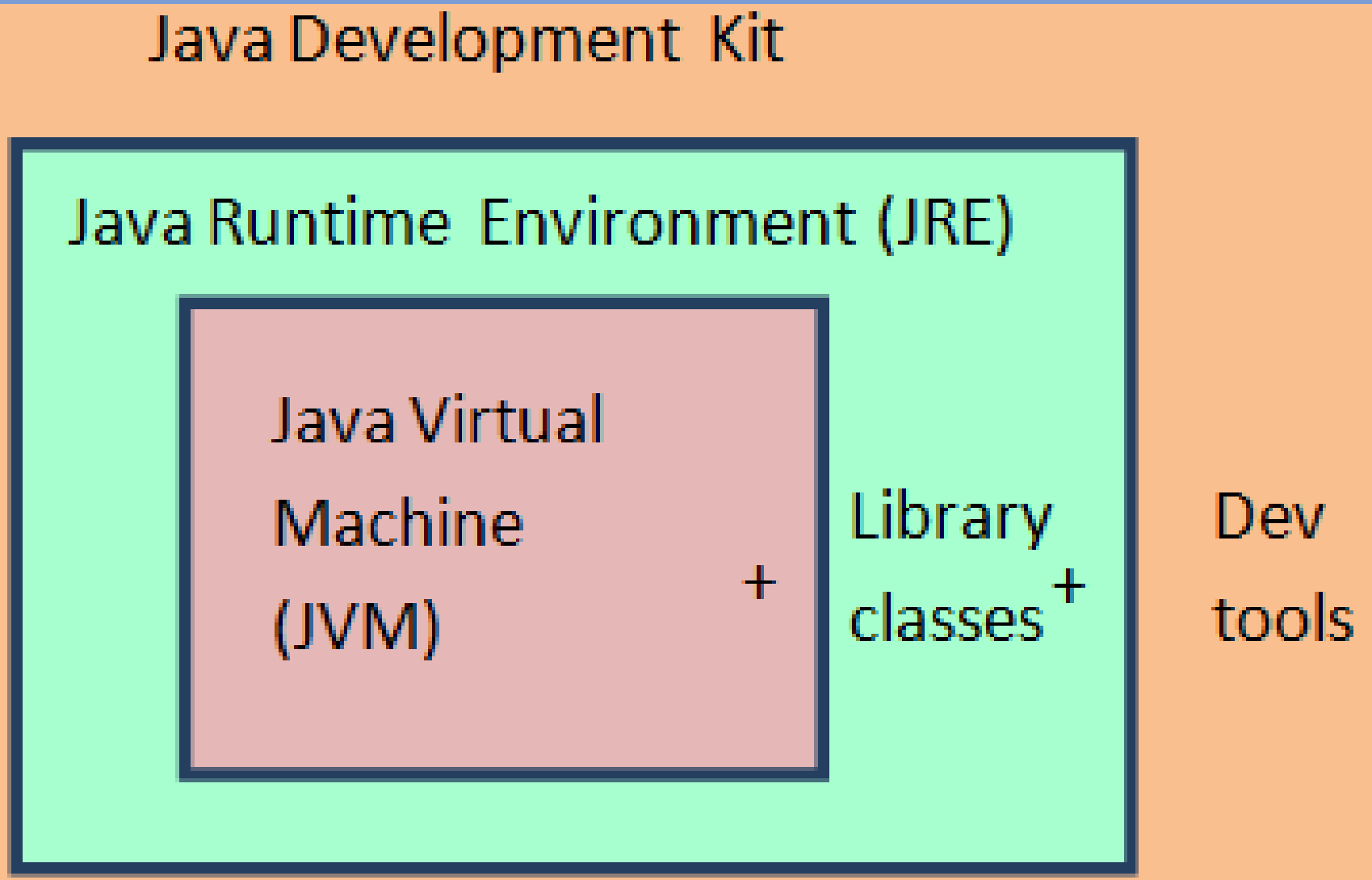
LẬP TRÌNH JAVA – CT276

Giảng viên: **Ông Thị Mỹ Linh**

NỘI DUNG

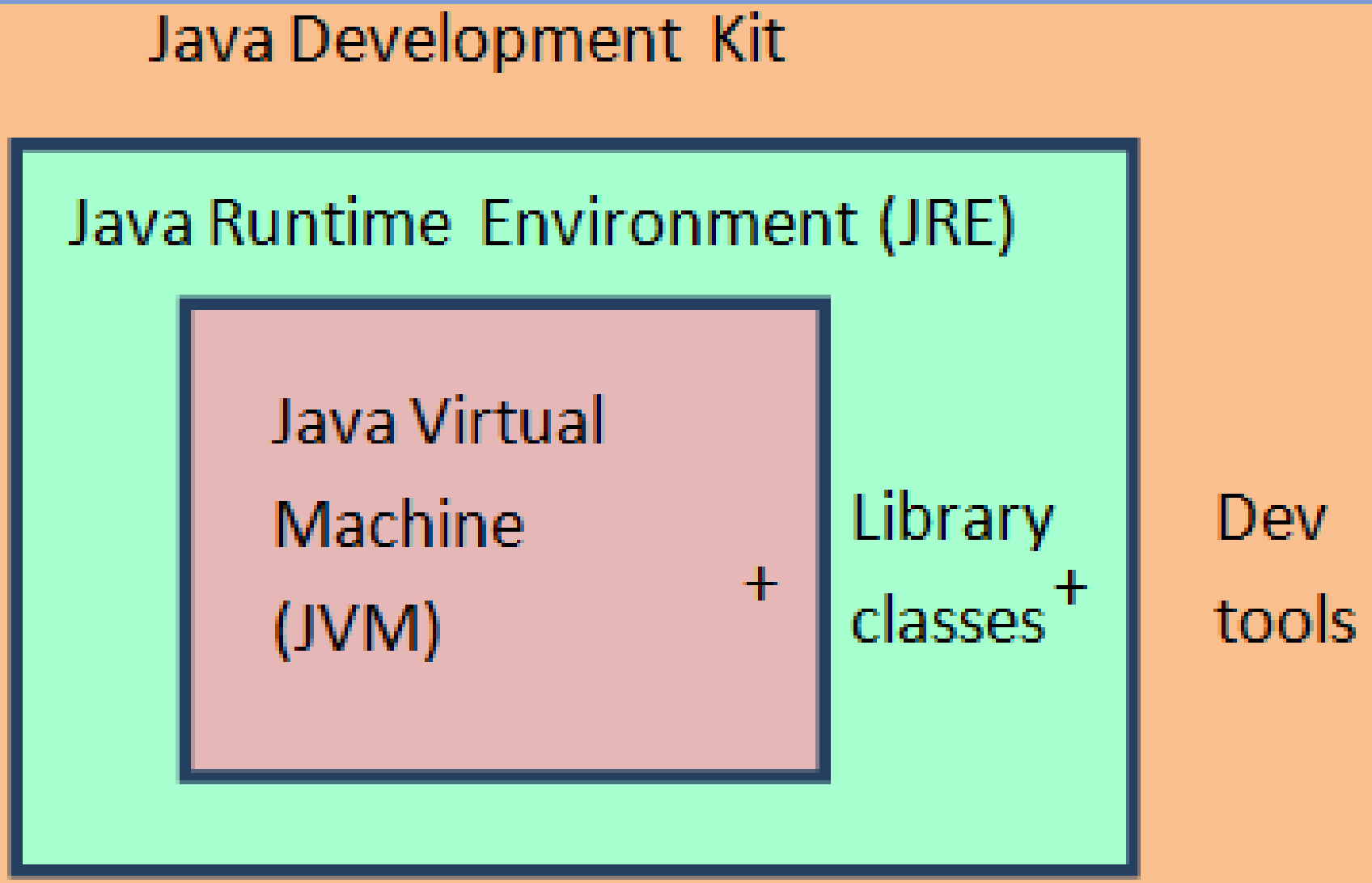
- Giới thiệu Java
- Ngôn ngữ Java
  - Kiểu dữ liệu, hằng, biến
  - Lệnh, toán tử, biểu thức
  - Cấu trúc điều khiển
  - Mảng

JAVA PLATFORM (1)



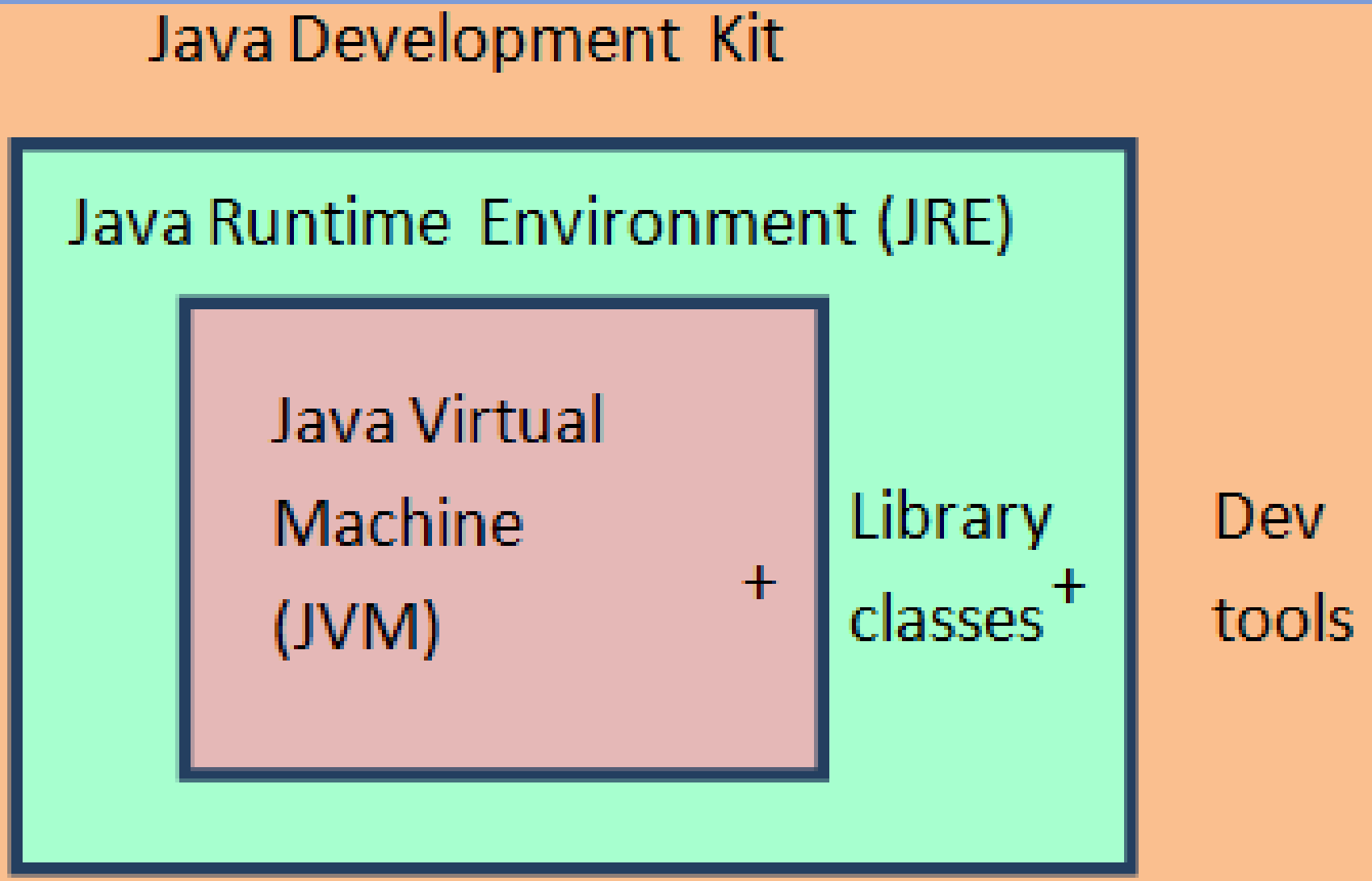
- JVM (Java Virtual Machine): Cung cấp môi trường thực thi Java code (tải + xác minh + thực thi Java code)

JAVA PLATFORM (2)



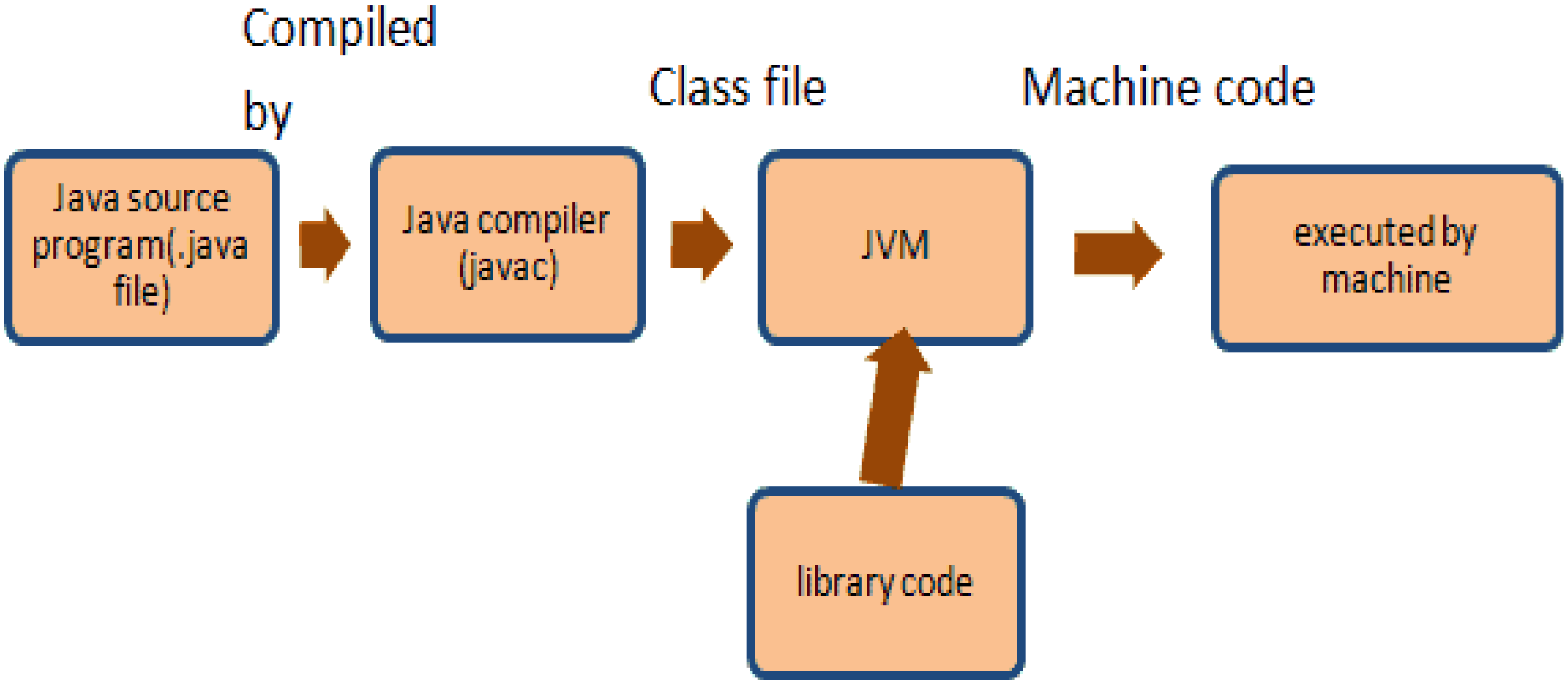
- JRE (Java Runtime Environment) = JVM + Library Class

JAVA PLATFORM (3)



- JDK (Java Development Kit) = JRE + Development Tools
- Development Tools: compiler + debugger + Java Doc

THỰC THI CHƯƠNG TRÌNH JAVA



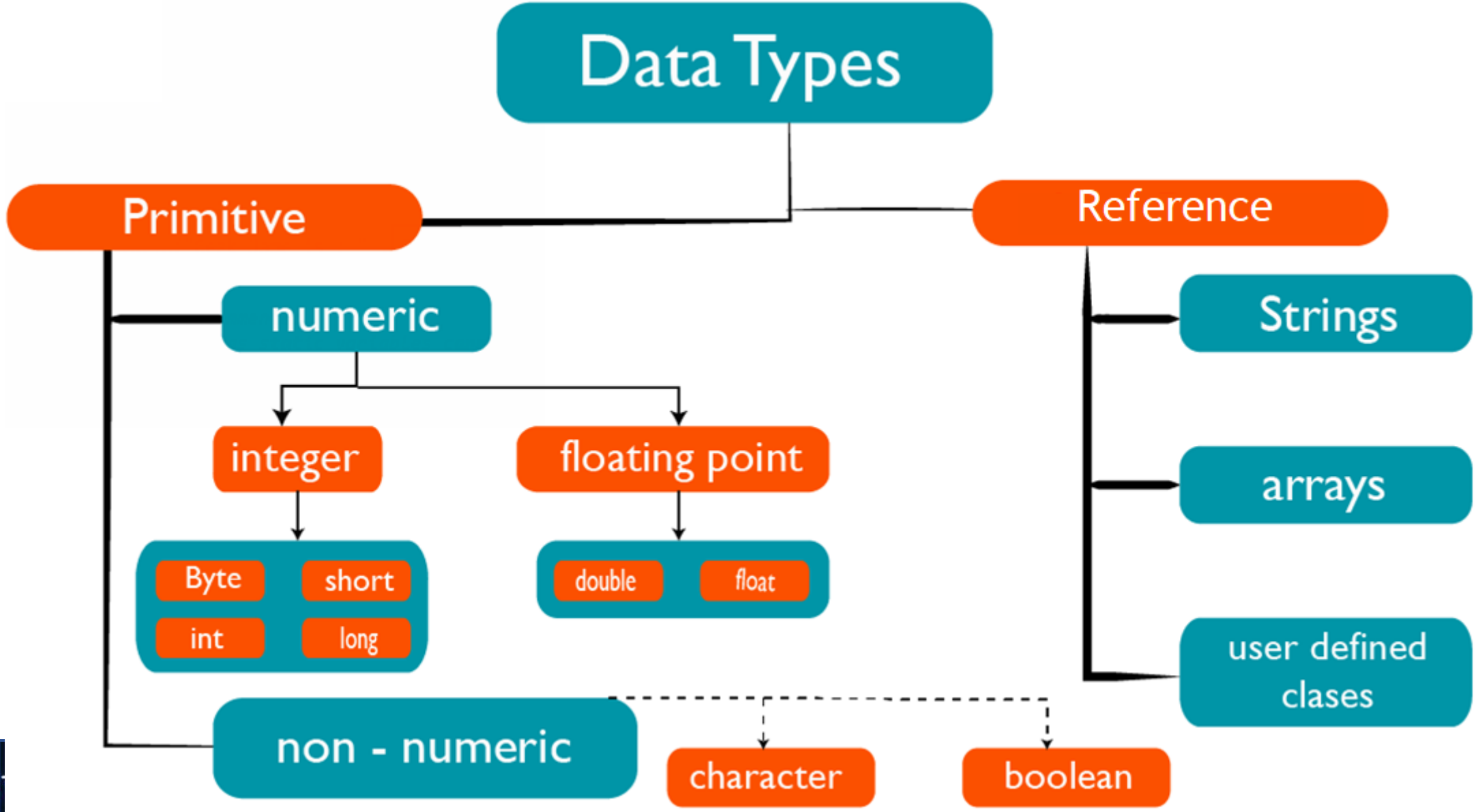
ĐẶC ĐIỂM NGÔN NGỮ JAVA

- Mã nguồn mở
- Phát triển các ứng dụng desktop, web, di động, …
- Hướng đối tượng
- Độc lập nền tảng
- Bảo mật cao
- Mạnh mẽ (kiểu, quản lý bộ nhớ, …)
- Trung lập kiến trúc
- Hỗ trợ đa luồng
- Hỗ trợ phân tán

COMMENT

- Ghi chú trên 1 dòng: //
- Ghi chú trên nhiều dòng: /* … */
- Ghi chú Javadoc: /** … */
  - Sinh file chứa nội dung ghi chú bằng câu lệnh “Javadoc”

KIỂU DỮ LIỆU


Data Types
Primitive
Reference
numeric
Strings
integer
floating point
arrays
Byte
short
int
long
double
float
user defined clases
non - numeric
character
boolean

KIỂU DỮ LIỆU

Kiểu nguyên thủy:

Type	Size
byte	1 bytes
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
boolean	1 bit
char	2 bytes

Lưu ý:

- Giá trị số thực có thể viết dạng khoa học
- Giá trị số thực mặc định là kiểu double. Ví dụ: `double a = 3.14;`
- Giá trị kiểu float: thêm hậu tố f sau giá trị. Ví dụ: `float a = 3.14f;`
- Ký tự đặt trong dấu nháy đơn. Ví dụ: `char op = '+';`
- Chuỗi đặt trong dấu nháy đôi. Ví dụ: `String s = "Viet Nam";`

HẰNG SỐ

- Hậu tố "F/f" cho kiểu `float`:
  - `float  x = 15.7F;`
- Hậu tố "L/l" cho kiểu `long`:
  - `long longNumber = 1000L;`
- Hậu tố "D/d" cho kiểu `double`:
  - `double doubleNumber = 10.7D;`
- Số ở các hệ khác:
  - `int hexNumber = 0X1c;`
  - `int binNumber =  0b1001;`

KÝ HIỆU GẠCH DƯỚI (_) TRONG HẰNG SỐ

- Dấu gạch dưới được sử dụng để phân cách các chữ số trong hằng số
  - `long total = 1_000_000L;`
- Dấu gạch dưới:
  - Không đặt ở vị trí đầu và cuối của số
  - Không đặt cạnh ký hiệu số thập phân
  - Không đặt trước hậu tố / ký tự xác định kiểu số, ví dụ F, L, b, x

BIẾN

- Biến: 1 vị trí trong bộ nhớ được sử dụng để lưu dữ liệu

```
datatype variableName;
```

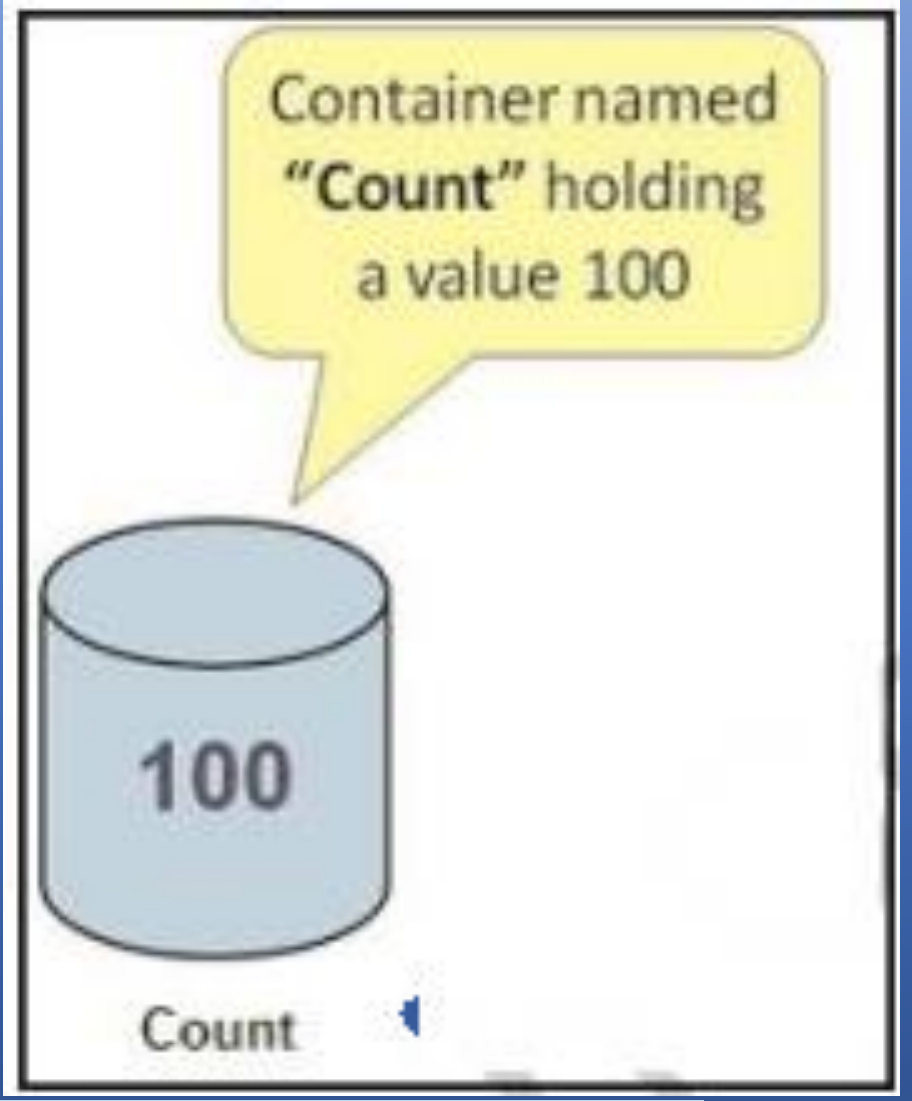
```
int rollNumber;
char gender;
```

```
int rollNumber = 101;
char gender = 'M';
```

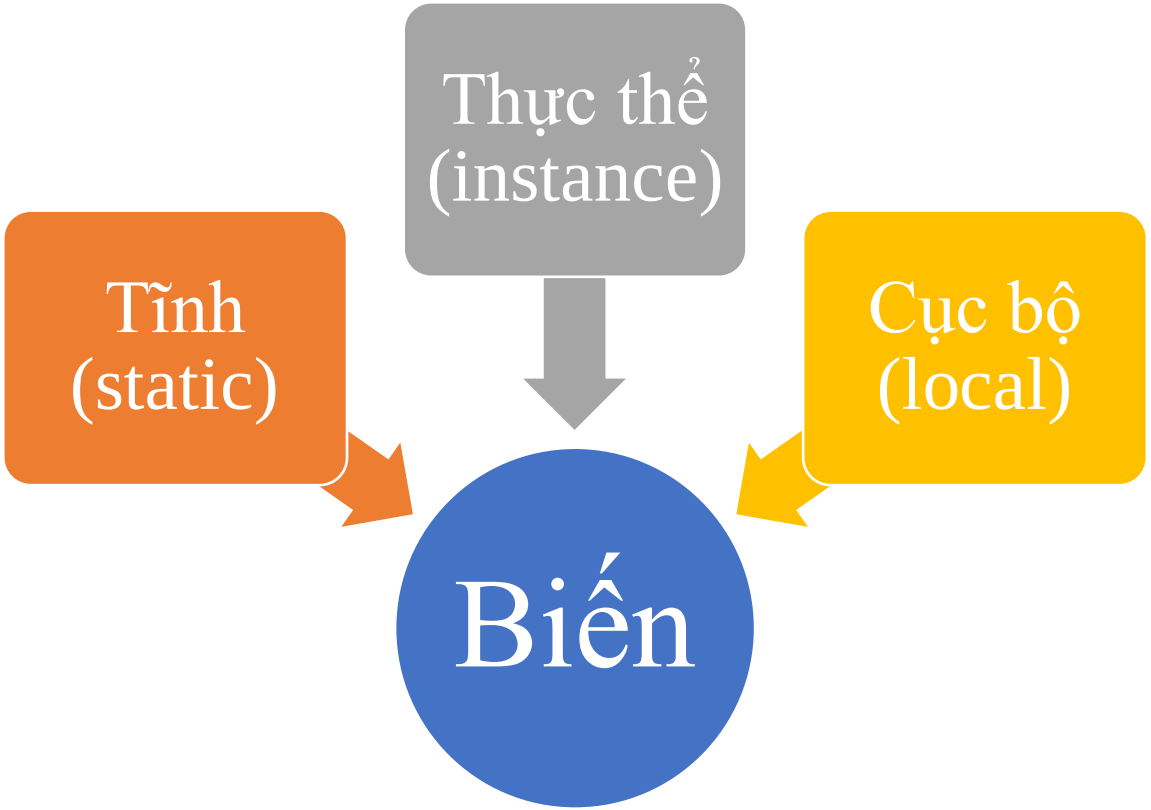
```
int x, y, z;
```

```
int a = 5, b, c = 10;
```

```
int a = 5, b, c = 10;
```



CÁC LOẠI BIẾN



BIẾN CỤC BỘ

Biến cục bộ:

- Khai báo bên trong 1 khối lệnh / phương thức / hàm
- Phạm vi của các biến này chỉ tồn tại trong khối lệnh mà các biến được khai báo
- Có giá trị NULL nếu không khởi tạo

```
class GFG {
    public static void main(String[] args)
    {
        // Declared a Local Variable
        int var = 10;

        // This variable is local to this main method only
        System.out.println("Local Variable: " + var);
    }
}
```

BIẾN THỰC THỂ (1)

Biến thực thể:

- Khai báo bên trong lớp nhưng bên ngoài tất cả các khối lệnh / phương thức / hàm
- Được tạo khi một đối tượng của lớp được tạo và bị hủy khi đối tượng đó bị hủy
- Được truy xuất cùng với đối tượng: "`Tên_đối_tượng.Tên_biến`"
- Có giá trị mặc định (tùy kiểu dữ liệu) nếu không được khởi tạo

BIẾN THỰC THỂ (2)

Biến thực thể:

```
class GFG {

    // Declared Instance Variable
    public String geek;
    public int i;
    public Integer I;
    public GFG()
    {
        // Default Constructor
        // initializing Instance Variable
        this.geek = "Shubham Jain";
    }

    // Main Method
    public static void main(String[] args)
    {
        // Object Creation
        GFG name = new GFG();

        // Displaying O/P
        System.out.println("Geek name is: " + name.geek);
        System.out.println("Default value for int is "
                           + name.i);

        // toString() called internally
        System.out.println("Default value for Integer is "
                           + name.I);
    }
}
```

BIẾN TĨNH (1)

Biến tĩnh:

- Còn được gọi là biến lớp
- Được khai báo như biến thực thể nhưng có thêm từ khóa "`static`"
- Có giá trị mặc định (tùy kiểu dữ liệu) nếu không được khởi tạo
- Được khởi tạo khi chương trình bắt đầu và được hủy khi chương trình kết thúc (không phụ thuộc vào đối tượng của lớp)
- Được truy xuất cùng với đối tượng: "`Tên_lớp.Tên_biến`"

BIẾN TĨNH (2)

Biến tĩnh:

```
class GFG {
    // Declared static variable
    public static String geek = "Shubham Jain";

    public static void main(String[] args)
    {

        // geek variable can be accessed without object
        // creation Displaying O/P GFG.geek --> using the
        // static variable
        System.out.println("Geek Name is : " + GFG.geek);

        // static int c=0;
        // above line,when uncommented,
        // will throw an error as static variables cannot be
        // declared locally.
    }
}
```

CÂU HỎI

- Cách truy xuất biến a, b?
- Sự khác nhau giữa a và b?

```
class GFG
{
    // Static variable
    static int a;

    // Instance variable
    int b;
}
```

HẰNG (CONSTANT)

- Giá trị của hằng:
  - bắt buộc được gán khi khởi tạo
  - không thay đổi sau khi khởi tạo
- Khai báo:

```
final float PI = 3.14;
```

ENUMERATION

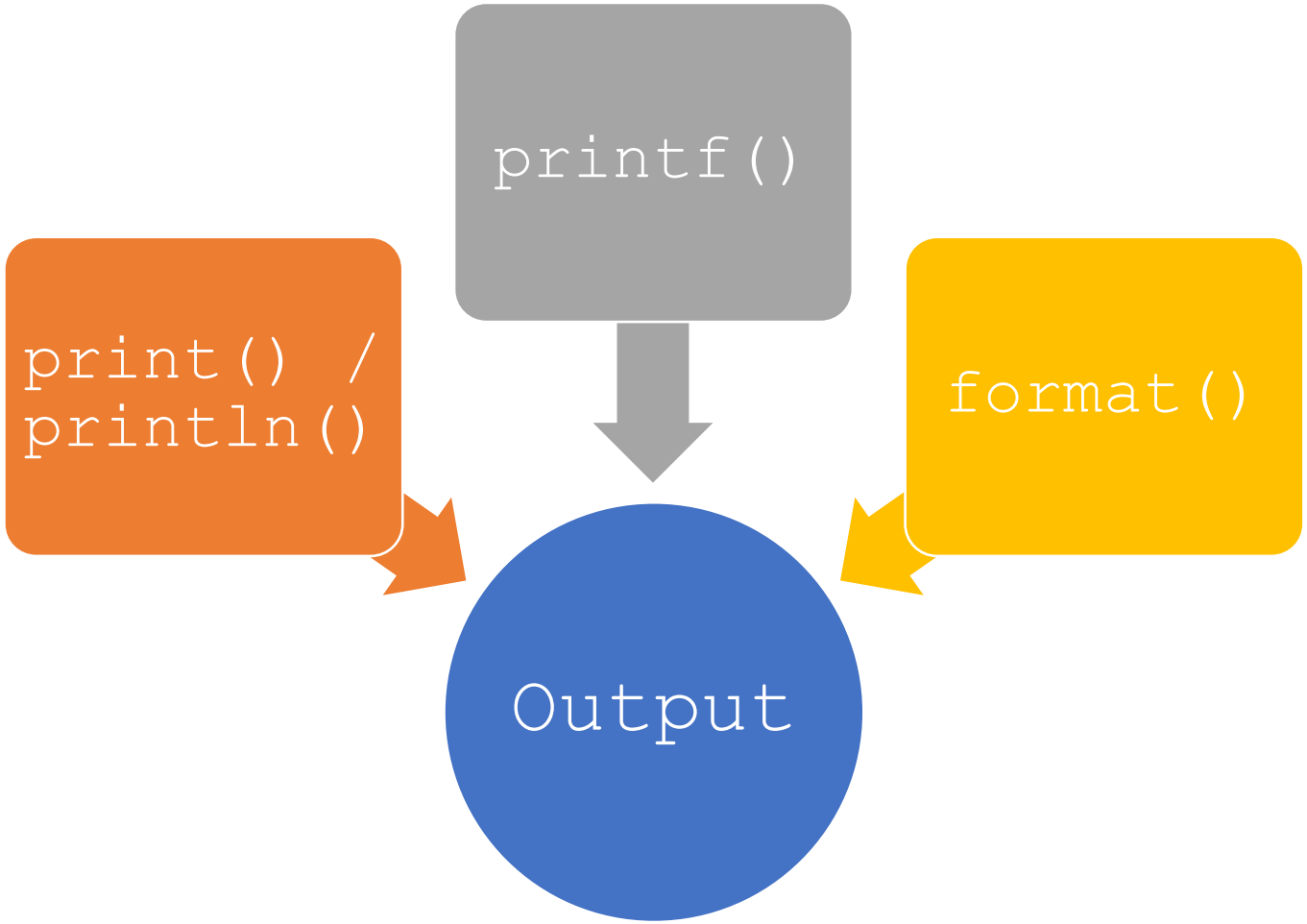
- Là một danh sách các hằng
- Thuộc kiểu lớp (`class`)
  - Chứa biến thể hiện, phương thức và hàm xây dựng
- Được khai báo bằng từ khóa "`enum`"

```
enum tên_enum {hằng1, hằng2, …, hằngN}
```

- Gọi giá trị trong enumeration:

```
tên_enum.hằngN
```

XUẤT DỮ LIỆU (OUTPUT)

- 

OUTPUT

```
int a = 10, b = 20;

System.out.print("String " + a); // In (không xuống dòng)

System.out.println("String " + a); // In và xuống dòng

System.out.printf("String %d %d", a, b); // In với chuỗi định dạng

int x = 10;

float y = 2.5f;

System.out.printf("x = %d %n", x);

System.out.printf("y = %f %n", y );

System.out.printf("PI = %5.3f, E = %5.4f %n", Math.PI, Math.E);
```

```
String 10String 10
String 10 20
```

```
x = 10
y = 2.500000
PI = 3.142, E = 2.7183
```

OUTPUT

```
int x = 10;

float y = 2.5f;

System.out.format("x = %d %ny = %f", x, y);
```

```
x = 10
y = 2.500000
```

```
System.out.format("%2$,3.2f %1$s", "meters", 1260.5052);

System.out.format("%,3.2f %s", 1260.5052, "meters");
```

```
1,260.51 meters
```

OUTPUT

- Chuỗi định dạng:

```
%[arg$][flags][width][.precision]conversion
```

- `flags`: số thứ tự đối số

flags	Mô tả
-	Canh trái
+	Thêm dấu + /-
0	Thêm 0 vào vị trí trống
,	Ký tự phân cách
(	Đặt () vào số âm

- `arg`: số thứ tự đối số
- `width`: số ký tự hiển thị
- `precision`: số chữ số thập phân
- `conversion`: mã định dạng

NHẬP (INPUT)

- Nhập dữ liệu từ bàn phím:

```
import java.util.Scanner;

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();    // Đọc số nguyên

double b = sc.nextDouble();    // Đọc số thực

String s = sc.nextLine();    // Đọc chuỗi thì đọc cả dòng
```

```
nextByte()
nextInt()
nextLong()
nextFloat()
nextDouble()
```

- Lưu ý: + Sử dụng `nextLine()` để xóa ký tự `'\n'` trong bộ đệm sau khi nhập số.

+ Nên sử dụng 1 object của Scanner trong toàn bộ chương trình

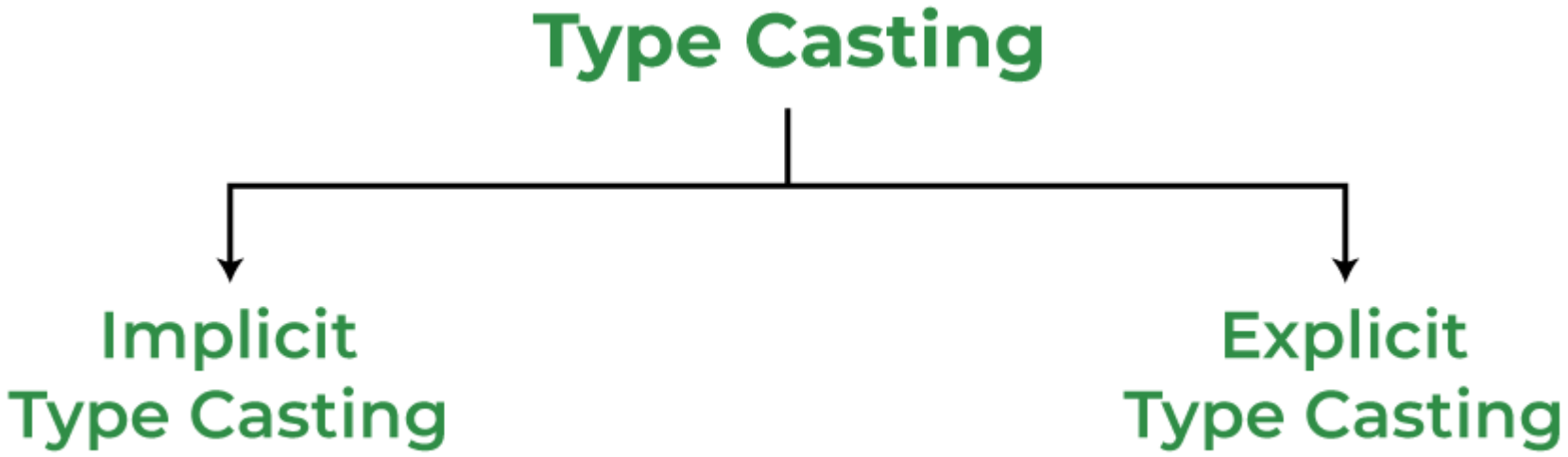
TOÁN TỬ

- Toán học: + , - , * , / , %
- Gán: = , += , -=, *= , /=
- Quan hệ: ==, != , > , >= , < , <=
- Luận lý: &&, ||, !
- Điều kiện: <Điều_kiện> ? Giá_Trị_1: Giá_trị_2

```
String s = (a % 2 == 0) ? "Số chẵn" : "Số lẻ";

String s = (a > 0) ? "Số dương" : (a < 0) ? "Số âm" : "Số không";
```

ÉP KIỂU (1)

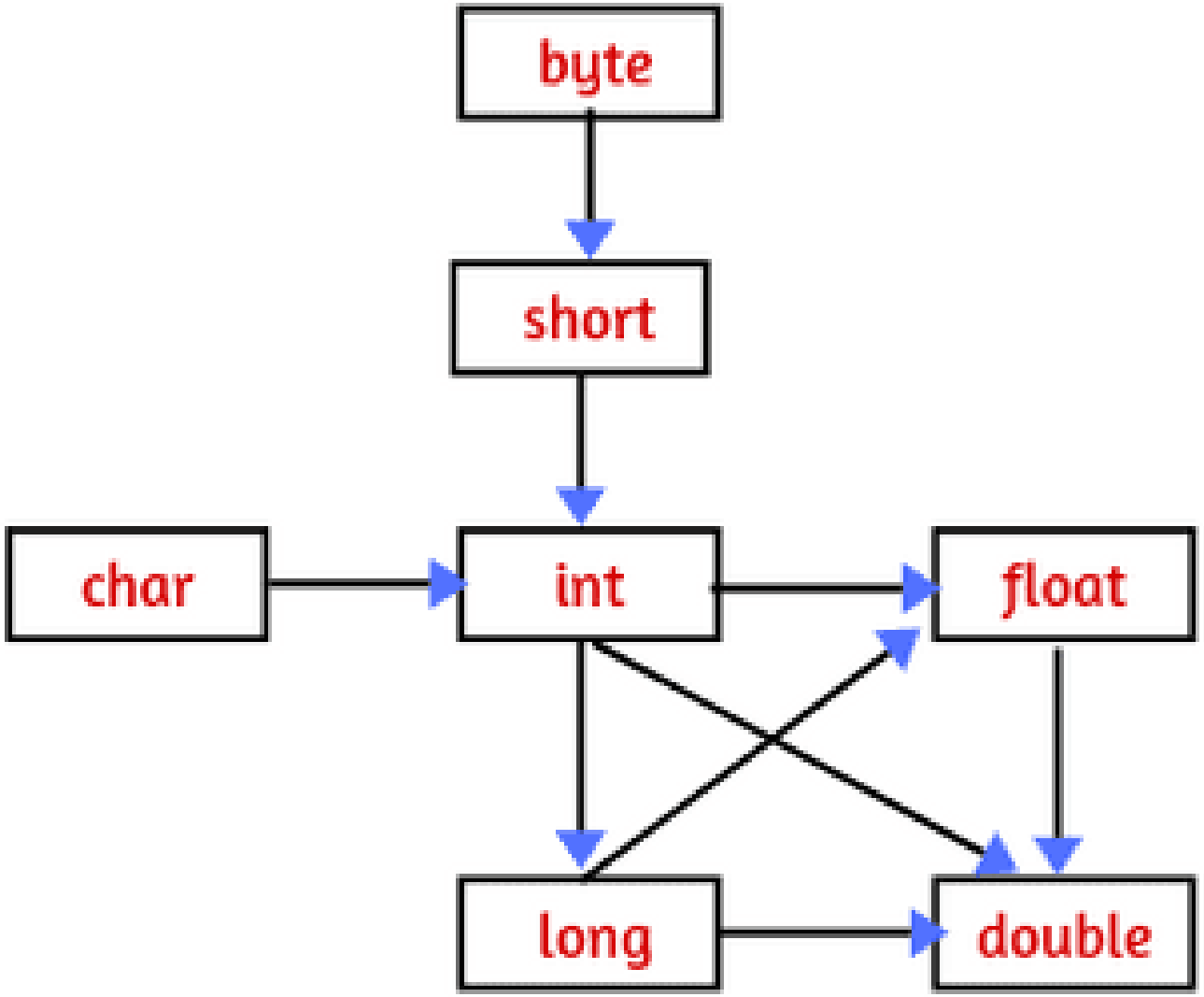


ÉP KIỂU (2)

- Implicit Type Casting

```
int a = 10;

long b = a;
```



ÉP KIỂU (2)

- Explicit Type Casting

```
(Kiểu) Giá_Trị;
```

```
double x = 12345.67890;
float y = (float) x;
```

CÂU LỆNH ĐIỀU KIỆN

```
if (condition)
{
      ……
}
else
{
      ……
}
```

```
switch (expression)
 {
      case v1:
            // Lệnh 1
            break;
      case v2:
            // Lệnh 2
            break;
      case v3:
      case v4:
      case v5:
            // Lệnh 3
            break;
      default:
            ...
            break;
}
```

Chú ý: switch chỉ sử dụng khi so sánh các kiểu byte, short, int, char, String

VÒNG LẶP

- `for (init; condition; increment){…}`
- `while (condition) {…}`
  - Kiem tra dieu kien truoc khi thuc hien
  - Khong biet so lan lap
- `do`

`{…}`

`while (condition);`

- `int[] arr = { 1, 2, 3, 4, 5};`

`for (int x: arr) {…}`

VÒNG LẶP

- `for (init; condition; increment){…}`
  - Kiem tra dieu kien truoc khi thuc hien
  - Biet duoc so lan lap
- `while (condition) {…}`
  - Kiem tra dieu kien truoc khi thuc hien
  - Khong biet so lan lap
- `do`

`{…}`

`while (condition);`

  - Kiem tra dieu kien SAU khi thuc hien
  - Khong biet so lan lap

CHÚ Ý:

- `break` dừng toàn bộ vòng lặp đang chạy
- `continue` dừng lần lặp hiện tại, và đi tới lần lặp tiếp theo

MẢNG

- Lưu một tập hợp các giá trị cùng kiểu dữ liệu
- Mỗi giá trị trong mảng được xem như là 1 phần tử trong mảng
- Mỗi phần tử trong mảng được truy xuất bằng chỉ số (index)
  - Chỉ số: là số nguyên, bắt đầu bằng 0
- Các loại mảng:
  - Mảng 1 chiều: có 1 chỉ số
  - Mảng n chiều: có n chỉ số

MẢNG 1 CHIỀU

- Cú pháp:
  - `DataType[]  ArrayName;`
  - `DataType[]  ArrayName =  new DataType[NumberOfItems] ;`
- Khởi tạo mảng:
  - `DataType[]  ArrayName = {value1, value2, …};`
- Truy xuất mảng:

```
for (int i= 0; i< arr.length; i++)
{Phần tử: arr[i]}

for (int x: arr)
{Phần tử: x}
```

MẢNG 1 CHIỀU

```
Scanner sc= new Scanner(System.in);

int[] arr = new int[5];
for(int i=0; i<5; i++){
    System.out.format("Nhap phan tu thu %d: ", i+1);
    arr[i] = sc.nextInt();
}

System.out.print("MANG SAU KHI NHAP (1): ");
for(int i=0; i<arr.length; i++){
    System.out.print("   " + arr[i]);
}

System.out.print("\nMANG SAU KHI NHAP (2): ");
for(int x: arr){
    System.out.print("   " + x);
}
```

```
Nhap phan tu thu 1: 10
Nhap phan tu thu 2: 20
Nhap phan tu thu 3: 30
Nhap phan tu thu 4: 40
Nhap phan tu thu 5: 50
MANG SAU KHI NHAP (1):    10   20   30   40   50
MANG SAU KHI NHAP (2):    10   20   30   40   50
```

MẢNG NHIỀU CHIỀU

- Cú pháp:
  - `DataType[][] ArrayName;`
  - `DataType[][] ArrayName = new DataType[NumberOfRows][NumberOfColumns] ;`
- Khởi tạo mảng:
  - `DataType[][] ArrayName = {{value1, value2, …},`
    `{value3, value4, …},`
    `{…….},`
    `…….`
    `};`

MẢNG NHIỀU CHIỀU

- Truy xuất mảng:

```
for (int i= 0; i< arr.length; i++)
    for (int j= 0; j< arr[i].length; j++)
    {Phần tử: arr[i][j]}

for (int[] row: arr)
   for (int x: row)
   {Phần tử: x}
```

MẢNG NHIỀU CHIỀU

```
int x = 10;
int[][] matrix = new int[3][4];
for(int i=0; i<3; i++)
    for(int j=0; j<4; j++)
        matrix[i][j] = x++;

System.out.print("\nIN MANG (1): ");
for(int i=0; i<matrix.length; i++){
    System.out.println();
    for(int j=0; j<matrix[i].length; j++){
        System.out.print("   " + matrix[i][j]);
    }
}

System.out.print("\nIN MANG (2): ");
for (int[] row : matrix) {
    System.out.println();
    for (int element : row) {
        System.out.print("   " + element);
    }
}
```

```
IN MANG (1):
   10   11   12   13
   14   15   16   17
   18   19   20   21
IN MANG (2):
   10   11   12   13
   14   15   16   17
   18   19   20   21
```

THỰC HÀNH (1)

Viết chương trình:

1. Tính chu vi và diện tích hình chữ nhật. Kiểm tra dữ liệu đầu vào hợp lệ.

2. Tính điểm trung bình 3 môn toán, văn, anh và xếp loại

- Điểm trung bình >=9: Xuất sắc
- 8 <= Điểm trung bình <9 : Giỏi
- 7 <= Điểm trung bình <8 : Khá
- 4 <= Điểm trung bình <7 : Trung bình
- Điểm trung bình < : Kém

THỰC HÀNH (2)

Viết chương trình:

3. Tính tổng các số nguyên từ 1 đến n. Với n là số nguyên dương lớn hơn 5 và nhỏ hơn 1000
4. Tính giai thừa một số nguyên n. Với n là số nguyên dương lớn hơn 1 và nhỏ hơn 50.
5. Kiểm tra một số nguyên dương n:
    - Có bao nhiêu chữ số
    - Tính tổng số chữ số

THỰC HÀNH (3)

6. Viết chương trình nhập 1 dãy n số nguyên:

a. Hiển thị các số nguyên vừa nhập

b. Hiển thị các số lẻ có trong dãy số

c. Tính tổng các số nguyên có trong dãy số

d. Tính tổng các số chia hết cho 5 có trong dãy số

e. Tìm số nguyên lớn nhất/nhỏ nhất có trong dãy số

f. Sắp xếp dãy số tăng dần

THỰC HÀNH (4)

7. Viết chương trình nhập 1 ma trận 2 chiều n dòng và m cột:

   a. Hiển thị ma trận vừa nhập

   b. Tìm số nguyên lớn nhất/nhỏ nhất trong ma trận

CÂU HỎI