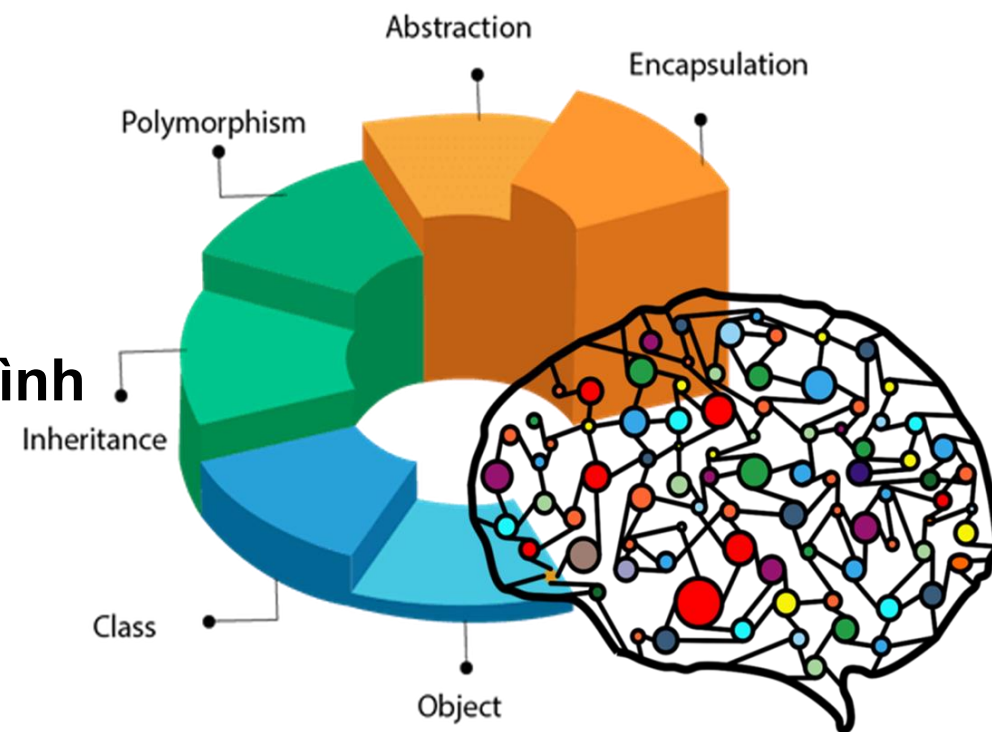


# KỸ THUẬT LẬP TRÌNH

---

ĐẶNG VĂN NGHĨA  
0975079414  
nghiadv@donga.edu.vn

1. Tập ký tự dùng trong ngôn ngữ Java
2. Từ khóa (keywords)
3. Tên (name)
4. Một số quy tắc cần nhớ khi viết chương trình
5. Khai báo và toán tử gán
6. Đưa kết quả lên màn hình
7. Nhập số liệu từ bàn phím
8. Ví dụ chương trình đơn giản
9. Vận hành chương trình trên máy tính



# 1. TẬP KÝ TỰ DÙNG TRONG NGÔN NGỮ JAVA



Mọi ngôn ngữ lập trình đều được xây dựng từ một bộ ký tự nhất định. Các ký tự được nhóm lại theo nhiều cách khác nhau tạo nên các từ. Các từ được liên kết theo một qui tắc nào đó để tạo thành các câu lệnh. Một chương trình bao gồm nhiều câu lệnh và diễn đạt một thuật toán để giải một bài toán cụ thể.

❖ **Ngôn ngữ Java được xây dựng dựa trên bộ ký tự sau:**

- 26 chữ cái hoa: A B C D ... Z
- 26 chữ cái thường: a b c d ... z
- 10 chữ số: 0 1 2 ... 9
- Các ký hiệu toán học: + - \* / = ( )
- Ký tự gạch nối: \_
- Các ký tự khác: . , : ; [ ] { } ! \ & % # \$ ...
- Dấu cách (space) khoảng trống dùng để tách từ. Ví dụ: “Xin chào” có 8 ký, “Xinchào” có 7 ký tự

**Lưu ý:** Khi viết chương trình không được sử dụng bất kỳ ký hiệu nào khác ngoài tập các ký tự nói trên.

## ❖ Ký tự đặc biệt trong Java:

STT	Ký tự	Ý nghĩa
1	\b	Backspace
2	\t	Ký tự Tab
3	\n	Ký tự xuống dòng (LF)
4	\r	Ký tự xuống dòng (CR)
5	\f	Tách trang
6	\'	Dấu nháy đơn (')
7	\"	Dấu nháy kép (")
8	\\	Dấu gạch chéo ngược
9	\ooo	ASCII ký tự hệ cơ số 8
10	\uhhhh	ASCII ký tự hệ cơ số 16

## ❖ Ký tự đặc biệt trong Java: thường hay sử dụng nhiều nhất \' \'\" \n \t

### ❖ Ví dụ:

- `System.out.println("Xin chào\nCác bạn");`
  - ✓ Xin chào
  - ✓ Các bạn
- `System.out.println("STT\tKý tự\tÝ nghĩa");`
  - ✓ STT      Ký tự      Ý nghĩa
- `System.out.println("Tèo "rất nhớ" tụt");`
  - ✓ Lỗi
- `System.out.println("Tèo \"rất nhớ\" tụt");`
  - ✓ Tèo "rất nhớ" tụt

## 2. TỪ KHÓA (KEYWORDS)



**Từ khóa** là những từ có ý nghĩa hoàn toàn xác định. Chúng thường được sử dụng để khai báo các kiểu dữ liệu, viết các toán tử và các câu lệnh.

## 2. TỪ KHÓA (KEYWORDS)

❖ **Từ khóa sử dụng trong Java:** có 51 từ khóa liệt kê từ phiên bản JDK 17

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while
_ (underscore)				

❖ **Lưu ý:**

- Không được dùng từ khóa để đặt tên cho các hằng, biến, mảng, hàm, ...
- Từ khóa phải được viết bằng chữ thường. Chẳng hạn không được viết INT mà phải viết int.

## 2. TỪ KHÓA (KEYWORDS)

❖ **Ví dụ:** sử dụng tên biến trùng với từ khóa **switch**

```
public class Example {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int switch;  
    }  
}
```

Exception in thread "main" java.lang.Error:  
Unresolved compilation problem:  
Syntax error on token "switch", invalid  
VariableDeclarator

at Example.main([Example.java:7](#))



## 2. TỪ KHÓA (KEYWORDS)

STT	Từ khóa	Ý nghĩa
1	assert	Kiểm tra điều kiện đúng hay sai (hay dùng trong Unit Test)
2	boolean	Khai báo biến kiểu logic với hai giá trị
3	break	Lệnh switch-case hoặc dùng để thoát khỏi vòng lặp
4	byte	Các giá trị nguyên chiếm 8 bit (1byte)
5	case	Trường hợp được chọn theo Switch (chỉ dùng khi được đi kèm Switch)
6	catch	Dùng để bắt ngoại lệ, dùng kèm với try để xử lý những ngoại lệ phát sinh trong chương trình
7	char	Là kiểu ký tự Unicode, mỗi ký tự có 16 bit (2 byte)
8	class	Dùng để định nghĩa class
9	const	Không thể dùng trong java bởi nó chưa được sử dụng
10	continue	Được dùng để dừng chu trình (iteration) lặp hiện tại và bắt đầu chu trình kế tiếp

## 2. TỪ KHÓA (KEYWORDS)

STT	Từ khóa	Ý nghĩa
11	default	Mặc định được thực thi nếu không có case nào trả về giá trị True – được dùng trong Switch case.
12	do	Dùng ở vòng lặp While
13	double	Là kiểu số thực có các giá trị được biểu diễn bởi dấu phẩy động 64 bit (8byte)
14	else	Rẽ nhánh điều kiện ngược với If
15	enum	Kiểu dữ liệu Enum – tương đối giống với kiểu dữ liệu mảng. Khác biệt ở chỗ các phần tử của kiểu này có thể bổ sung thêm các phương thức.
16	extends	Dùng để định nghĩa lớp con kế thừa những thuộc tính và phương thức từ lớp cha.
17	final	Dùng để chỉ ra các biến – phương thức không thay đổi sau khi đã được định nghĩa. Những phương thức final không được kế thừa và override.
18	finally	Thực hiện một khối lệnh đến cùng, bỏ qua các ngoại lệ – dùng trong Try-catch.
19	float	Kiểu số thực – Các giá trị được biểu diễn bởi dạng dấu phẩy động 32 bit.
20	for	Dùng trong vòng lặp for – Các bước lặp đã xác định từ trước.

## 2. TỪ KHÓA (KEYWORDS)

STT	Từ khóa	Ý nghĩa
21	goto	Chưa được sử dụng
22	if	Là lệnh chọn theo điều kiện logic
23	implements	Xây dựng 1 lớp mới cài đặt những phương thức từ interface xác định trước
24	import	Dùng để yêu cầu 1 hay 1 vài lớp ở các gói chỉ định cần nhập vào để sử dụng trong ứng dụng hiện thời
25	long	Là kiểu số nguyên lớn – Các giá trị chiếm 64 bit (8 byte)
26	native	Sử dụng khi lập trình viên muốn dùng code bằng ngôn ngữ khác
27	new	Khởi tạo đối tượng
28	package	Sử dụng khi xác định 1 gói sẽ chứa một số lớp trong file mã nguồn
29	private	Khai báo biến dữ liệu, phương thức riêng trong từng lớp và chỉ cho phép truy cập trong lớp đó.
30	protected	Dùng để khai báo biến dữ liệu – Chỉ được truy cập ở lớp cha và những lớp con của lớp đó.

## 2. TỪ KHÓA (KEYWORDS)

STT	Từ khóa	Ý nghĩa
31	public	Dùng để khai báo biến dữ liệu, lớp – Phương thức công khai có thể tự truy cập ở mọi hệ thống.
32	return	Kết thúc phương thức, trả về giá trị cho phương thức.
33	short	Kiểu số nguyên ngắn – giá trị chiếm 16 bit (2byte)
34	static	Định nghĩa biến, phương thức của một lớp có thể được truy cập trực tiếp từ lớp mà không thông qua khởi tạo đối tượng của lớp.
35	super	Biến chỉ tới đối tượng ở lớp cha
36	switch	Sử dụng trong câu lệnh điều khiển Switch case
37	synchronized	Chỉ ra là ở mỗi thời điểm chỉ có 1 đối tượng hay 1 lớp có thể truy nhập đến biến dữ liệu hoặc phương thức loại đó – Thường được sử dụng trong lập trình đa luồng (multithreading).
38	this	Biến chỉ tới đối tượng hiện thời
39	throw	Tạo một đối tượng Exception nhằm chỉ định 1 trường hợp ngoại lệ xảy ra
40	throw	Chỉ định cho qua ngoại lệ nếu exception xảy ra

## 2. TỪ KHÓA (KEYWORDS)

STT	Từ khóa	Ý nghĩa
41	transient	Chỉ định rằng nếu một đối tượng được Serialized, giá trị của biến sẽ không cần được lưu trữ
42	try	Thử thực hiện cho đến khi xảy ra 1 ngoại lệ
43	void	chỉ định 1 phương thức không trả về giá trị
44	volatile	Báo cho chương trình dịch biết là biến khai báo volatile có thể thay đổi tùy ý trong các luồng (thread)
45	While	Sử dụng trong lệnh điều khiển while

### 3. TÊN (NAME)



**Tên** là một khái niệm rất quan trọng, nó dùng để xác định các đối tượng khác nhau trong chương trình. Trong chương trình có tên biến, tên hàm, tên mảng, tên tệp, tên lớp, tên đối tượng, tên hằng ...

## 3. TÊN (NAME)

### ❖ Quy ước đặt tên chung trong java:

- JAVA phân biệt chữ hoa chữ thường.
- Các tệp, lớp, phương thức, biến, ... phải được đặt tên theo ý nghĩa và mục đích.
- Tránh các tên chỉ có một hoặc quá nhiều ký tự.
- Không sử dụng các tên tương tự nhau trong cùng một lớp.  
Ví dụ: persistentObject và persistentObjects
- Không sử dụng tên khó hiểu và gây hiểu nhầm.
- Không sử dụng tên đa ngôn ngữ (tiếng Anh + tiếng Nhật, ...). Ví dụ: addtinhtoan
- Không sử dụng các từ khóa (keyword) của JAVA để đặt tên.
- Không đặt tên bắt đầu bằng số. Ví dụ: 123student
- Không đặt tên bao gồm dấu cách và ký hiệu toán học.

### ❖ Quy ước đặt tên biến trong Java:

- Đặt tên theo quy tắc camelCase: Chữ cái đầu tiên của tên biến viết thường và chữ cái đầu tiên của các từ tiếp theo viết hoa. Ví dụ: studentNumber.
- Kết hợp tên kiểu dữ liệu vào tên biến. Ví dụ: studentList cho kiểu List, studentArray cho kiểu Array, ...

### ❖ Quy ước đặt tên hằng số trong Java:

- Tuân theo các quy tắc đặt tên chung.
- Viết hoa tất cả các chữ cái và dùng gạch dưới \_ để nối nếu nhiều từ. Ví dụ: MAX\_PRIORITY



## 3. TÊN (NAME)

### ❖ Quy ước đặt tên phương thức (method) trong Java:

- Tuân theo các quy tắc chung.
- Đặt tên theo quy tắc CamelCase, giống như tên biến. Ví dụ: countStudent

### ❖ Quy ước đặt tên lớp (class) và giao tiếp (interface) trong Java:

- Tuân theo các quy tắc chung và ký tự đầu tiên của mỗi từ trong tên lớp phải viết hoa. Ví dụ: CountStudent
- Từ cuối cùng trong tên lớp được sử dụng để mô tả đặc điểm và kiểu của lớp. Ví dụ: DivZeroException (Lớp ngoại lệ trong xử lý try-catch)
- Chữ cái đầu tiên của giao tiếp (interface) phải viết hoa. Ví dụ: IFrame
- Từ đầu tiên bắt đầu một lớp trừu tượng (Abstract class) phải là Abstract. Ví dụ: AbstractCountStudent

## 3. TÊN (NAME)

### ❖ Quy ước đặt tên gói (Package) trong Java:

- Tuân theo các quy tắc chung.
- Tất cả viết chữ thường.

### ❖ Quy ước đặt tên dự án (Project) trong Java:

- Tuân theo các quy tắc chung.
- Ký tự đầu của mỗi từ viết hoa. Ví dụ SinhVienProject

## 3. TÊN (NAME)

### ❖ Ưu điểm đặt tên theo quy ước:

- Người lập trình và các lập trình viên khác có thể đọc mã nguồn dễ dàng.
- Mất ít thời gian hơn để tìm ra mã hoạt động.

### ❖ Nhược điểm không đặt tên theo quy ước:

- Tạo ra sự nhầm lẫn hoặc lỗi mã nguồn.

## ❖ Quy tắc viết chương trình Java: (^.-.^)

- Phải viết tên lớp (class) và phương thức có trong chương trình. Các lớp hoặc phương thức phải được đặt tên. Ví dụ: tên lớp **Example**, tên phương thức **main**.
- Tên lớp phải giống với tên **file .java** lưu mã nguồn (theo quy ước đặt tên trong Java). Ví dụ: tên lớp **Example** thì tên file lưu chương trình là **Example.java**.
- Bắt đầu và kết thúc các lớp hoặc phương thức hoặc khối lệnh bằng **cặp dấu ngoặc nhọn { }**
- Kết thúc câu lệnh bằng **dấu chấm phẩy;**
- Sử dụng thụt lề cũng như xuống dòng hợp lý làm cho cấu trúc chương trình rõ ràng, dễ đọc.
- Chú thích (Comment) cho mã nguồn giúp người đọc dễ hiểu (sử dụng `//`: lệnh đơn; `/*` và `*/`: khối lệnh).

```
public class Example {  
    public static void main(String[] args) {  
        //Các câu lệnh xử lý  
    }  
}
```

## 5. KHAI BÁO VÀ TOÁN TỬ GÁN



**Mọi biến** (variable) trước khi sử dụng đều phải khai báo để xác định kiểu của nó.



## ❖ Ví dụ khai báo:

- Khai báo biến nguyên (kiểu int) dùng từ khóa int.  
✓ **int a1, a2, a3 ;** /\*Khai báo các biến a1, a2, a3 kiểu int\*/
- Khai báo biến thực (kiểu float) dùng từ khóa float.  
✓ **float b1, b2, b3 ;** /\*Khai báo các biến b1, b2, b3 kiểu float\*/
- Sự khác nhau giữa biến kiểu int và biến kiểu float ở chỗ: biến kiểu int luôn luôn nhận giá trị nguyên trong quá trình tính toán còn biến kiểu float có thể nhận cả giá trị không nguyên.

## 5. KHAI BÁO VÀ TOÁN TỬ GÁN



**Toán tử gán (=)** là toán tử dùng để gán các giá trị (số, ký tự, ...) hoặc giá trị của một biểu thức hoặc giá trị của một biến cho một biến khác.

*Toán tử gán có thể kết hợp với các toán tử số học.*

## 5. KHAI BÁO VÀ TOÁN TỬ GÁN

### ❖ Ví dụ sử dụng toán tử gán:

Toán tử	Ví dụ	Ý nghĩa
=	<code>a1 = bt;</code>	<code>a1</code> là một biến, <code>bt</code> là một biểu thức toán học nào đó. Tác dụng của câu lệnh này: trước tiên tính biểu thức <code>bt</code> , sau đó gán giá trị tính được cho biến <code>a1</code> .
=	<code>a1 = 10;</code>	Gán số 10 cho <code>a1</code>
=	<code>float x;</code> <code>x = 10.5f;</code> <code>x = 2*x - 2.5f;</code>	Biến <code>x</code> sẽ nhận giá trị 18.5
=	<code>int a1, a2, a3;</code> <code>a1 = a2 = a3 = 8;</code>	Biến <code>a1</code> , <code>a2</code> , <code>a3</code> nhận giá trị 8
=	<code>int b1, b2, b3;</code> <code>b3 = 2;</code> <code>b2 = b3;</code> <code>b1 = b2;</code>	Biến <code>b1</code> , <code>b2</code> , <code>b3</code> nhận giá trị 2



## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH

- ❖ **Xuất kết quả:** Sử dụng `System.out.println` hoặc `System.out.print` hoặc `System.out.printf`
  - Cú pháp: `System.out.println(data);` hoặc `System.out.print(data);`  
hoặc `System.out.printf(data);` hoặc `System.out.format(data);`
    - ✓ `System.out.println`: xuất kết quả và con trỏ chuột xuống dòng
    - ✓ `System.out.print`: xuất kết quả và con trỏ chuột không xuống dòng
    - ✓ `System.out.printf`: xuất và định dạng kết quả
    - ✓ `System.out.format`: xuất và định dạng kết quả
  - `System`: tên lớp.
  - `System.out`: là một trường của lớp `System` quy định việc xuất dữ liệu tiêu chuẩn của Java.
  - `println`: là phương thức có tác dụng in xuống dòng.
  - `print`: là phương thức có tác dụng in không xuống dòng.
  - `printf`: là phương thức có tác dụng in và định dạng kết quả.

## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH

### ❖ Các bộ định dạng có sẵn trong printf

- %t: Định dạng ngày / giờ
- %%: Dấu phần trăm
- \%%: Dấu phần trăm

## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH

### ❖ Các bộ định dạng có sẵn trong printf


Chuỗi định dạng	Đại diện cho kiểu ký tự	Ý nghĩa
%c	char	Xuất ra một ký tự
%s	char *	Xuất ra một chuỗi ký tự
%d	int, short	Xuất ra một số nguyên dưới dạng thập phân
%u	unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng thập phân không dấu
%x	int, short, unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng thập lục phân
%o	int, short, unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng bát phân
%f	float	Xuất ra một số thực
%e	float	Xuất ra một số thực dưới dạng số mũ

Chuỗi định dạng	Đại diện cho kiểu ký tự	Ý nghĩa
%ld	long	Xuất ra số nguyên chính xác kép ở dạng thập phân
%lu	unsigned long	Xuất ra số nguyên chính xác kép ở dạng thập phân không dấu
%lo	long, unsigned long	Xuất ra số nguyên chính xác kép trong hệ bát phân
%lx	long, unsigned long	Xuất ra số nguyên chính xác kép ở hệ thập lục phân
%lf	double, unsigned long	Xuất ra số thực chính xác gấp đôi
%a	double	Xuất ra một số thực chính xác kép thập lục phân
%g	float	Xuất ra một số thực dưới dạng phù hợp nhất

## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH


### ❖ Ví dụ 1:

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        // in chuỗi  
        System.out.print("hello");  
        System.out.print("everyone");  
    }  
}
```



```
Console x  
<terminated> Example [Java Application]  
helloeveryone|
```

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        // in chuỗi  
        System.out.println("hello");  
        System.out.println("everyone");  
    }  
}
```



```
Console x  
<terminated> Example [Java Application]  
hello  
everyone
```

## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH

### ❖ Ví dụ 2:

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        // in số  
        System.out.print(123);  
        System.out.print(456);  
    }  
}
```



Console ✕  
<terminated> Example [Java Application]  
123456|

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        // in số  
        System.out.println(123);  
        System.out.println(456);  
    }  
}
```

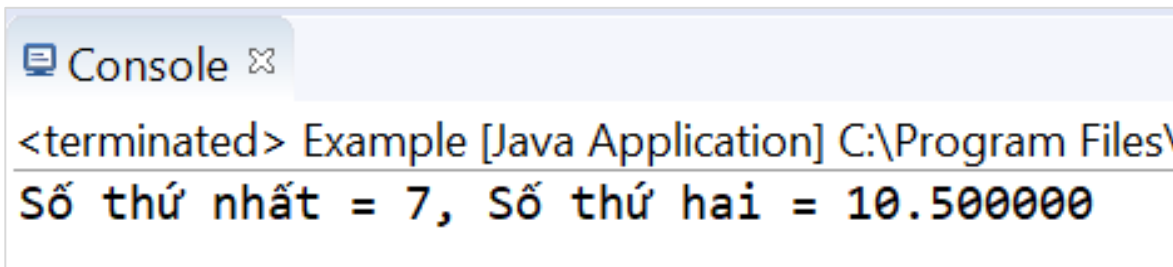


Console ✕  
<terminated> Example [Java Application]  
123  
456

## 6. ĐƯA KẾT QUẢ LÊN MÀN HÌNH

### ❖ Ví dụ 3:

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        int num1 = 7;  
        float num2 = 10.5f;  
        System.out.printf("Số thứ nhất = %d, Số thứ hai = %f", num1, num2);  
    }  
}
```



Console ✕

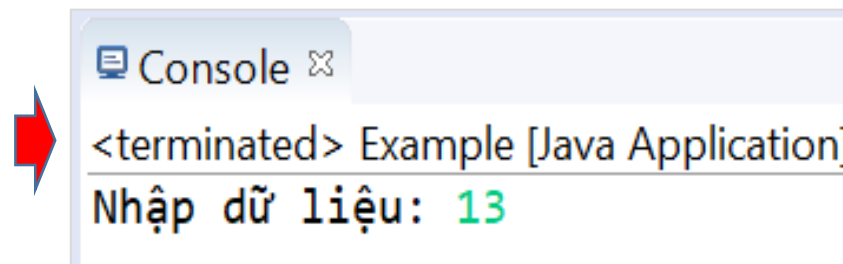
<terminated> Example [Java Application] C:\Program Files\  
Số thứ nhất = 7, Số thứ hai = 10.500000

# 7. NHẬP SỐ LIỆU TỪ BÀN PHÍM

❖ Nhập dữ liệu từ bàn phím được thực hiện thông qua lớp **Scanner**.

- Cú pháp:
  - ✓ `Scanner scanner = new Scanner(System.in);`
  - ✓ `scanner.nextX();`
- Tạo đối tượng `scanner` của lớp `Scanner`.
- Gọi phương thức `nextX()`, với `X` là tên kiểu dữ liệu. Ví dụ: `scanner.nextInt()` nhập vào kiểu số nguyên.

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        // Các câu lệnh xử lý
        Scanner scanner = new Scanner(System.in);
        System.out.print("Nhập dữ liệu: ");
        scanner.nextInt();
    }
}
```



**Nhập đúng kiểu dữ liệu**

# 7. NHẬP SỐ LIỆU TỪ BÀN PHÍM

❖ Nhập dữ liệu từ bàn phím được thực hiện thông qua lớp **Scanner**.

- Cú pháp:
  - ✓ `Scanner scanner = new Scanner(System.in);`
  - ✓ `scanner.nextX();`
- Tạo đối tượng `scanner` của lớp `Scanner`.
- Gọi phương thức `nextX()`, với `X` là tên kiểu dữ liệu. Ví dụ: `scanner.nextInt()` nhập vào kiểu số nguyên.

```
Console ✕
<terminated> Example [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Aug 25, 2022, 9:12:23 PM)
Nhập dữ liệu: abc
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:939)
    at java.base/java.util.Scanner.next(Scanner.java:1594)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
    at Example.main(Example.java:7)
```

**Nhập sai kiểu dữ liệu**



# 7. NHẬP SỐ LIỆU TỪ BÀN PHÍM

❖ Các phương thức thường dùng trong lớp Scanner.

Tên phương thức	Tác dụng
nextBoolean	Nhập vào kiểu Boolean (true – false) từ bàn phím
nextByte	Nhập vào kiểu dữ liệu Byte
nextShort	Nhập vào kiểu Short (số nguyên từ -32768 đến 32767)
nextInt	Nhập vào kiểu số nguyên từ bàn phím
nextFloat	Nhập vào kiểu số thực
nextDouble	Nhập vào kiểu Double (số thực lớn hơn float)
next/nextLine	Nhập vào kiểu String
nextLong	Nhập vào số nguyên lớn

## 8. VÍ DỤ CHƯƠNG TRÌNH ĐƠN GIẢN

❖ **Ví dụ 1:** Viết chương trình xuất ra dòng chữ **Hello world**

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        System.out.println("Hello world");  
    }  
}
```

❖ **Ví dụ 2:** Viết chương trình tính tổng hai số

```
public class Example {  
    public static void main(String[] args) {  
        // Các câu lệnh xử lý  
        int num1 = 5; //số thứ nhất  
        int num2 = 10; //số thứ hai  
        System.out.printf("Tổng của %d và %d = %d", num1, num2, (num1 + num2));  
    }  
}
```

## 1. Cài đặt và cấu hình Java

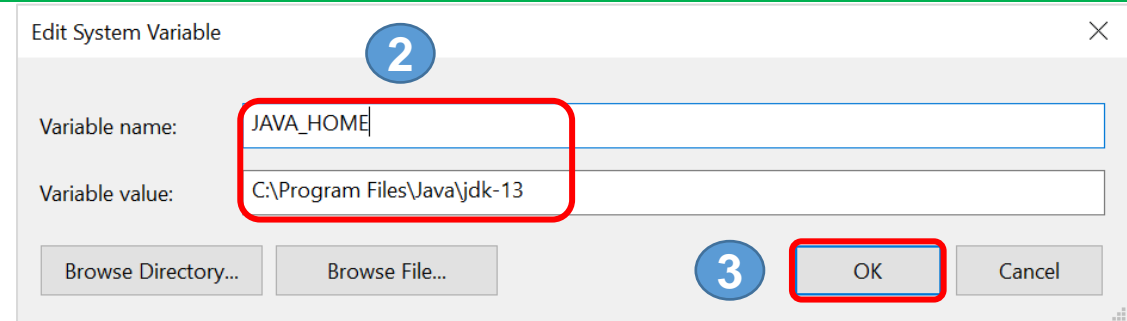
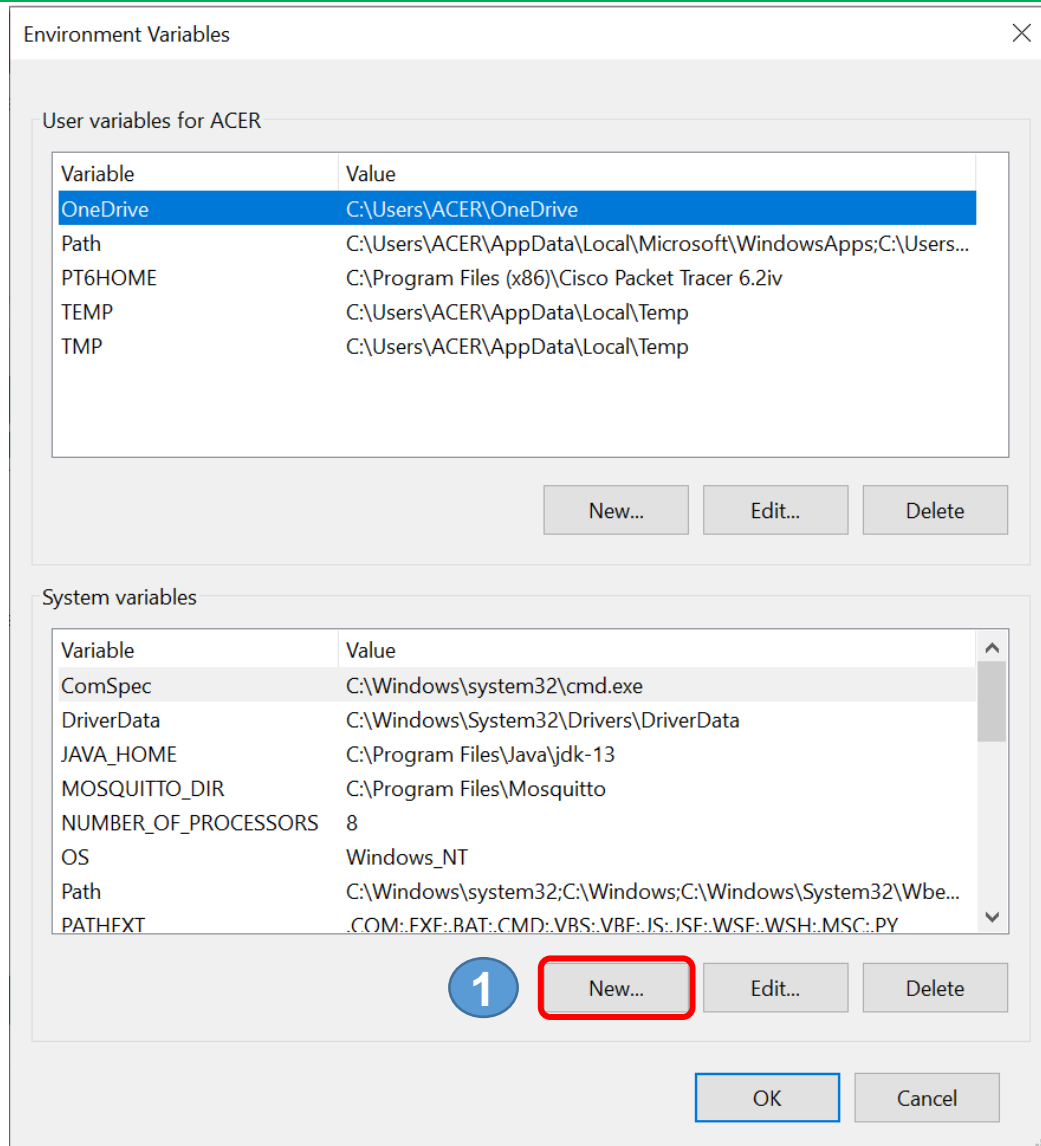
### ❖ Download và cài đặt JDK

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

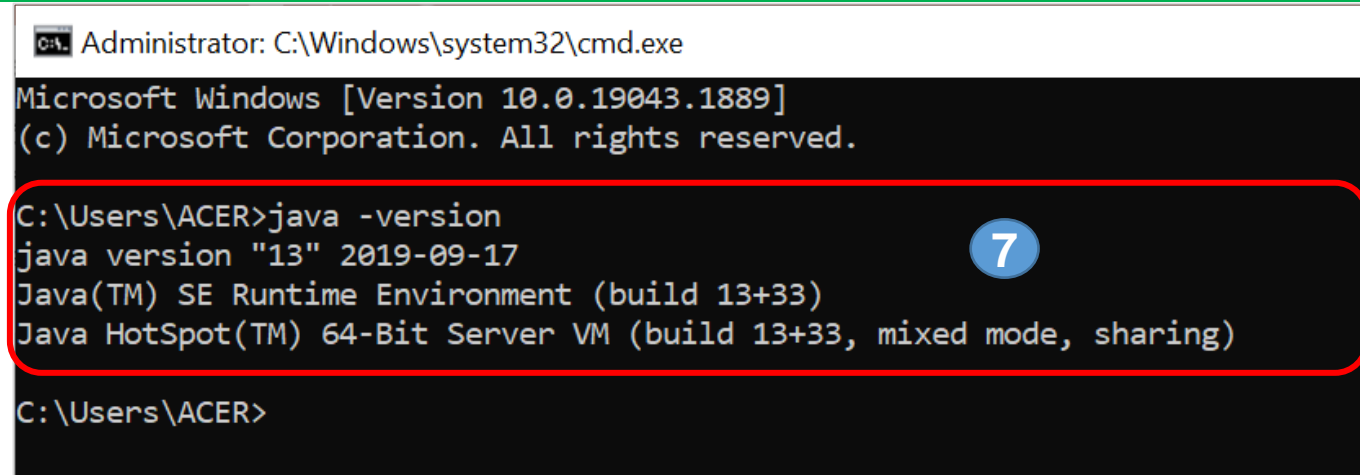
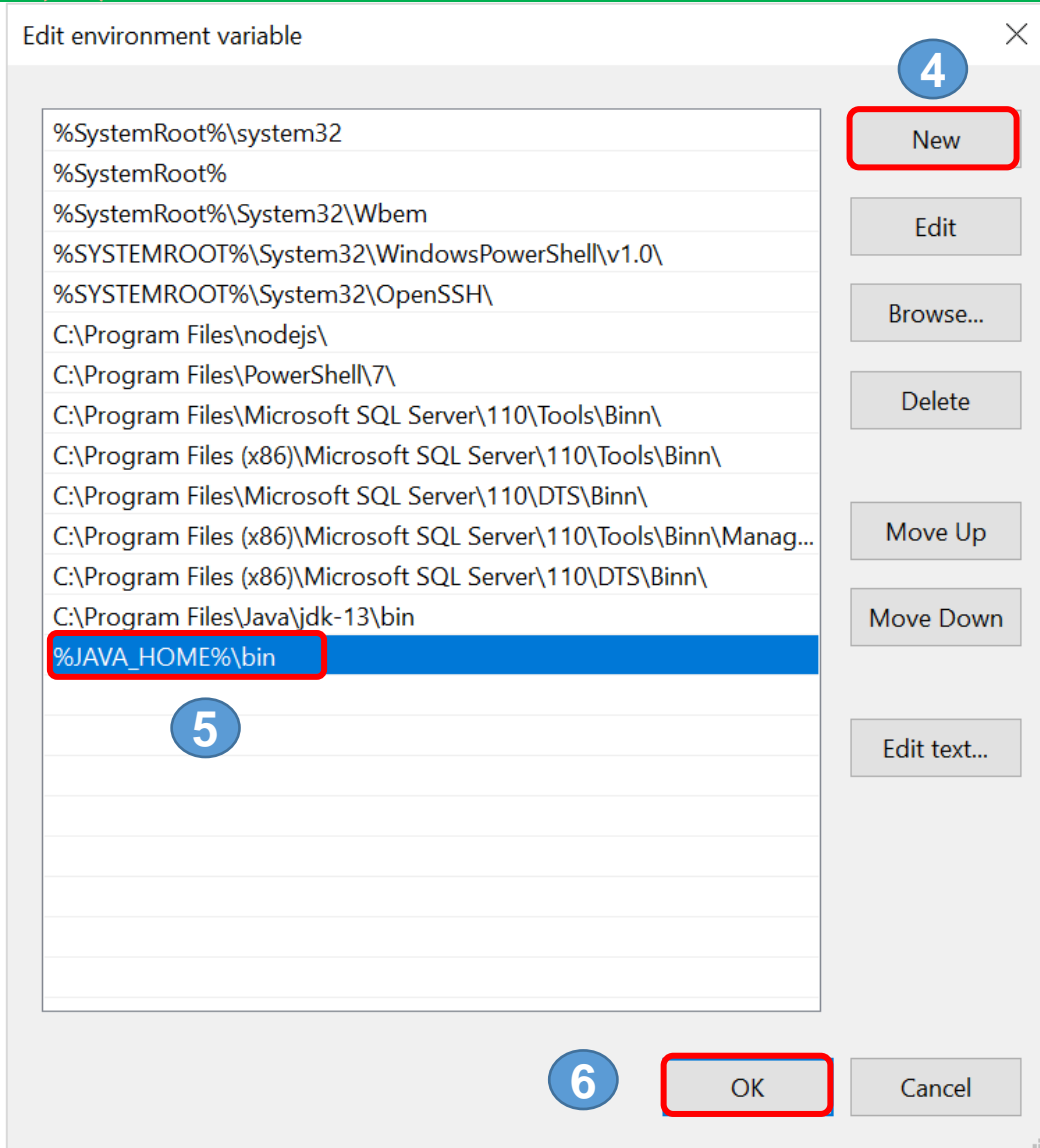
### ❖ Thiết lập biến môi trường và cấu hình cho Java

- Control Panel → System and Security → System → Advanced system settings → Advanced → Environment Variables...
  - ✓ Khai báo biến môi trường: System Variables → New: Variable name: **JAVA\_HOME**;  
Variable value: **chọn đường dẫn chứa file cài đặt jdk**
  - ✓ Hiệu chỉnh biến môi trường: System Variables → PATH → Edit → New →  
**%JAVA\_HOME%\bin**
- Run → cmd → java -version

# 9. VẬN HÀNH CHƯƠNG TRÌNH TRÊN MÁY TÍNH



# 9. VẬN HÀNH CHƯƠNG TRÌNH TRÊN MÁY TÍNH



## 2. Cài đặt và cấu hình Eclipse

### ❖ Download và cài đặt Eclipse

- <https://www.eclipse.org/downloads/>.
- Giải nén → chạy file eclipse.exe → chọn Launch

### ❖ Sử dụng Eclipse

- Thay đổi Workspace: File → Switch Workspace → Other... → Browse... → chọn đường dẫn muốn thay đổi → Launch.
- Kiểm tra phiên bản java: Window → Preferences → Compiler.
- Tạo Project: File → New → Java Project → Project name: nhập tên → Finish.
- Tạo Package: Right click trên Project → New → Package → Name: nhập tên → Finish.
- Tạo Class: Right click trên Package → New → Class → Name: nhập tên → Finish.

## 2. Cài đặt và cấu hình Eclipse

### ❖ Biên dịch chương trình trên Eclipse

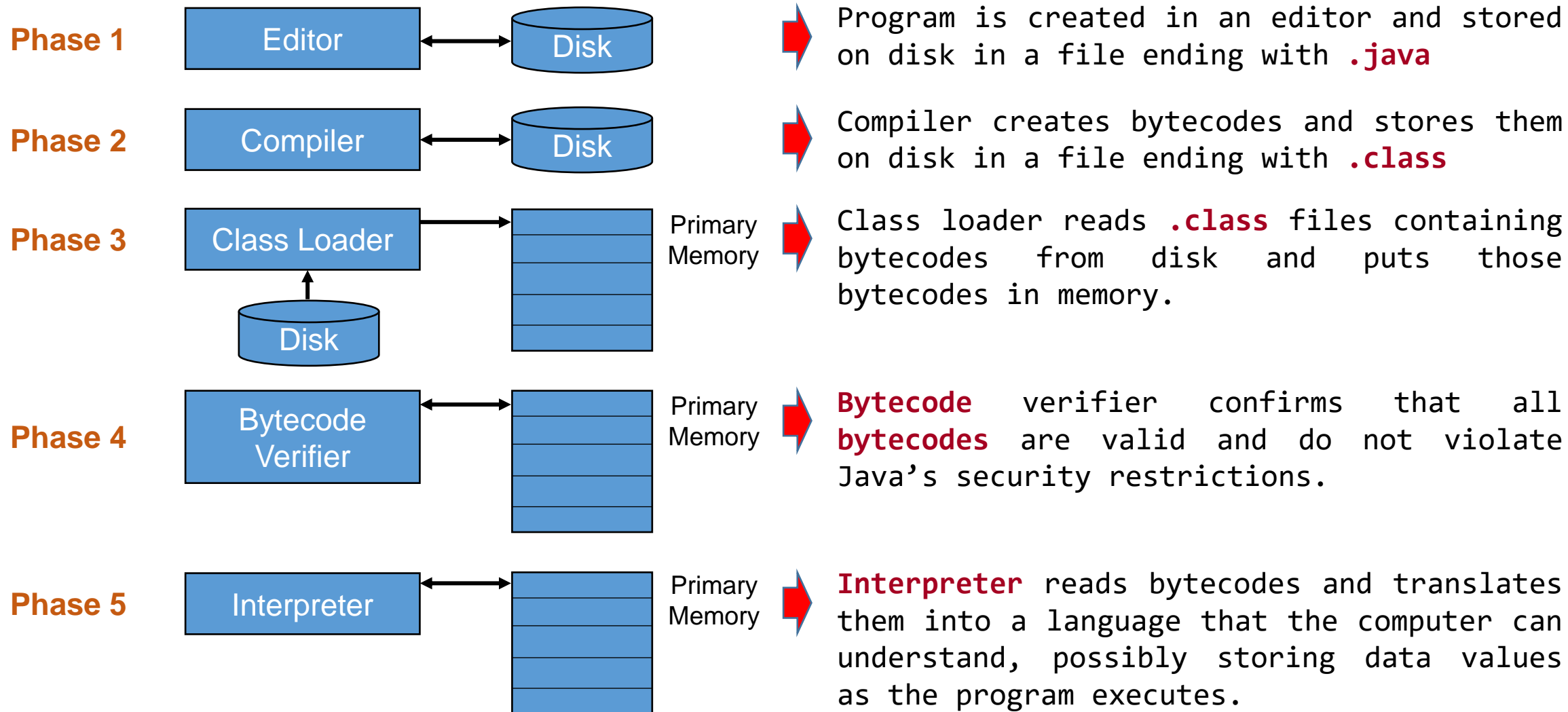
- Cách 1: Chọn vào biểu tượng Run trên thanh công cụ.
- Cách 2: Chọn vào chữ Run trên thanh công cụ và chọn Run (hoặc nhấn tổ hợp phím Ctrl + F11).
- Cách 3: Nhấp chuột phải vào tên Class và chọn Run As → Java Application.

❖ **Các chương trình Java trải qua 5 giai đoạn chính:**

1. **Editor:** Lập trình viên viết chương trình và lưu vào máy tính với định dạng file .java.
2. **Compiler:** Biên dịch chương trình thành bytecodes (định dạng .class) - *nhờ bước trung gian này mà Java được viết 1 lần và chạy trên các hệ điều hành khác nhau.*
3. **Class Loader:** Đọc file .class chứa mã bytecodes và lưu vào trong bộ nhớ.
4. **Bytecode Verifier:** Đảm bảo rằng mã bytecodes là hợp lệ và không vi phạm các vấn đề về bảo mật của Java.
5. **Interpreter:** Dịch bytecodes thành mã máy để máy tính có thể hiểu được và sau đó thực thi chương trình.



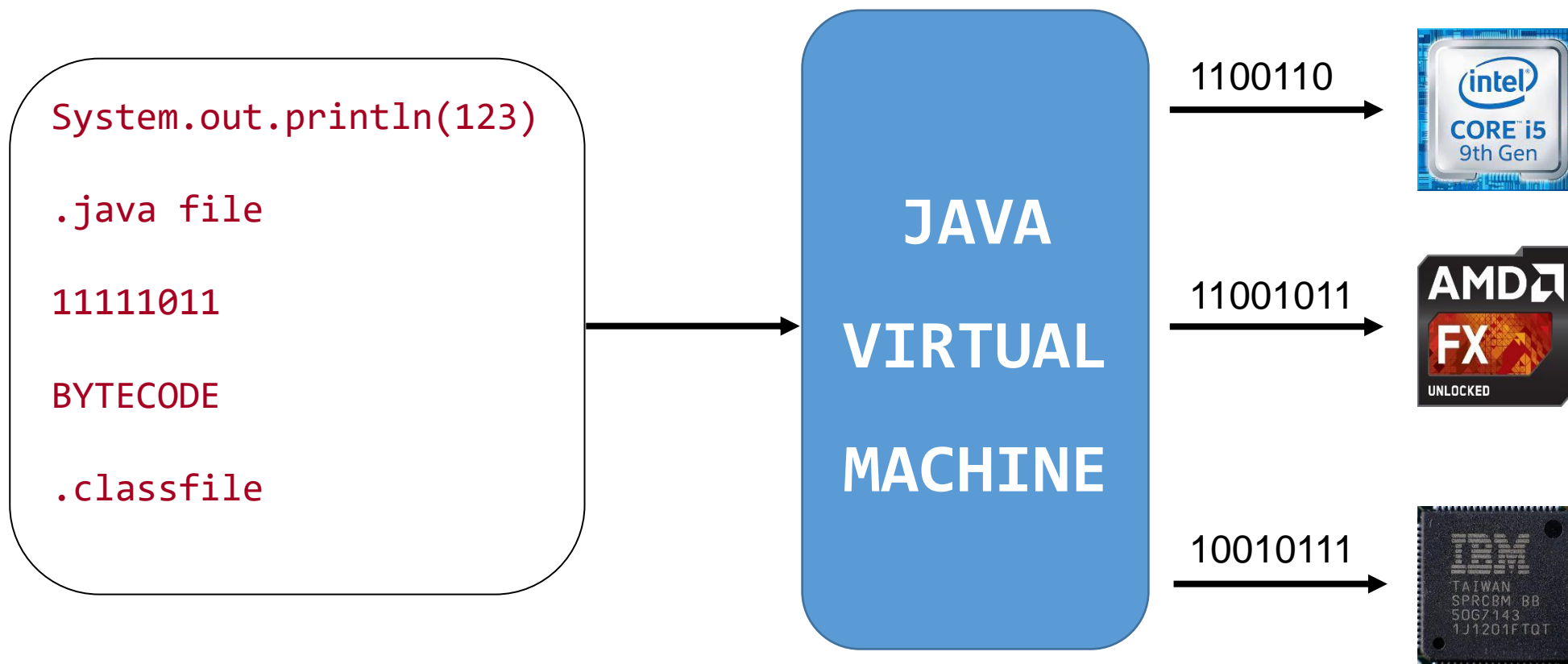
# 9. VẬN HÀNH CHƯƠNG TRÌNH TRÊN MÁY TÍNH



Typical Java environment

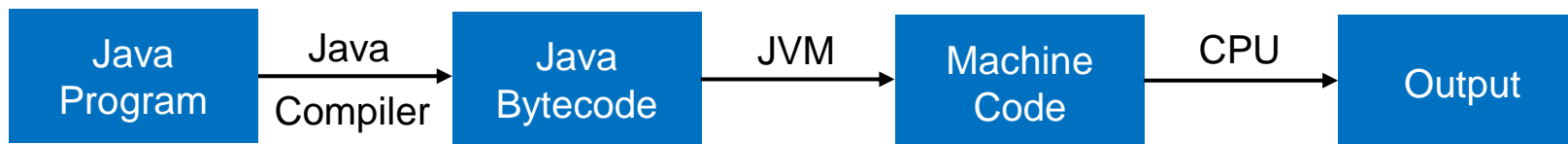
- ❖ **Java** là ngôn ngữ lập trình và là nền tảng điện toán máy tính để phát triển ứng dụng trên máy tính / di động và nhiều thiết bị khác.
- ❖ **Java** được phát hành lần đầu tiên bởi Sun microsystem vào năm 1995 và sau đó được Tập đoàn Oracle mua lại.
- ❖ **Platform** (nền tảng) là một nhóm các công nghệ được sử dụng làm cơ sở cho các ứng dụng, quy trình hoặc công nghệ khác phát triển.
  - **Ví dụ:** *Trong máy tính cá nhân, một platform là phần cứng cơ bản (máy tính) và phần mềm (hệ điều hành) để các ứng dụng có thể chạy.*
- ❖ Để máy tính chạy các ứng dụng phần mềm, các ứng dụng phải ở ngôn ngữ máy được mã hóa nhị phân. Do đó, trong lịch sử, các chương trình ứng dụng được viết cho một nền tảng sẽ không hoạt động trên nền tảng khác.

Bằng cách sử dụng Máy ảo Java, vấn đề mã nguồn phụ thuộc vào nền tảng có thể được giải quyết. Nhưng làm thế nào nó hoạt động trên các bộ xử lý khác nhau và các hệ điều hành khác nhau?.

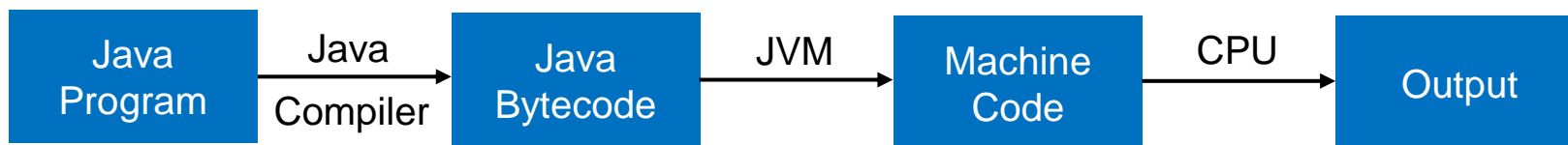


- ❖ **Bước 1:** Mã lệnh xuất 123 là **System.out.println(123)** và được lưu dưới dạng tệp .java.
- ❖ **Bước 2:** Sử dụng trình biên dịch java, mã được chuyển đổi thành mã trung gian được gọi là bytecode. Đầu ra là một .class file.
- ❖ **Bước 3:** Bytecode chỉ có một nền tảng ảo JVM (Java Virtual Machine) có thể hiểu.
- ❖ **Bước 4:** Máy ảo này nằm trong RAM của hệ điều hành. Khi máy ảo nhận được mã bytecode sẽ tự động nhận dạng nền tảng đang hoạt động để chuyển đổi mã bytecode thành mã máy gốc.
- ❖ **Code Java** khi được biên dịch có thể chạy trên tất cả các nền tảng PC, điện thoại di động và các thiết bị điện tử có hỗ trợ java.

- ❖ **Trình biên dịch** Java không tạo mã thực thi riêng cho một máy cụ thể. Thay vào đó, Java tạo ra một định dạng duy nhất được gọi là bytecode. Nó thực thi theo các quy tắc được đặt ra trong đặc tả của máy ảo Java.
- ❖ **Bytecode** có thể hiểu đối với bất kỳ JVM được cài đặt trên bất kỳ hệ điều hành. *Mã nguồn Java có thể chạy trên tất cả các hệ điều hành.*



- ❖ **Java Platform** (nền tảng Java) là một tập hợp các chương trình giúp phát triển và chạy các chương trình được viết bằng ngôn ngữ lập trình Java.
- ❖ **Java Platform** bao gồm một công cụ thực thi (execution engine), trình biên dịch (compiler) và một bộ thư viện Java.
- ❖ Java là ngôn ngữ độc lập với nền tảng. Nghĩa là chỉ cần cài Java Platform thì bất kỳ bộ xử lý hoặc hệ điều hành nào cũng có thể chạy Java.



❖ IDE (Integrated Development Environment): Môi trường phát triển tích hợp là một ứng dụng tạo điều kiện đầy đủ để lập trình.

❖ **Ưu điểm**

- Tiết kiệm thời gian và công sức lập trình
- Thực thi các tiêu chuẩn của dự án lập trình
- Quản lý dự án

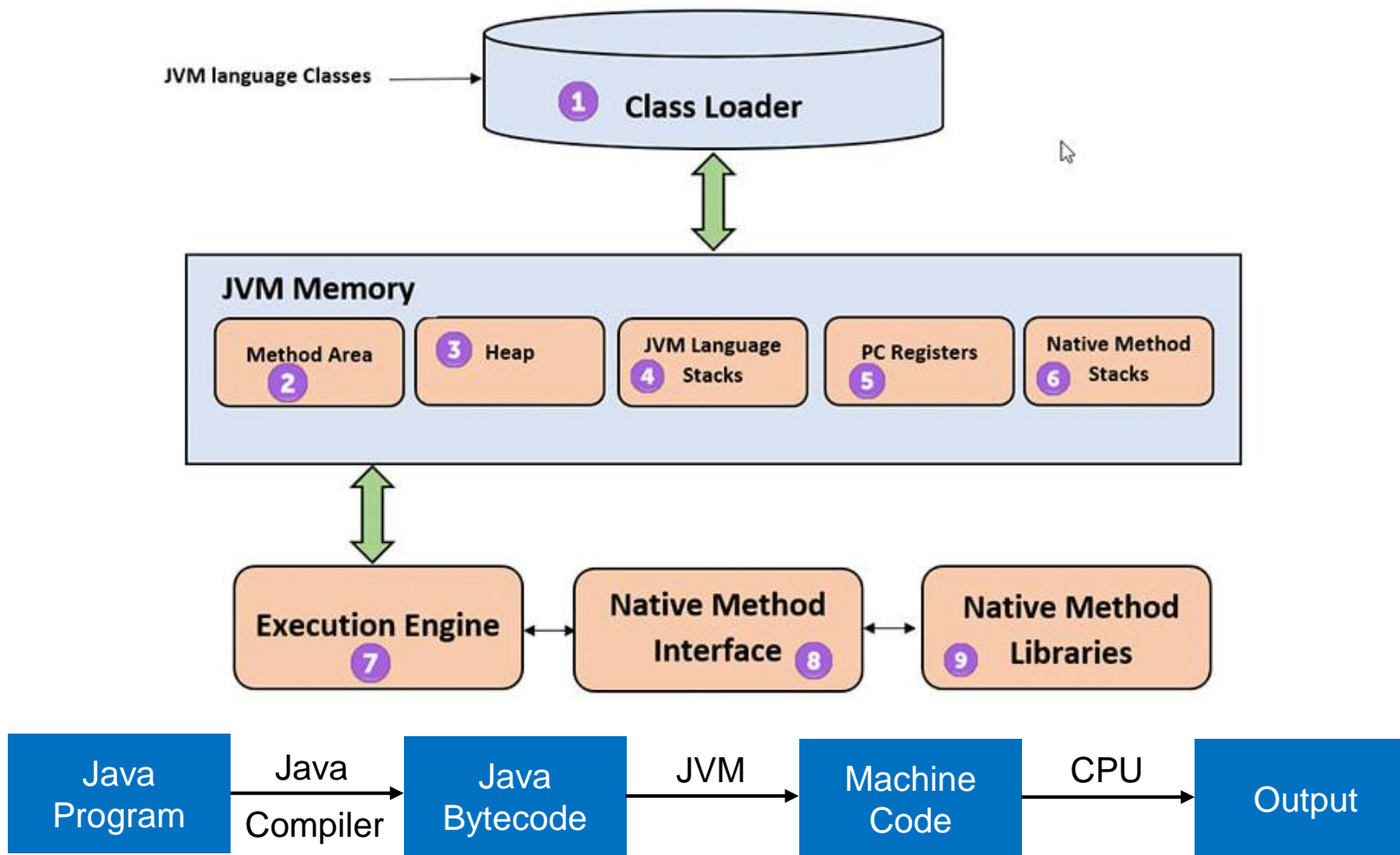
## ❖ Danh sách các IDE

- Eclipse
- IntelliJ
- NetBeans
- JDeveloper
- MyEclipse
- BlueJ
- Visual Studio Code



- ❖ **Máy ảo Java (JVM – Java Virtual Machine)** là một thiết bị trừu tượng (ảo) có thể giúp máy tính chạy các chương trình Java. Nó cung cấp môi trường runtime mà trong đó Java Bytecode có thể thực thi.
- ❖ JVM có sẵn cho nhiều nền tảng (Windows, Linux, MAC ...).
- ❖ JVM, JRE và JDK phụ thuộc vào nền tảng, bởi vì cấu hình của mỗi hệ điều hành (OS) là khác nhau. Nhưng Java là độc lập nền tảng.
- ❖ Các nhiệm vụ chính của JVM
  - Tải code
  - Kiểm tra code
  - Thực thi code
  - Cung cấp môi trường runtime

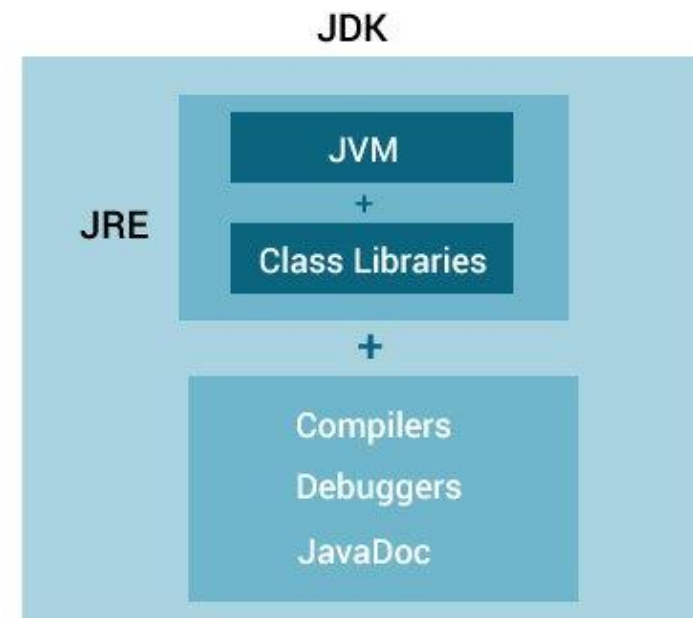
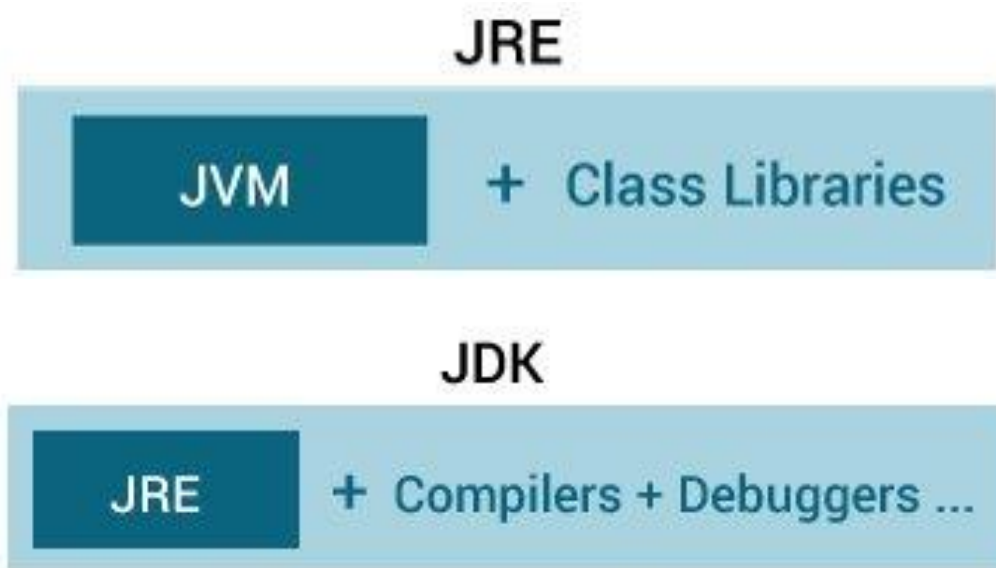
## Kiến trúc của máy ảo Java



- ❖ **ClassLoader** là một hệ thống con được sử dụng để tải các class file. Nó thực hiện ba chức năng chính: Loading, Linking và Initialization (khởi tạo).
- ❖ **Method Area** lưu trữ các cấu trúc lớp (class) như: metadata, the constant runtime pool và code của các method.
- ❖ **Heap** lưu trữ tất cả các Object, các biến đối tượng liên quan và mảng (Array) của chúng. Bộ nhớ này là phổ biến và được chia sẻ trên nhiều luồng.
- ❖ **JVM language Stacks** lưu trữ các biến cục bộ (local variables) và một phần kết quả của nó. Mỗi luồng có JVM stack riêng, được tạo đồng thời khi luồng được tạo. Một khung mới được tạo bất cứ khi nào một phương thức được gọi và nó sẽ bị xóa khi quá trình gọi phương thức hoàn tất.

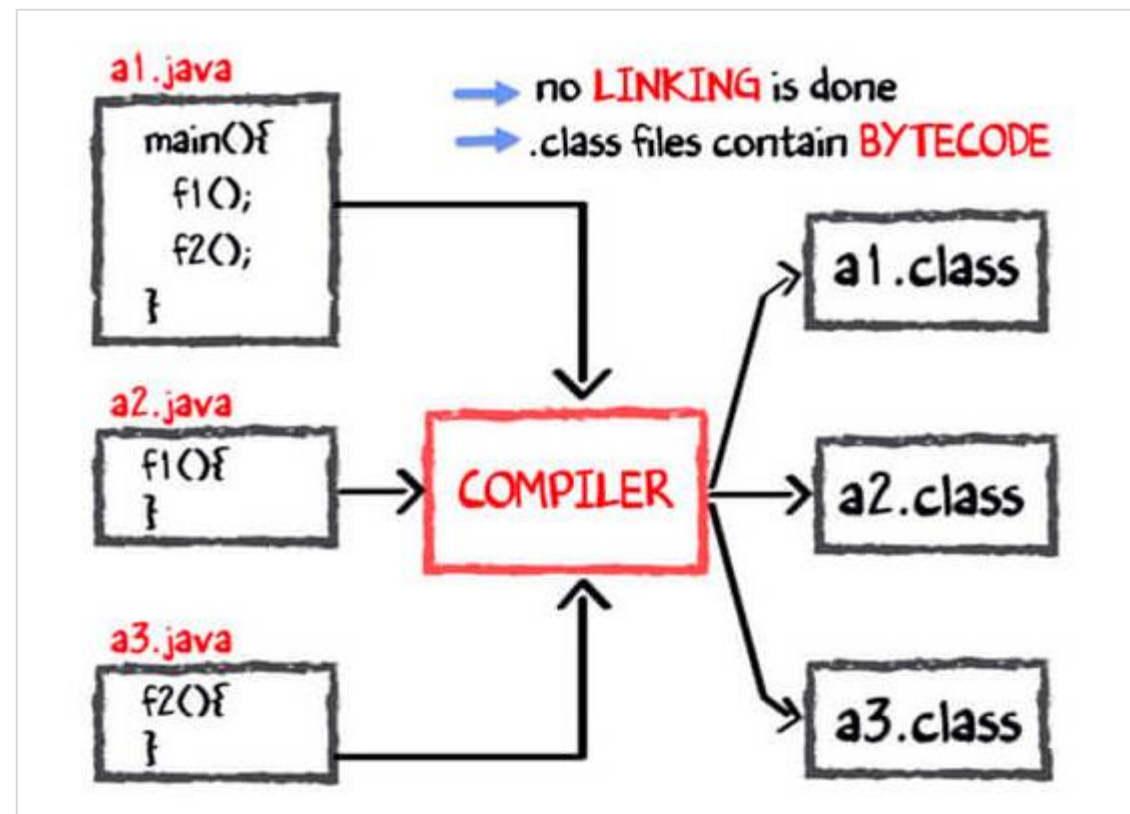
- ❖ **PC Register** lưu trữ địa chỉ của hướng dẫn máy ảo Java hiện đang thực thi. Trong Java, mỗi luồng có thanh ghi PC riêng biệt.
- ❖ **Native Method Stacks** giữ hướng dẫn của mã gốc phụ thuộc vào thư viện riêng được viết bằng ngôn ngữ khác thay vì Java.
- ❖ **Execution Engine** là một loại phần mềm được sử dụng để kiểm tra phần cứng, phần mềm hoặc hệ thống hoàn chỉnh. Công cụ thực hiện kiểm tra không bao giờ mang bất kỳ thông tin nào về sản phẩm được kiểm tra.
- ❖ **Native Method interface** là một Framework cho phép mã Java đang chạy trong JVM được gọi bởi các thư viện và ứng dụng gốc.
- ❖ **Native Libraries** là một tập hợp các thư viện riêng (C, C++) cần thiết cho Execution Engine (công cụ thực thi).

- ❖ **JRE (Java Runtime Environment)** được sử dụng để cung cấp môi trường runtime. Nó là trình triển khai của JVM. JRE bao gồm tập hợp các thư viện và các file khác mà JVM sử dụng tại runtime. Trình triển khai của JVM cũng được công bố bởi các công ty khác ngoài Sun Micro Systems.
- ❖ **JDK (Java Development Kit)** bao gồm JRE và các Development Tool.



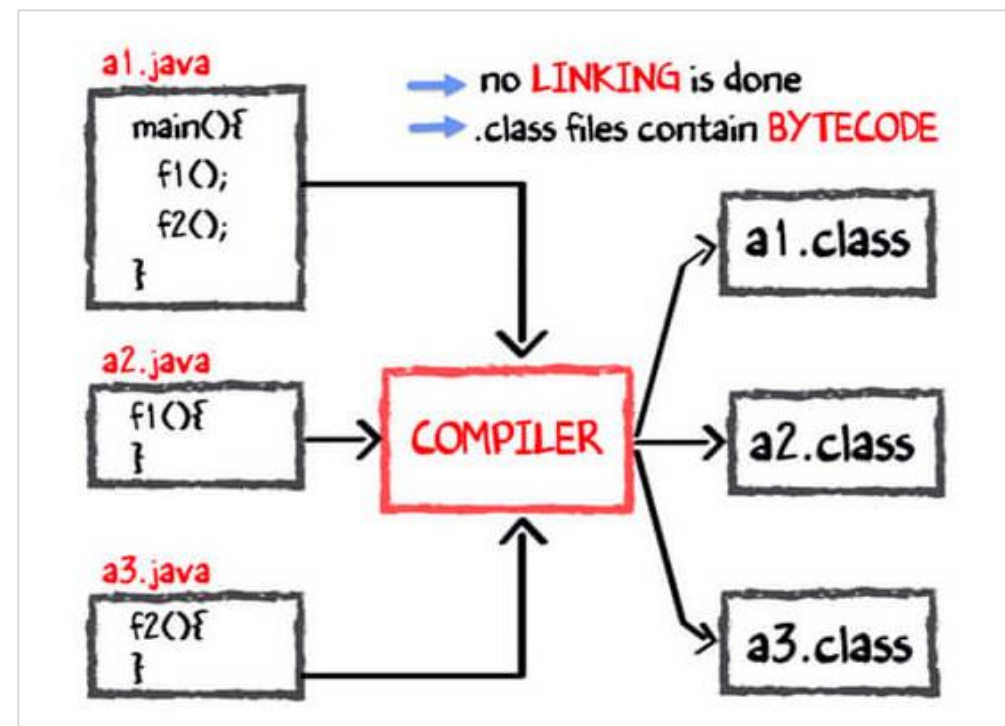
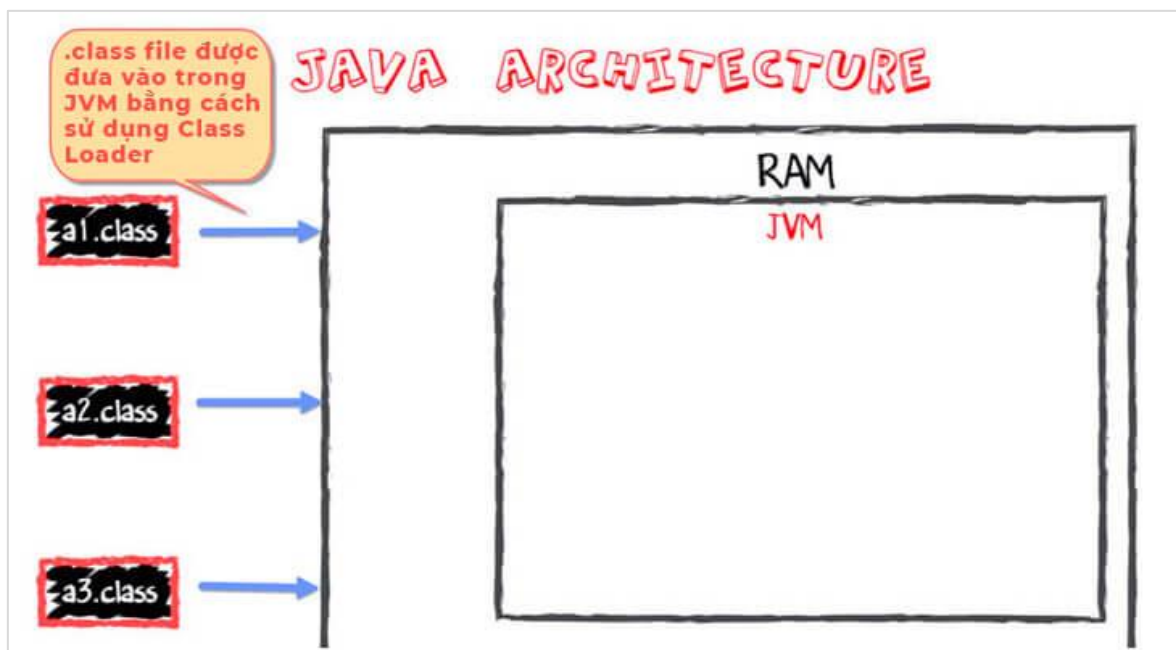
## ❖ Ví dụ: biên dịch và thực thi mã Java trong JVM

- Hàm main chứa hai phương thức f1 và f2 (thực hiện gọi hàm f1, f2).
- Hàm main được lưu trữ trong tệp a1.java
- f1 được lưu trữ trong tệp a2.java
- f2 được lưu trữ trong tệp a3.java



## 9. VẬN HÀNH CHƯƠNG TRÌNH TRÊN MÁY TÍNH

- ❖ Trình biên dịch sẽ biên dịch ba tệp và tạo ra 3 tệp .class tương ứng chứa bytecode. Không giống như C, không có liên kết được thực hiện.
- ❖ Java VM hoặc Máy ảo Java nằm trên RAM. Trong quá trình thực thi, sử dụng class loader, class files được đưa vào RAM. Tại đây bytecode được xác minh cho tính bảo mật.





- ❖ Tiếp theo, Execution Engine sẽ chuyển đổi bytecode thành mã máy gốc trong thời gian biên dịch. Vì vậy Java chạy tương đối chậm.
- ❖ JIT (Just-In-Time) là một phần của Java Virtual Machine (JVM). JIT chuyển đổi bytecode thành mã máy gốc có cùng cấp độ ➔ **Interpreter**

