. Runnable, Remote, List, ActionListener etc.
Method Name	Should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
Variable Name	Should start with lowercase letter e.g. firstName, lastName, orderNumber etc.
Package Name	Should be in lowercase letter e.g. java.lang, java.sql, java.util etc.
Constants Name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Các khái niệm cơ bản

- **Hằng**
  - Là giá trị không thay đổi trong toàn bộ chương trình.
  - Trong Java, để hỗ trợ cho các hằng số, chúng ta sử dụng từ khóa *final* và *static*.
  - i.e.
    - *public static final double PI = 3.14f;*
  - Chú ý:
    - Với các hằng là đối tượng, giá trị thuộc tính có thể thay đổi, chỉ có con trỏ tới đối tượng là không đổi.



Các khái niệm cơ bản

- **Cấu trúc chương trình Java đơn giản**

```java
// 1. Khai báo package: cấu trúc chương trình
package example;

// 2. Khai báo các thư viện cần dùng
import java.util.Scanner;

// 3. Định nghĩa lớp
public class SimpleClass {

    // 3.1. Khai báo các hằng số
    public static final double PI = 3.14f;

    // 3.2. Định nghĩa chương trình chính
    public static void main(String[] args) {
        // 3.2.1. Khai báo biến
        double R = 10.0f;
        double S = 0.0f;

        // 3.2.2. Nhập dữ liệu cho biến đầu vào với Scanner
        Scanner sc = new Scanner(System.in);
        System.out.print("R = ");
        R = sc.nextDouble();

        // 3.2.3. Thuật toán
        S = SimpleClass.PI * R * R;

        // 3.2.4. Hiển thị kết quả đầu ra với System.out
        System.out.println("Diện tích hình tròn là: " + S);

        // 3.2.5. Đóng các connections
        sc.close();
    }

}
```

Các khái niệm cơ bản

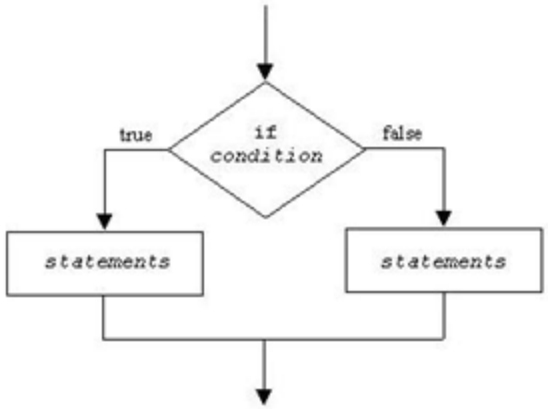
- **Vào ra dữ liệu với thiết bị vào ra chuẩn**
  - Nhập dữ liệu từ bàn phím thông qua Scanner
    - ***import java.util.Scanner;***
    - *Scanner sc =* ***new Scanner(System.in);***
    - *double R = sc.nextDouble();*
    - *sc.close();*
  - Xuất dữ liệu thông qua System.out.
    - *System.out.print("R = ");*
    - *System.out.println("Diện tích hình tròn là: " + S);*
    - *System.out.printf("Diện tích hình tròn là: %f", S);*

Chương 2:

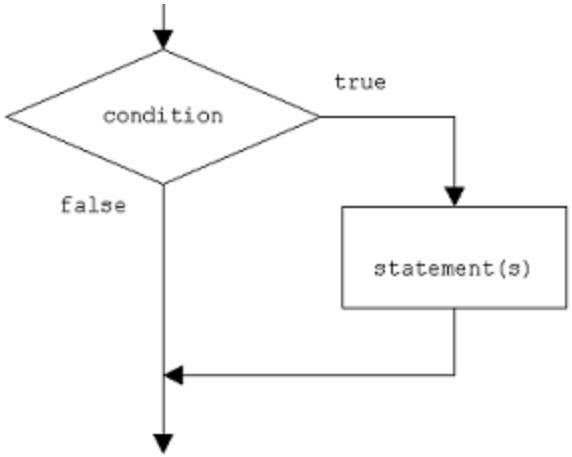
2. Cấu trúc điều khiển trong Java

Cấu trúc điều khiển

- **Cấu trúc điều khiển rẽ nhánh**
  - Cho phép kiểm soát rẽ nhánh chương trình dựa vào điều kiện.
  - Java hỗ trợ các cách tiếp cận:



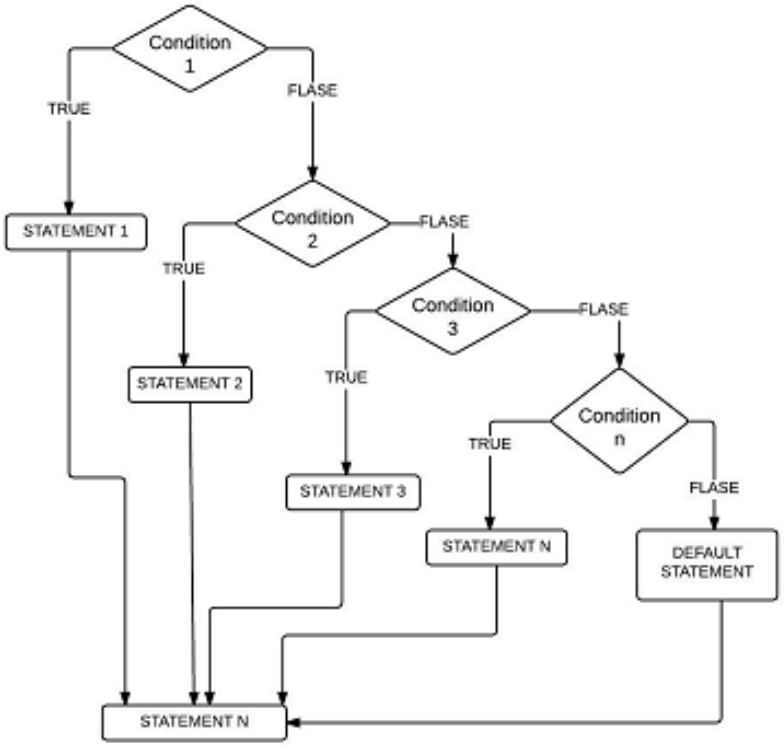

```
// start
if (condition){
    // statements - true
}else{
    // statements - false
}
// continue ...
```




```
// start
if (condition){
    // statements - true
}
// continue ...
```

Cấu trúc điều khiển

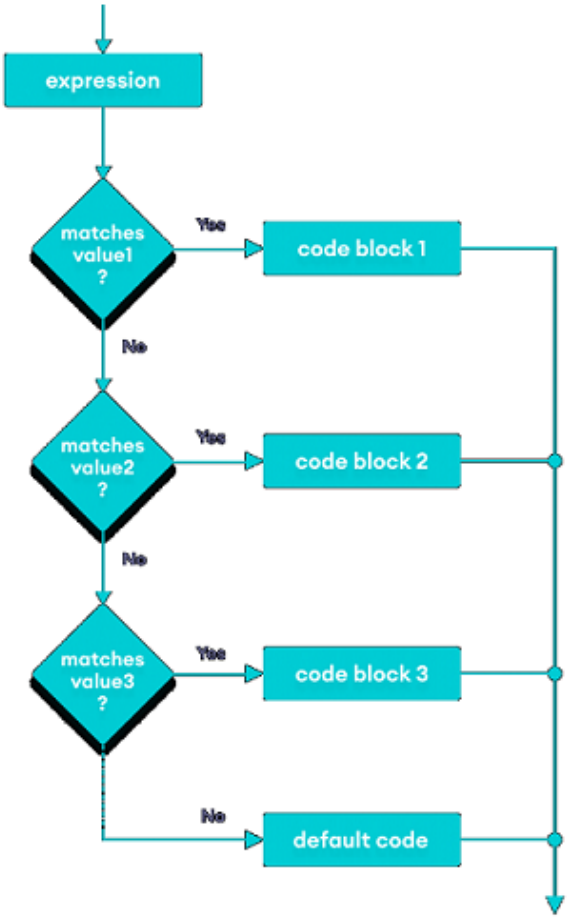
- **Cấu trúc điều khiển rẽ nhánh**
  - Cấu trúc rẽ nhánh lồng nhau




```
// start
if (condition_1){
    if (condition_2){
        // statements - true - 1 & 2
    }else{
        // statements - true - 1 & false - 2
    }
}else{
    if (condition_3){
        // statements - false - 1 & true - 3
    }else{
        if (condition_4){
            // statements - false - 1 & false - 3 & true - 4
        }
    }
}
// continue ...
```

Cấu trúc điều khiển

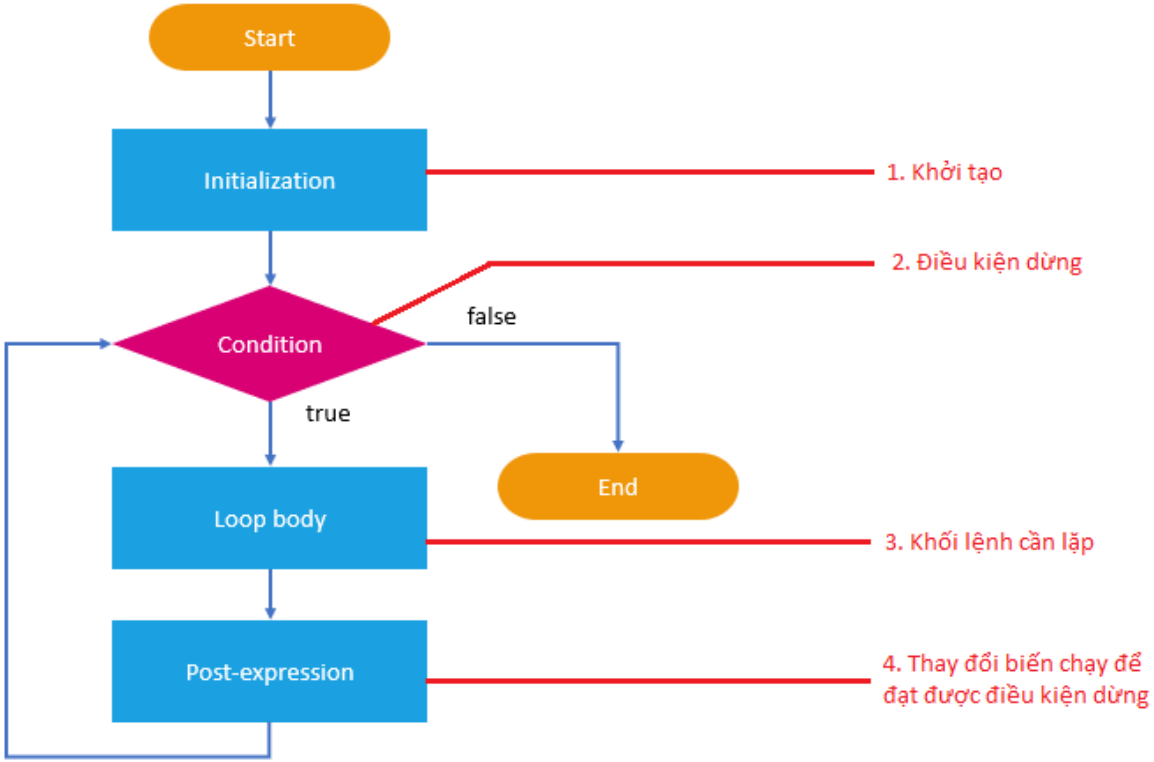
- **Cấu trúc điều khiển rẽ nhánh**
  - Cấu trúc rẽ nhánh với switch




```
// start
switch(today % 7){
case 0:
    System.out.println("Chủ nhật");
    break;
case 1:
    System.out.println("Thứ hai");
    break;
case 2:
    System.out.println("Thứ ba");
    break;
case 3:
    System.out.println("Thứ tư");
    break;
case 4:
    System.out.println("Thứ năm");
    break;
case 5:
    System.out.println("Thứ sáu");
    break;
default:
    System.out.println("Thứ bảy");
    break;
}
// continue ...
```

Cấu trúc điều khiển

- **Cấu trúc điều khiển lặp**
  - Cho phép chương trình thực hiện lặp hữu hạn một khối lệnh.
  - Nếu số lần lặp xác định: *for*
  - Nếu số lần lặp chưa xác định: *while, do … while*



Cấu trúc điều khiển

- **Cấu trúc điều khiển lặp**
  - Cấu trúc lặp với *for*

```
/*
     for (initialization; condition ; post_expression) {
        statements;
     }
 */
// i.e.
int i;
for (i = 0; i < 10; i++) {
    System.out.println(i);
}
```

```
/*
     for (initialization: list) {
        statements;
     }
 */
// i.e.
int[] a = {1, 4, 6, 12, 9, 20};
for (int value: a) {
    System.out.println(value);
}
```

Cấu trúc điều khiển

- **Cấu trúc điều khiển lặp**
  - Cấu trúc lặp với *while, do … while*

```
/*
     initialization;
     while (condition){
        statements;
        post_expression;
     }
 */

int i = 0;
while (i<10){
    System.out.println(i);
    i++;
}
```

```
/*
     initialization;
     do {
        statements;
        post_expression;
     } while (condition);
*/

int i = 0;
do{
    System.out.println(i);
    i++;
}while (i>9);
```

Chương 2:

3. Các kiểu dữ liệu mở rộng

Các kiểu dữ liệu mở rộng

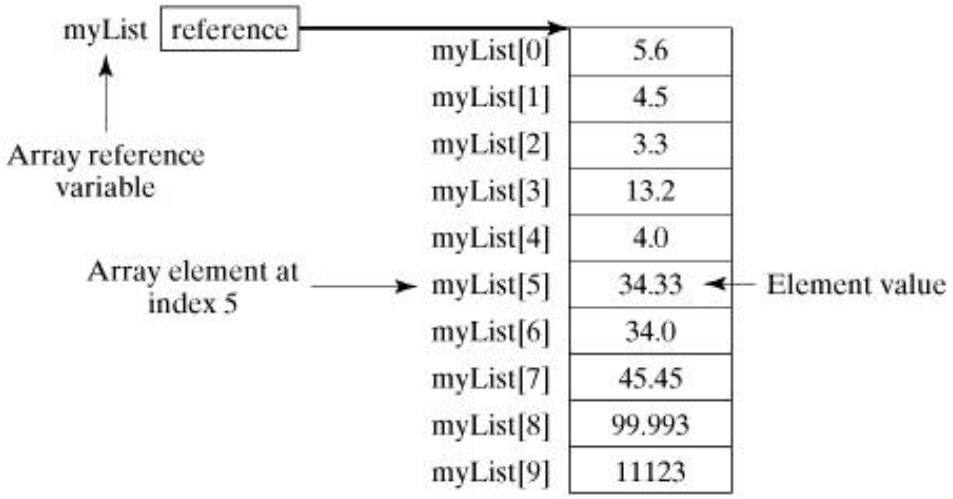
- **Mảng**
  - Cấu trúc dữ liệu tĩnh cho phép lưu trữ một tập các phần tử có cùng kiểu dữ liệu.
  - Khai báo biến mảng:
    - kiểu dữ liệu
    - tên biến mảng
    - các phần tử mặc định
  - Khai báo trong Java
    - *data_type[] variable_name;*
    - *data_type[] variable_name = {element_1, element_2, …, element_n}*
    - *cấp phát bộ nhớ: variable_name = new data_type[10];*
    - *i.e.*

```
double[] d_array;
int[] i_array = {1, 2, 4, 9, 100};
d_array = new double[10];
System.out.println(d_array.length);
System.out.println(i_array.length);
```

KẾT QUẢ:
```
10
5
```

Các kiểu dữ liệu mở rộng

- **Mảng**
  - Quản lý bộ nhớ



  - Truy cập các phần tử
    - Thông qua tên biến mảng và chỉ số bắt đầu từ 0.
  - Các phép toán trên mảng
    - Thực thi thông qua từng phần tử với các toán tử cho kiểu dữ liệu tương ứng.
    - i.e.
      - *System.out.println(d_array[0]);*
      - *double sum = d_array[0] + d_array[1];*

Các kiểu dữ liệu mở rộng

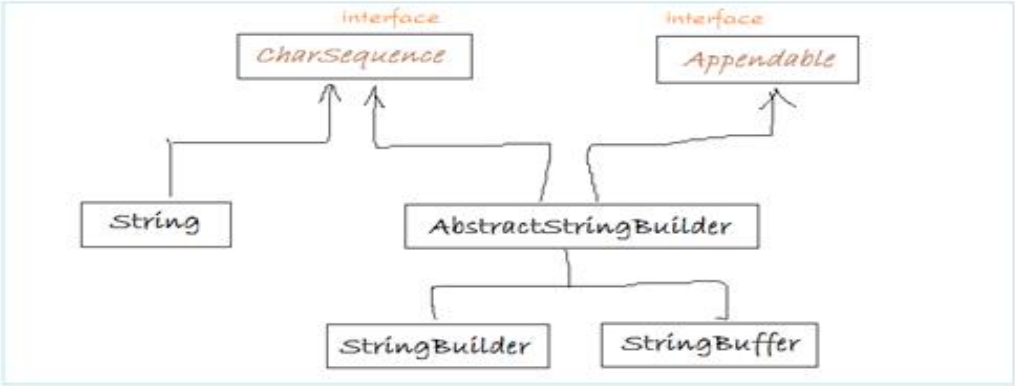
- **Mảng**

```
package example;

public class TestArray {

    public static void main(String[] args) {
        double[] myList = { 1.9, 2.9, 3.4, -3.5, 1.2 };

        // Hiển thị danh sách các phần tử
        for (int i = 0; i < myList.length; i++) {
            System.out.print(myList[i] + " ");
        }
        System.out.println();

        // Tính tổng các phần tử
        double sum = 0.0;
        for (int i = 0; i < myList.length; i++) {
            sum += myList[i];
        }
        System.out.printf("Tổng là: %.2f\n", sum);

        // Tìm phần tử lớn nhất
        double max = myList[0];
        for (double value: myList) {
            if (value > max)
                max = value;
        }
        System.out.println("Giá trị lớn nhất là: " + max);

    }

}
```

Các kiểu dữ liệu mở rộng

- **String**
  - Là lớp cho phép xử lý một chuỗi các giá trị char.
  - i.e.
    - *char[] ch={'H','e','l','l','o',' ','W','o','r','l','d'};*
    - *String s = "Hello World";*
  - Các đặc điểm của String
    - Immutable
    - CharSequence
    - Comparable
    - Serialization

	String	StringBuffer	StringBuilder
Storage	String pool	Heap	Heap
Modifiable	No(immutable)	Yes (mutable)	Yes (mutable)
Thread safe	Yes	Yes	No
Synchronized	Yes	Yes	No
Performance	Fast	Slow	Fast

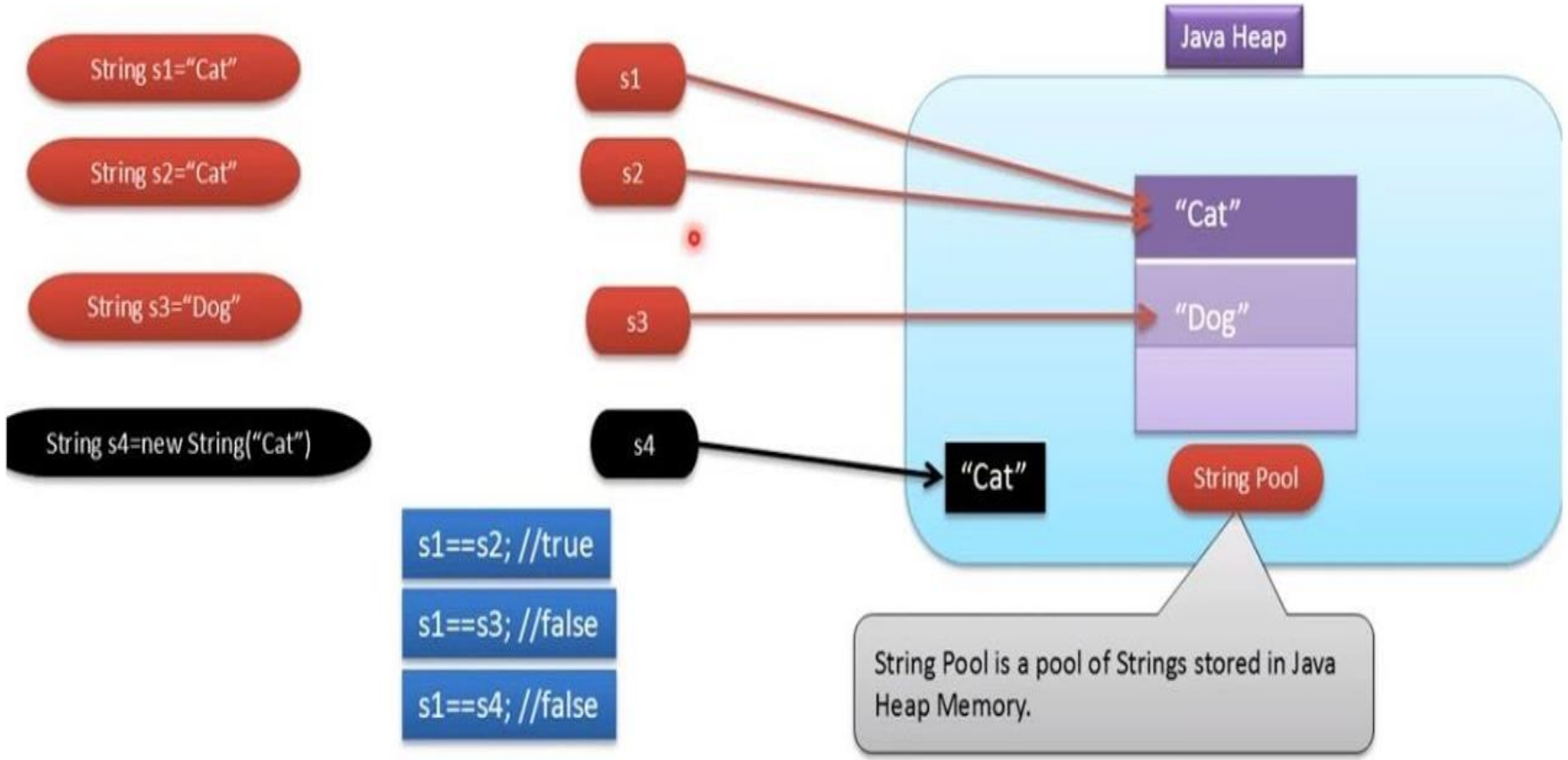


Các kiểu dữ liệu mở rộng

- **String**
  - Khai báo và quản lý bộ nhớ
    - Thông qua String Pool với việc gán giá trị cho biến String.
    - i.e.
      - String s1 = “abc”;
      - String s2 = “abc;
      - String s3 = “def”;
    - Thông qua bộ nhớ chung (HEAP) với toán tử new.
    - i.e.
      - String s4 = new String(“abc”);
      - String s5 = new String(“def”);
  - String Pool: là cơ chế quản lý bộ nhớ chung cho các String thông qua gán giá trị nhằm giảm bớt không gian bộ nhớ.
    - Khi thực hiện các phép toán hoặc new, Java sẽ cung cấp bộ nhớ mới trên Heap, không đưa vào String Pool.
    - Muốn đưa một String từ Heap vào Pool dùng lệnh: intern();

Các kiểu dữ liệu mở rộng

- **String Pool**



Các kiểu dữ liệu mở rộng

- **String**

```
public class TestString {

    public static void main(String[] args) {
        String s1 = "abc";
        String s2 = "abc";
        String s3 = new String("abc");
        System.out.println(s1);                     // abc
        System.out.println(s2);                     // abc
        System.out.println(s3);                     // abc
        System.out.println(s1==s2);                 // true
        System.out.println(s1==s3);                 // false

        String s4 = "ABC";
        System.out.println(s4);                     // ABC
        System.out.println(s4.toLowerCase());       // abc
        System.out.println(s1==s4.toLowerCase());   // false
        String s5 = s4.toLowerCase().intern();
        System.out.println(s1==s5);                 // true

        String s6 = "abcdef";
        String s7 = "def";
        String s8 = s1 + s7;
        System.out.println(s6);                     // abcdef
        System.out.println(s8);                     // abcdef
        System.out.println(s6==s8);                 // false
    }

}
```

Các kiểu dữ liệu mở rộng

- **Một số hàm thông dụng trong String**
  - split
    - Cắt chuỗi thành các chuỗi nhỏ theo điều kiện.
    - *String [ ] split (String regex, int limit)*

```
String s = "myemail@gmail.com@ccc";
String[] sArr_1 = s.split("@", 2);
for (String s_e: sArr_1){
    System.out.println(s_e);    // myemail; gmail.com@ccc
}

String[] sArr_2 = s.split("@");
for (String s_e: sArr_2){
    System.out.println(s_e);    // myemail; gmail.com; ccc
}
```

  - trim
    - Loại bỏ khoảng trắng trước và sau của chuỗi.
  - indexOf
    - Tìm vị trí đầu tiên của chuỗi con trong chuỗi.
  - …

Chương 2:

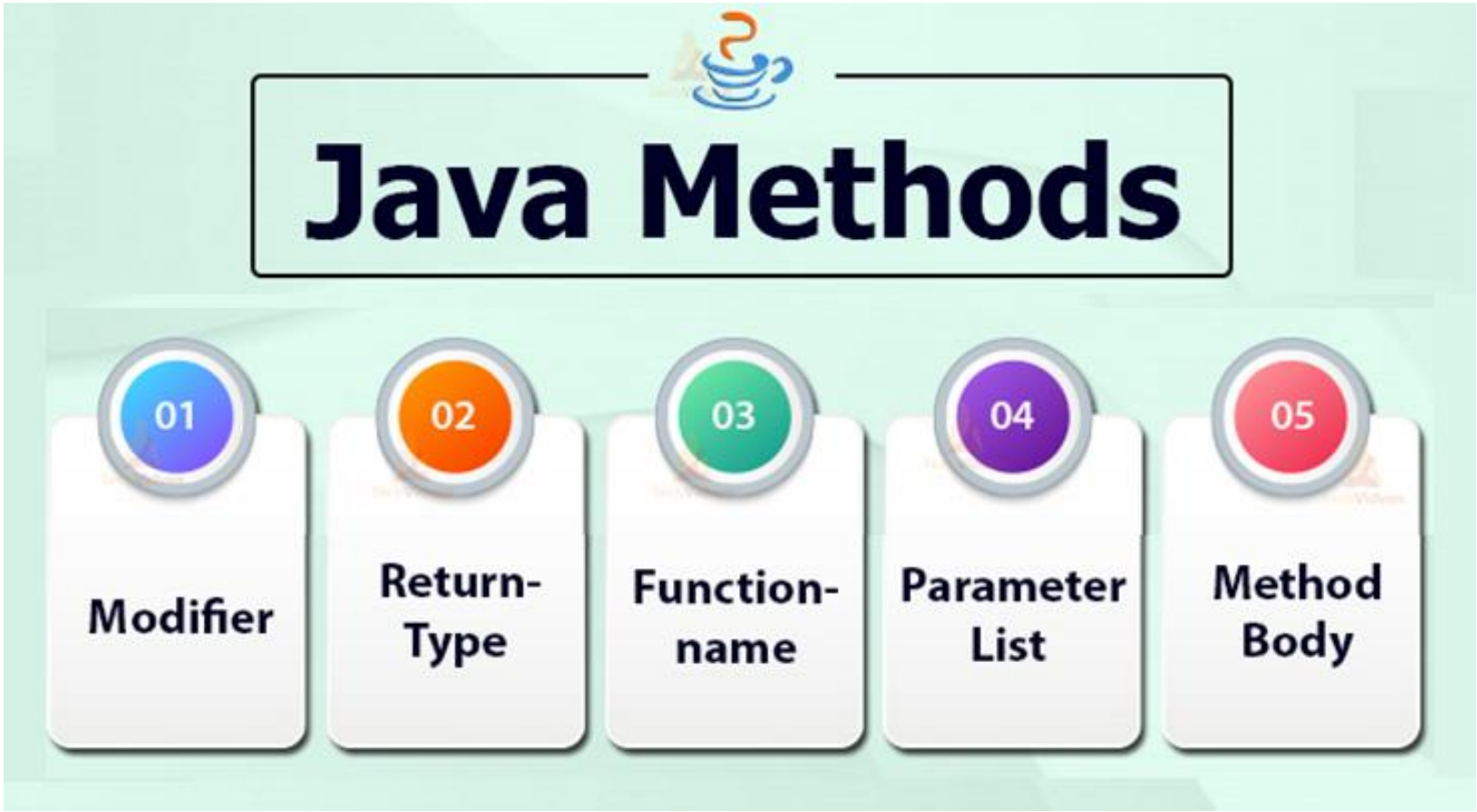
4. Khai báo phương thức trong Java

Khai báo phương thức

- **Phương thức**
  - Là một tập hợp các câu lệnh nhằm thực hiện một tác vụ nào đó trong chương trình.
  - Đây là cách tiếp cận chia để trị: phân rã bài toán thành các bài toán nhỏ để thực hiện.
  - Ưu điểm:
    - Tái sử dụng các code chương trình.
    - Tránh bị trùng lặp code.
    - Dễ dàng cho việc kiểm soát lỗi, sửa đổi chương trình.
  - Trong Java, phương thức bao gồm 2 loại:
    - Phương thức của lớp – tiếp cận hướng lập trình cấu trúc.
    - Phương thức của đối tượng – tiếp cận hướng OOP.
  - Chú ý:
    - Java là ngôn ngữ OOP ➔ tránh sử dụng phương thức lớp: khó mở rộng sau này.

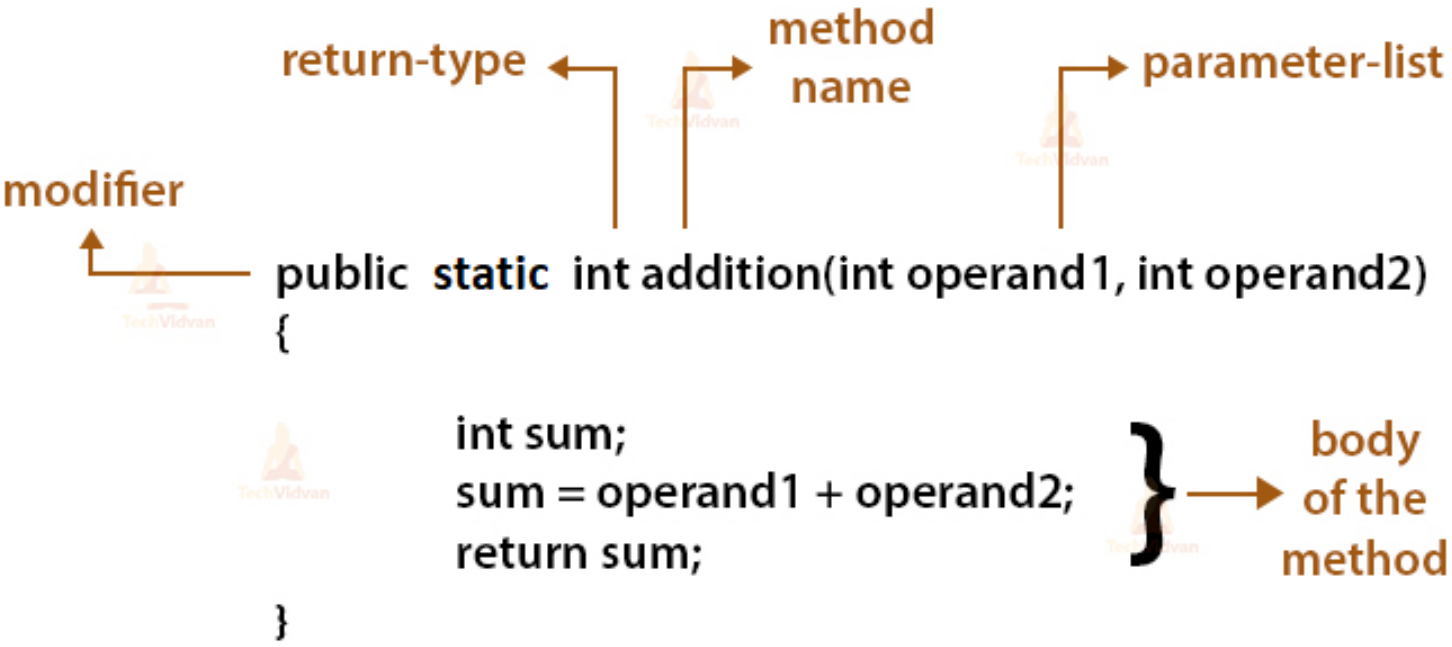
Khai báo phương thức

- **Phương thức**
  - Cấu trúc của một phương thức



Khai báo phương thức

- **Phương thức**
  - Java chỉ hỗ trợ truyền tham số theo tham trị.
  - Ví dụ về một phương thức



Khai báo phương thức

- **Phương thức**

```
public class TestArray {

    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, -3.5, 1.2 };

        // Hiển thị danh sách các phần tử
        for (int i = 0; i < myList.length; i++) {
            System.out.print(myList[i] + " ");
        }
        System.out.println();

        // Tính tổng các phần tử
        double sum = 0.0;
        for (int i = 0; i < myList.length; i++) {
            sum += myList[i];
        }
        System.out.printf("Tổng là: %.2f\n", sum);

        // Gọi hàm trong Java
        double max = TestArray.maxList(myList);
        System.out.println("Max là: " + max);
    }

    public static double maxList(double[] list){
        double max = list[0];
        for (double d : list) {
            if (max < d){
                max = d;
            }
        }
        return max;
    }

}
```

Bài tập

- **Các bài tập với câu lệnh điều khiển**

1. Viết chương trình giải phương trình bậc 2.
2. Cho dãy Fibonacci như sau: 1, 1, 2, 3, 5, 8, 13, …
   - Viết chương trình tính giá trị phần tử thứ n của dãy.
   - Viết chương trình cho phép với giá trị đầu vào n, tìm phần tử trong dãy Fibonacci có giá trị gần nhất với n.
3. Xây dựng chương trình tính $S = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \cdots + \frac{n!x^n}{n!}$

- **Các bài tập với mảng**

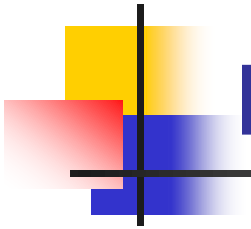
1. Cho tập hợp số nguyên, hãy xây dựng chương trình tính
   - Đếm số phần tử dương không chia hết cho 3 trong dãy.
   - Tính tổng các phần tử nằm trong khoảng [-5, 25] và trung bình cộng của chúng.
   - Xác định phần tử lớn nhất trong dãy chia hết cho 3.
   - Sắp xếp dãy số để các phần tử có giá trị tuyệt đối tăng dần.
   - Loại bỏ các phần tử chia hết cho 5 nhưng không chia hết cho 3 khỏi dãy số.
   - Hiển thị kết quả các bước ra màn hình.

Bài tập

- **Các bài tập với String**

1. Viết chương trình cho phép loại bỏ các ký tự ‘a’ và đảo ngược chuỗi sau khi đã loại bỏ.
   - S_input = “Lap trinh Java khong don gian”
   - S_output = “nig nod gnohk vJ hnirt pL”
2. Viết chương trình nhập chuỗi String là các số cách nhau bởi dấu cách và kiểm tra xem có thỏa mãn không, nếu đúng thì có bao nhiêu số.
   - S_input = “2 34.5 -12.9 0 12 1.98”
   - Output: true, có 6 số.
   - S_input = “2 34.5 -12a.9 0 12 1b.98”
   - Output: false.

- **Chú ý**
  - Các bài tập cần xây dựng phương thức static để trong sáng code.
  - Chú ý cách đặt tên biến, hàm, lớp, etc...
  - Làm quen với phím tắt và cách thức code trong Eclipse.



Bài tập

- **Bài tập mở rộng**

1. Viết chương trình trò chơi Sodoku bằng ngôn ngữ lập trình Java.

- **Chú ý**
  - Các bài tập cần xây dựng phương thức static để trong sáng code.
  - Chú ý cách đặt tên biến, hàm, lớp, etc...
  - Làm quen với phím tắt và cách thức code trong Eclipse.

CÔNG NGHỆ JAVA