



Tổng quan

Môn học: Kỹ thuật lập trình [Tuần 1]

Giảng Viên: ThS. Trần Đức Hiếu

Lập trình là gì ?



Thuật giải

*Nhắc lại kiến thức từ môn "**Cơ sở lập trình**"*

Thuật toán là một tập các quy tắc (hay quy trình cụ thể) để giải quyết một vấn đề nào đó với số lần thực hiện là hữu hạn, dựa trên các dữ liệu đầu vào được cung cấp cho quá trình xử lý của chương trình

Ví dụ

Xây dựng thuật toán giải phương trình bậc nhất $ax + b = c$

(Với a, b, c là các số thực)

1 – Nếu $a = 0$

* $b = c$ thì phương trình có vô số nghiệm (Nghiệm bất kỳ)

* $b \neq c$ thì phương trình vô nghiệm

2 – Nếu $a \neq 0$

Phương trình có 1 nghiệm duy nhất $x = (c - b) / a$

Đặc tính của thuật toán

- ❖ Chính xác, Đúng
- ❖ Tổng quát (*Luôn đúng dẫn trong nhiều tình huống khác nhau – Còn gọi là tính phổ dụng*)
- ❖ Rõ ràng
- ❖ Hữu hạn (*Số lần thực hiện các bước là xác định, có tính dừng*)

Các phương pháp biểu diễn

- ❖ Biểu diễn bằng ngôn ngữ tự nhiên
(*Native language*)
- ❖ Biểu diễn bằng mã giả
(*Pseudo code*)
- ❖ Biểu diễn bằng lưu đồ
(*Flow chart*)

Minh họa bằng ngôn ngữ tự nhiên

Khi biểu diễn thuật toán theo ngôn ngữ tự nhiên, người ta sử dụng ngôn ngữ thường ngày để liệt kê các bước của thuật toán

1 – Nhập các giá trị a, b, c

2 – Nếu $a = 0$

Nếu $b = c$ thì thông báo phương trình có vô số nghiệm

nếu $b \neq c$ thì thông báo phương trình vô nghiệm

3 – Nếu $a \neq 0$

Thông báo: Phương trình có 1 nghiệm duy nhất

$$x = (c - b) / a$$

Minh họa bằng ngôn ngữ tự nhiên

Thuật toán giải phương trình $ax + b = c$

1 – Nhập các giá trị a, b, c

2 – Nếu $a = 0$

Nếu $b = c$ thì thông báo phương trình có vô số nghiệm

nếu $b \neq c$ thì thông báo phương trình vô nghiệm

3 – Nếu $a \neq 0$

Thông báo: Phương trình có 1 nghiệm duy nhất

$$x = (c - b) / a$$

Minh hoạ bằng mã giả

Khi thể hiện thuật toán bằng mã giả, người ta thường **vay mượn** các cú pháp của một ngôn ngữ lập trình nào đó để thể hiện thuật toán. Việc dùng mã giả vừa tận dụng được các khái niệm trong ngôn ngữ lập trình, vừa giúp người cài đặt dễ dàng nắm bắt nội dung thuật toán

Minh họa bằng mã giả

Thuật toán giải phương trình $ax + b = c$

Nhập a, b, c

If $a = 0$ then

if $b = c$ then

write('có vô số nghiệm');

if $b \neq c$ then

write('phương trình vô nghiệm');

Else

begin

$x := (c - b) / a;$

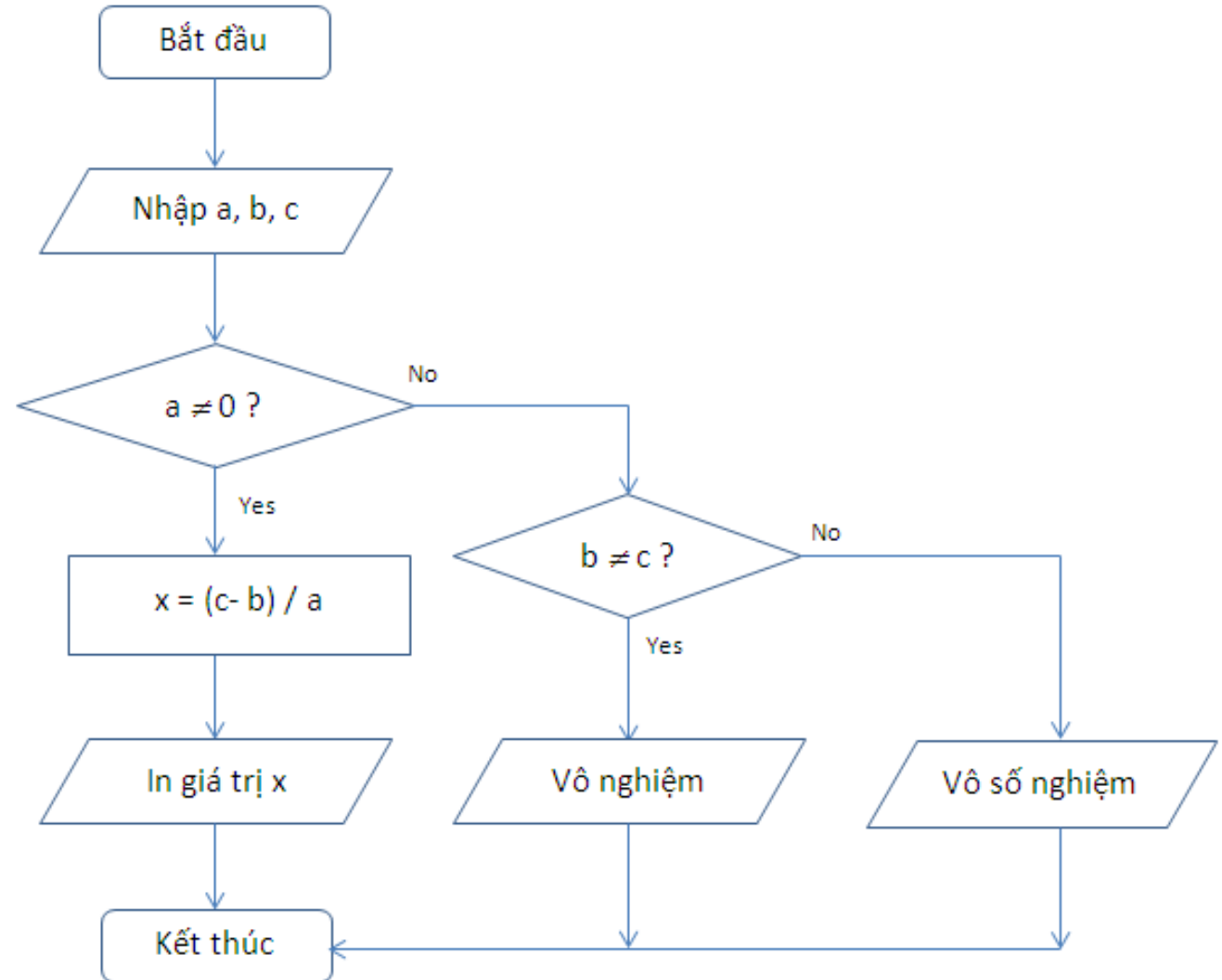
write('Phương trình có 1 nghiệm duy nhất $x =$ ', x);

end

Minh họa bằng lưu đồ

Lưu đồ hay sơ đồ khối là *một công cụ trực quan để diễn đạt các bước thực thi của thuật toán.*

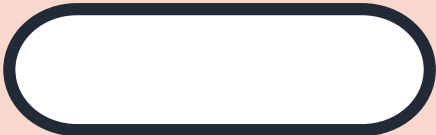


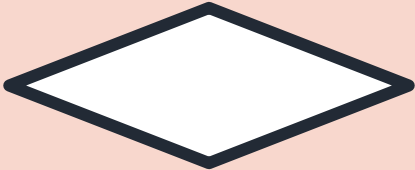
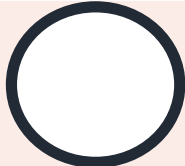
Biểu diễn thuật toán bằng lưu đồ sẽ giúp người đọc theo dõi được sự phân cấp các trường hợp và quá trình xử lý của thuật toán



Biểu diễn thuật toán bằng lưu đồ

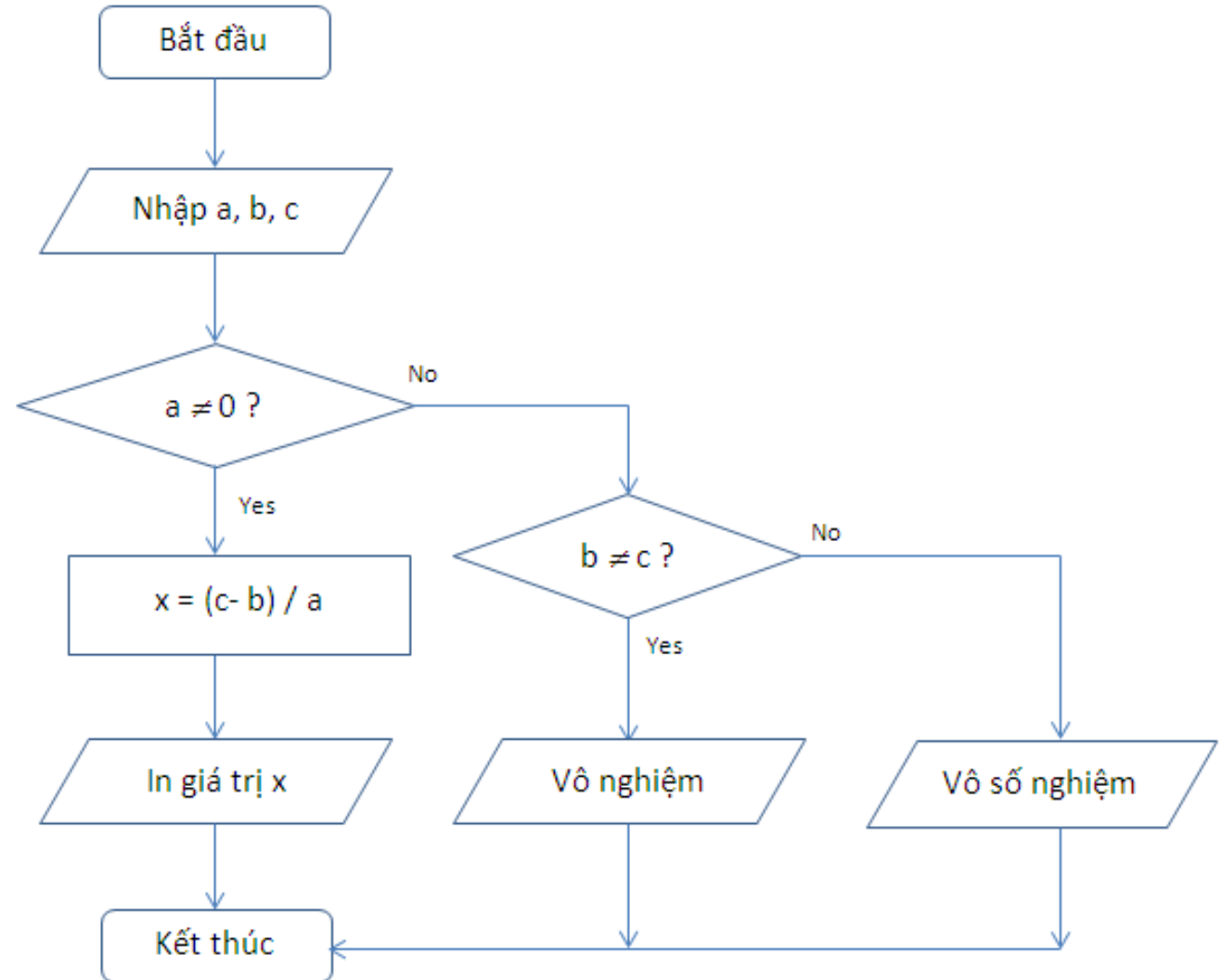
Khi sử dụng lưu đồ thuật toán, chúng ta phải biểu diễn được các quá trình: nhập dữ liệu (*input*), xử lý dữ liệu (*processing*), và xuất dữ liệu (*output*) dựa trên các ký hiệu hình học qui ước

Ký hiệu sử dụng trên lưu đồ

| Ký hiệu - Biểu tượng | Ý nghĩa sử dụng trong lưu đồ |
|---|--|
|  | Biểu thị bắt đầu hoặc kết thúc chương trình |
|  | Biểu thị hướng xử lý trong một quá trình / chương trình |
|  | Biểu thị thông tin vào hoặc thông tin ra (<i>Nhập hoặc xuất dữ liệu</i>) trong chương trình |
|  | Biểu thị một hoạt động (<i>Hay một quá trình</i>) trong thuật toán |
|  | Biểu thị một quyết định khi đứng trước một vấn đề logic cần phải lựa chọn (<i>hoặc phân nhánh xử lý trong chương trình</i>). |
|  | Điểm nối trên lưu đồ (<i>Sử dụng khi lưu đồ có kích thước lớn, phức tạp</i>) |

Minh họa bằng lưu đồ

Ví dụ: lưu đồ thuật toán giải phương trình $ax + b = c$



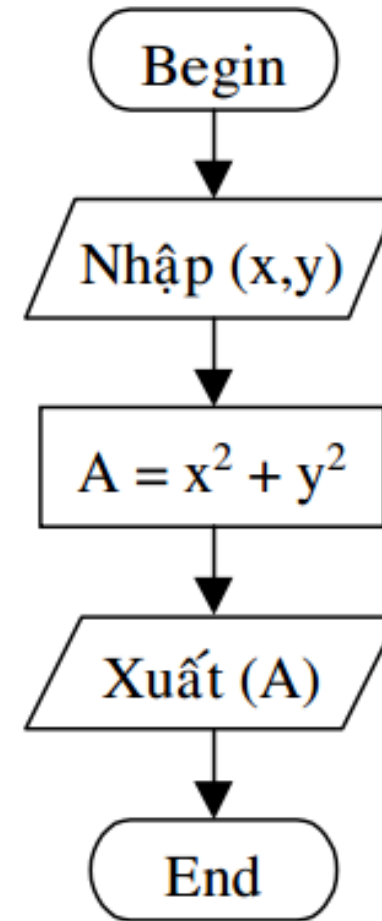
Các cấu trúc điều khiển

- ✓ Điều khiển tuần tự
- ✓ Điều khiển phân nhánh
(*Có lựa chọn hướng xử lý*)
- ✓ Điều khiển chu trình
(*Cấu trúc lặp*)

Cấu trúc tuần tự

Yêu cầu:

- Nhập 2 số x, y. Sau đó tính tổng bình phương của 2 số và in kết quả ra màn hình

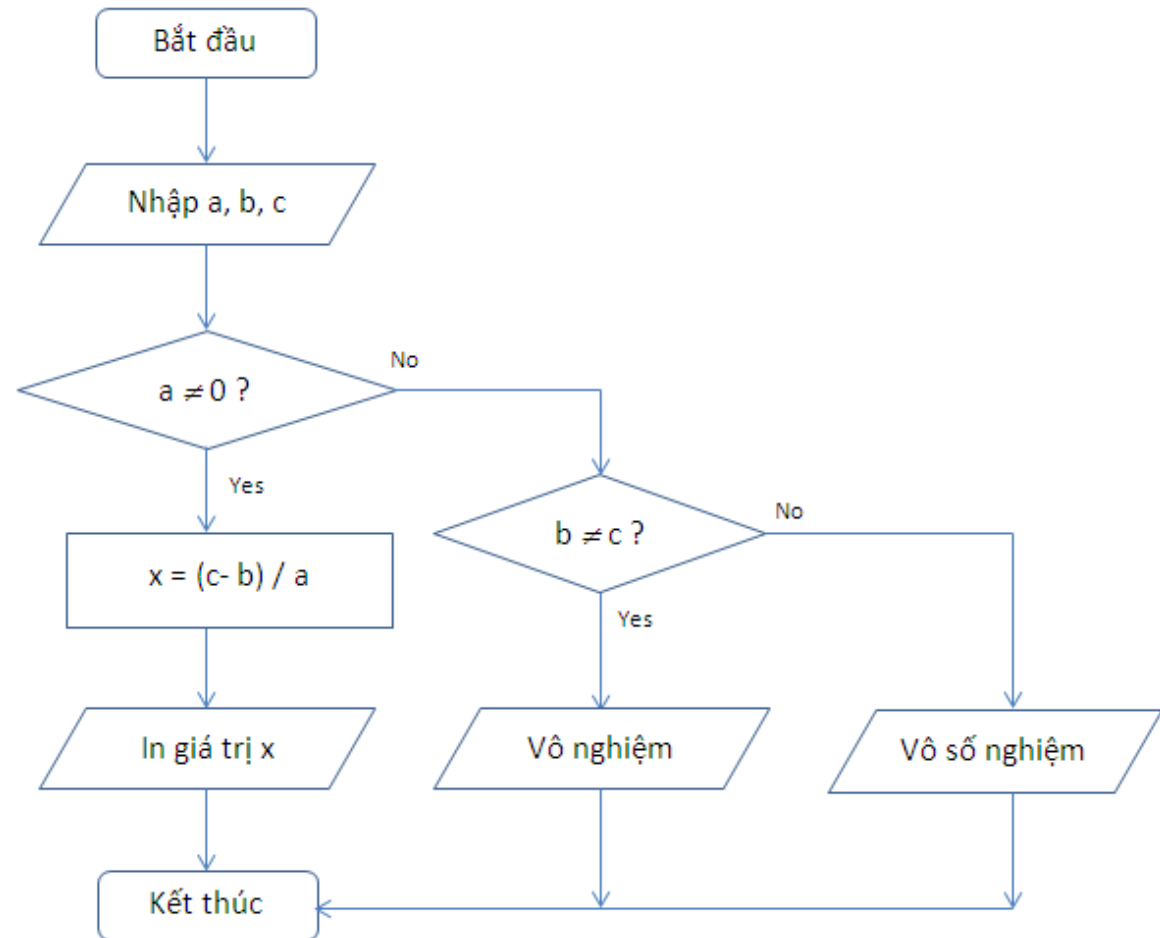


Cấu trúc phân nhánh

Yêu cầu:

- Giải phương trình

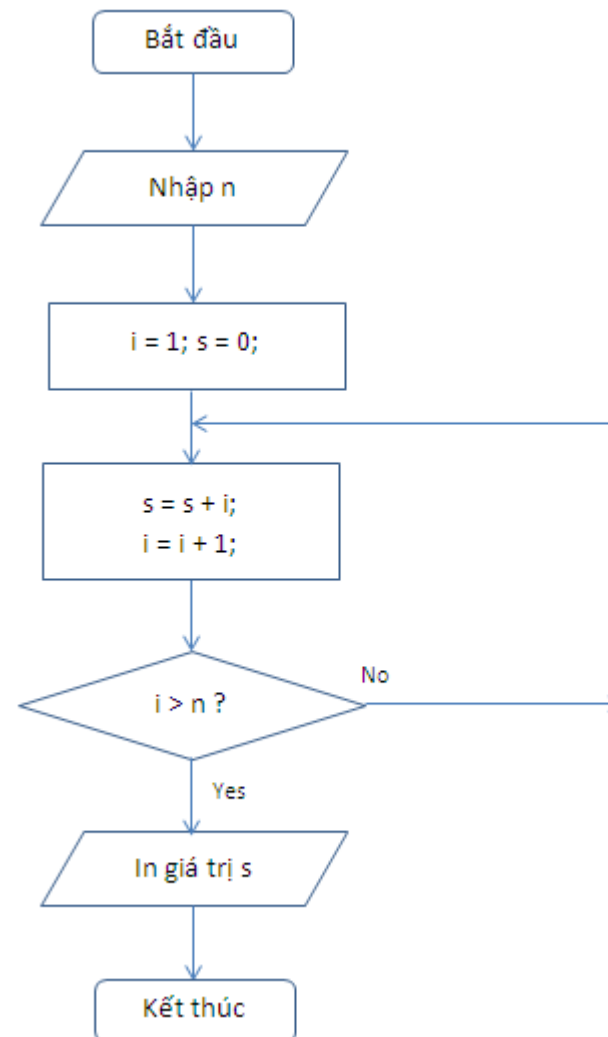
$$P(x): aX + b = c$$



Lặp, xử lý theo chu trình

Yêu cầu:

- Tính tổng các số tự nhiên từ 1 đến n
(n được nhập từ bàn phím)



Ngôn ngữ lập trình

*Nhắc lại kiến thức từ môn "**Cơ sở lập trình**"*

Ngôn ngữ lập trình

Ngôn ngữ lập trình (*Programming language*) là một dạng ngôn ngữ được thiết kế và chuẩn hóa (*So với ngôn ngữ tự nhiên*) để truyền các chỉ thị cho máy tính (*hoặc các thiết bị khác có bộ xử lý: Smartphone, Tablet, Smart TV, ...*).

Ngôn ngữ lập trình thường được dùng để tạo ra các chương trình nhằm phục vụ cho việc điều khiển máy tính hoặc mô tả các thuật toán xử lý của chương trình máy tính.

Ngôn ngữ lập trình

Ngôn ngữ lập trình thường được chia làm 3 dạng

- ❖ **Ngôn ngữ máy** (*Machine language*)
- ❖ **Hợp ngữ** (*Assembly language*)
- ❖ **Ngôn ngữ lập trình cấp cao** (*Higher-Level language*)

Chương trình dịch

Do máy tính (*Và các thiết bị cho phép lập trình*) chỉ có thể hiểu được ngôn ngữ máy, cho nên một chương trình sau khi đã được lập trình (*Viết bằng ngôn ngữ cấp cao, hợp ngữ*) cần phải chuyển đổi thành ngôn ngữ máy thì mới có thể thi hành được.

Những công cụ làm nhiệm vụ chuyển đổi cho mục đích này thường được gọi là ***chương trình dịch***.

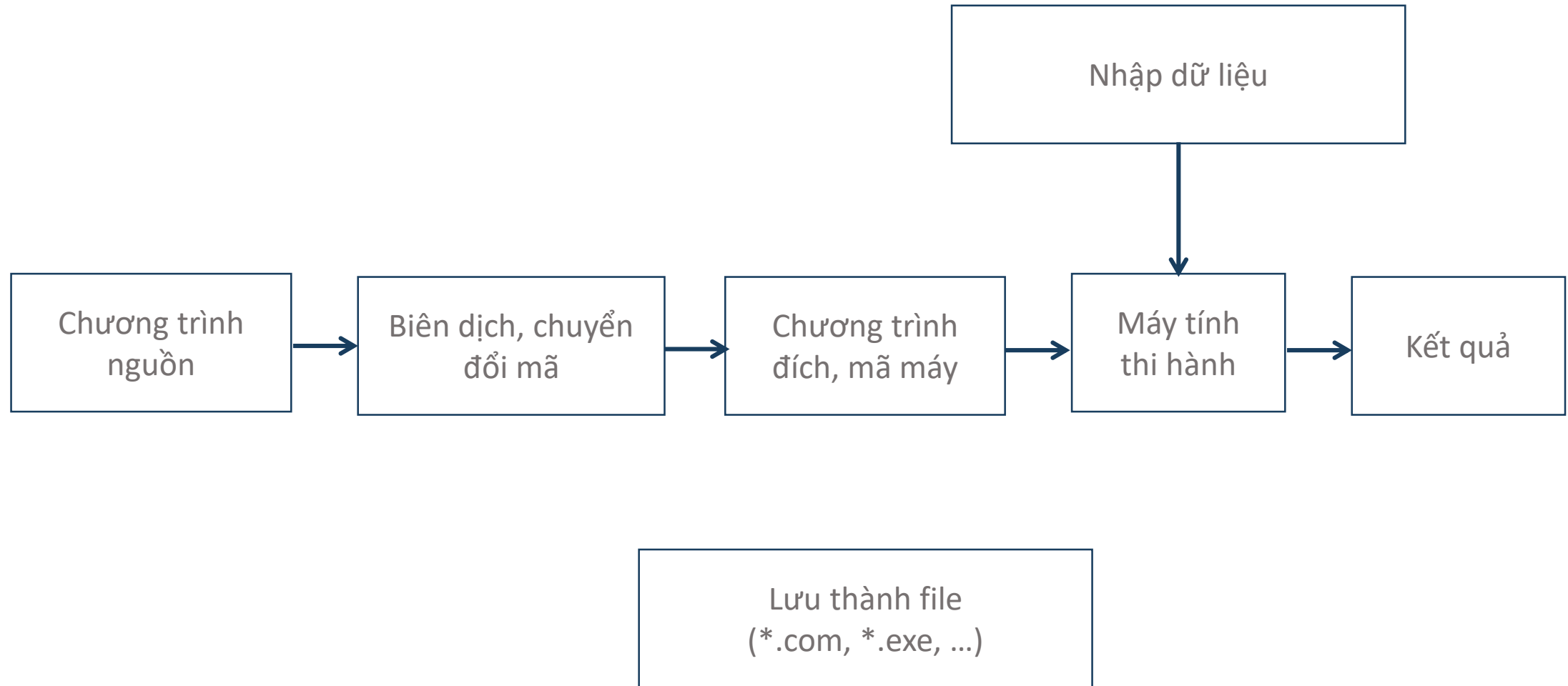
Chương trình dịch thường được phân biệt ở một trong hai dạng:

- ❖ **Trình biên dịch** (*Compiler*)
- ❖ **Trình thông dịch** (*Interpreter*)

Trình biên dịch - Compiler

- Làm nhiệm vụ chuyển đổi một chương trình đã được viết bằng một ngôn ngữ lập trình nào đó (*Còn gọi là chương trình nguồn*) thành ngôn ngữ máy (*Chương trình đích*).
- Quá trình chuyển đổi từ chương trình nguồn thành chương trình đích thường được gọi là **thời gian dịch** (*Compile-time*) và thời gian thực thi chương trình sau khi đã biên dịch thành công được gọi là **thời gian thực thi** (*Run-time*)

Cơ chế biên dịch



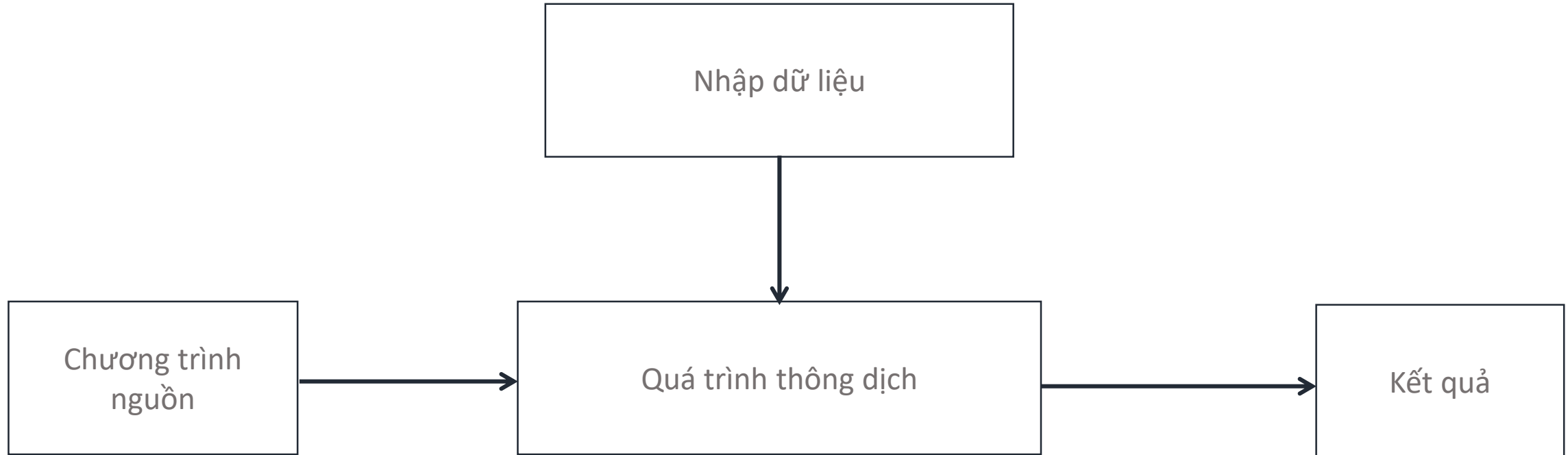
Trình thông dịch - Interpreter

Những phần mềm có khả năng đọc và chuyển đổi mã nguồn của một chương trình đã được viết bằng ngôn ngữ lập trình ra mã máy để ra lệnh cho máy tính thi hành được gọi là trình thông dịch.

Khác với trình biên dịch, trình thông dịch sẽ dịch từng câu lệnh từ chương trình nguồn theo yêu cầu thực thi.

Như vậy, thời gian dịch diễn ra đồng thời với thời gian thực thi chương trình, quá trình này được gọi là ***Thông dịch***

Cơ chế thông dịch



So sánh Compiler vs Interpreter

| Tiêu chí | Trình biên dịch | Trình thông dịch |
|------------------------|--|--|
| Đầu vào | Toàn bộ trường trình | Chỉ một dòng code |
| Đầu ra | Mã đối tượng trung gian | Không tạo ra bất kì mã đối tượng trung gian nào |
| Cơ chế hoạt động | Việc biên dịch sẽ phải hoàn thành công việc trước khi thực thi | Việc biên dịch và thực thi sẽ là đồng thời |
| Tốc độ | Nhanh hơn | Chậm hơn |
| Bộ nhớ | Yêu cầu bộ nhớ nhiều hơn do việc tạo mã đối tượng | Nó đòi hỏi ít bộ nhớ hơn vì nó không tạo mã đối tượng trung gian |
| Errors | Hiển thị tất cả các lỗi sau khi biên dịch, tất cả cùng một lúc | Hiển thị lỗi của từng dòng một |
| Phát hiện error | Rất khó khăn | Tương đối dễ |
| Các ngôn ngữ lập trình | C, C++, C#, Scala, typescript | PHP, Perl, Python, Ruby |

Java

Giới thiệu ngôn ngữ Java

- Java là một ngôn ngữ lập trình hướng đối tượng (**OOP** – *Object-Oriented Programming*) cho phép xây dựng các chương trình điều khiển thiết bị dựa trên các thành phần là các lớp (**class**). Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành **bytecode**, bytecode sau đó sẽ được thực thi trong một môi trường (*runtime environment*) độc lập, hoạt động trên hệ điều hành gọi là *máy ảo Java* (*Java Virtual Machine*).

Giới thiệu ngôn ngữ Java

- Java được phát triển bởi James Gosling và các chuyên gia của Sun Microsystems vào năm 1991. Ban đầu ngôn ngữ này được đặt tên là **Oak**, với mục tiêu thiết kế lại và loại bỏ bớt các tính năng “*nguy hiểm*” có trong ngôn ngữ C++,
- Java được tạo ra với tiêu chí “*Viết một lần, chạy ở mọi nơi*” thông qua khẩu hiệu “*Write Once, Run Anywhere*” (**WORA**). Môi trường thực thi của Sun Microsystems (*Máy ảo Java*) được thiết kế để chạy trên hầu hết các hệ điều hành nổi tiếng như: Sun Solaris, Linux, Mac OS, FreeBSD & Windows.
- Vào năm 1994, Java chính thức được phát hành cho đến nay, Java trở nên nổi tiếng là một ngôn ngữ lập trình được sử dụng trong cả lĩnh vực phát triển phần mềm và nghiên cứu, cũng như giảng dạy cho sinh viên ngành CNTT
- Giai đoạn 2009-2010, Oracle mua lại Sun Microsystems, Java đã chính thức trở thành một trong những sản phẩm chủ lực của Oracle Corporation

Mục tiêu thiết kế Java

- **Safety**, *Người dùng hoàn toàn yên tâm khi tải các ứng dụng Java thông qua WWW*
- **Portable**, *Có thể phát triển sản phẩm trên 1 hệ thống, nhưng hoàn toàn có thể chạy trên các hệ thống khác*
- **Distributed**, *Người dùng thông thường có thể khai thác & sử dụng các dịch vụ mạng khác nhau*
- **Scalable**, *Để phát triển các ứng dụng thực tế bằng cách mở rộng thư viện các lớp đã xây dựng trước đó*

Một số đặc điểm

- Java là ngôn ngữ hoàn toàn hướng đối tượng. Tất cả các chương trình đều gọi & sử dụng đối tượng trong quá trình hoạt động.
- Các chương trình Java luôn được biên dịch thành **bytecode**, độc lập so với hệ điều hành.
- Các **bytecode** của Java thi hành trong Máy ảo Java (**JVM**)
- Các chương trình Java có thể được xây dựng thành các “**Package**” để tạo nên các thư viện mã nguồn, cung cấp nhiều giải pháp đa dạng & linh hoạt cho các nhu cầu thực tế của ứng dụng.

JVM - Java Virtual Machine

- Máy ảo Java cung cấp môi trường thực thi các bytecode, mã thực thi được biên dịch từ các Java classes
- Các nhiệm vụ chính của JVM bao gồm:
 - ✓ Loads code
 - ✓ Verifies code
 - ✓ Executes code
 - ✓ Provides runtime environment
- Các thành phần JVM cung cấp, bao gồm:
 - ✓ Memory area
 - ✓ Class file format
 - ✓ Register set
 - ✓ Garbage-collected heap
 - ✓ Fatal error reporting etc.

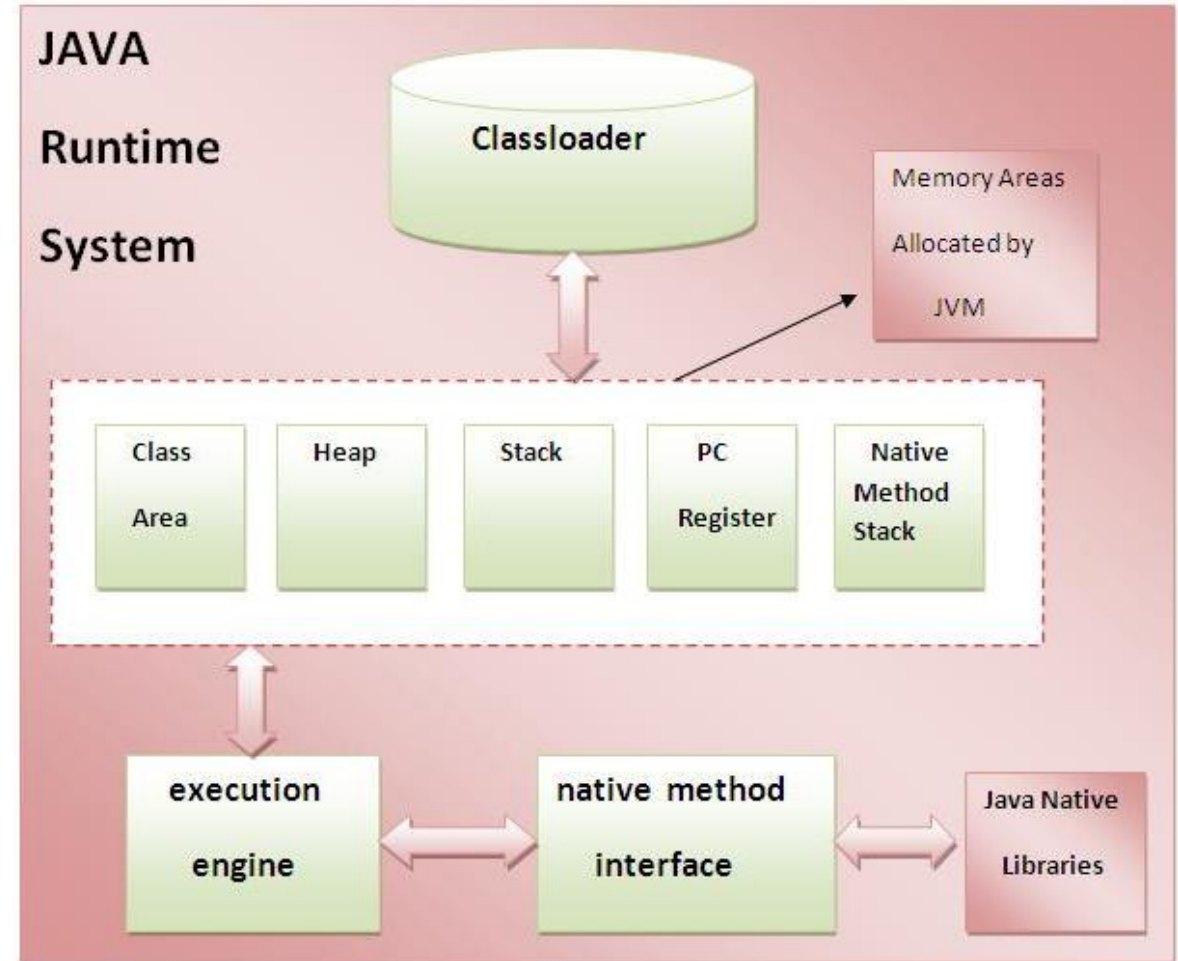


Image link: <https://www.javatpoint.com/images/jvminernal.JPG>

JRE - Java Runtime Environment

- JRE cung cấp môi trường thực thi, bao gồm cả việc triển khai JVM. Các thành phần mà JRE cung cấp là tập các thư viện mã nguồn đã được biên dịch, cùng các tài nguyên cần thiết để cho JVM sử dụng trong quá trình hoạt động

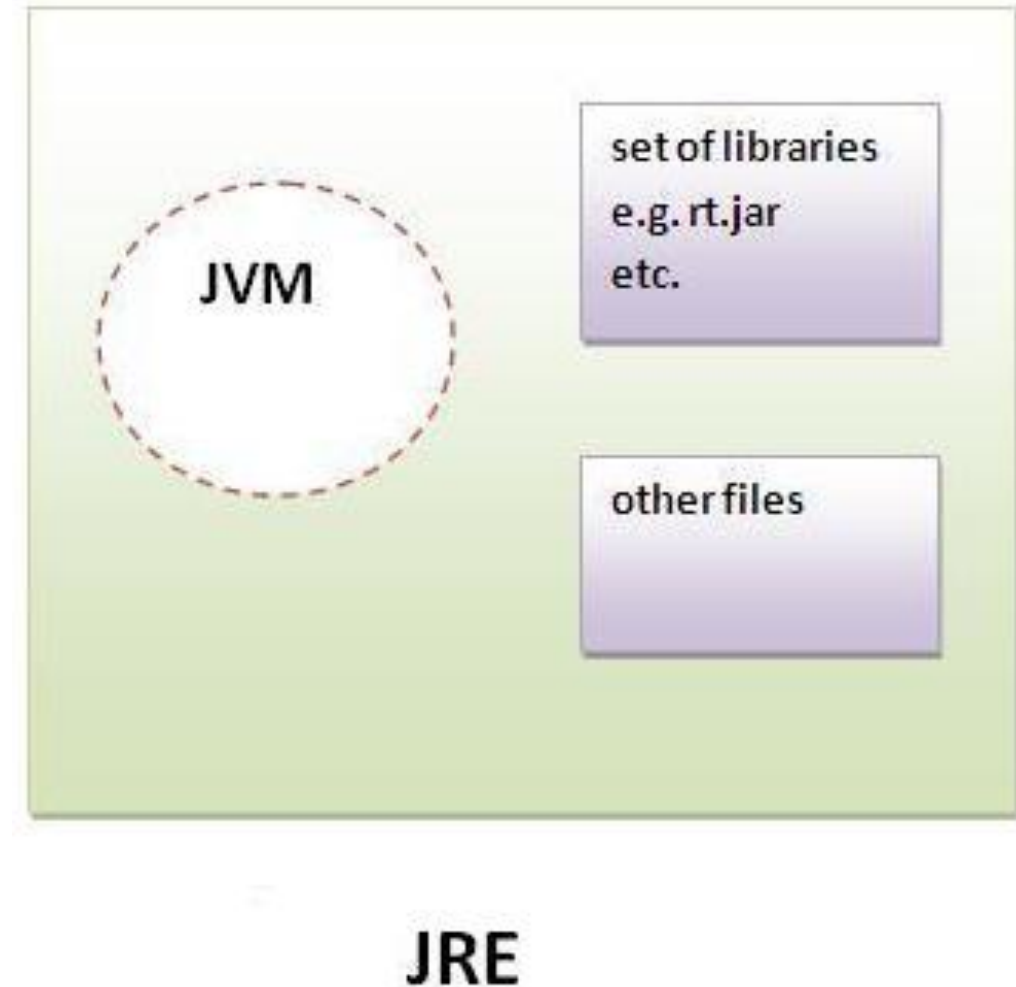


Image link: <https://www.javatpoint.com/images/jre2.JPG>

JDK – Java Development Kit

- Bộ phát triển phần mềm Java, JDK bao gồm JRE và các công cụ cần thiết khác phục vụ cho mục tiêu phát triển ứng dụng bằng ngôn ngữ Java

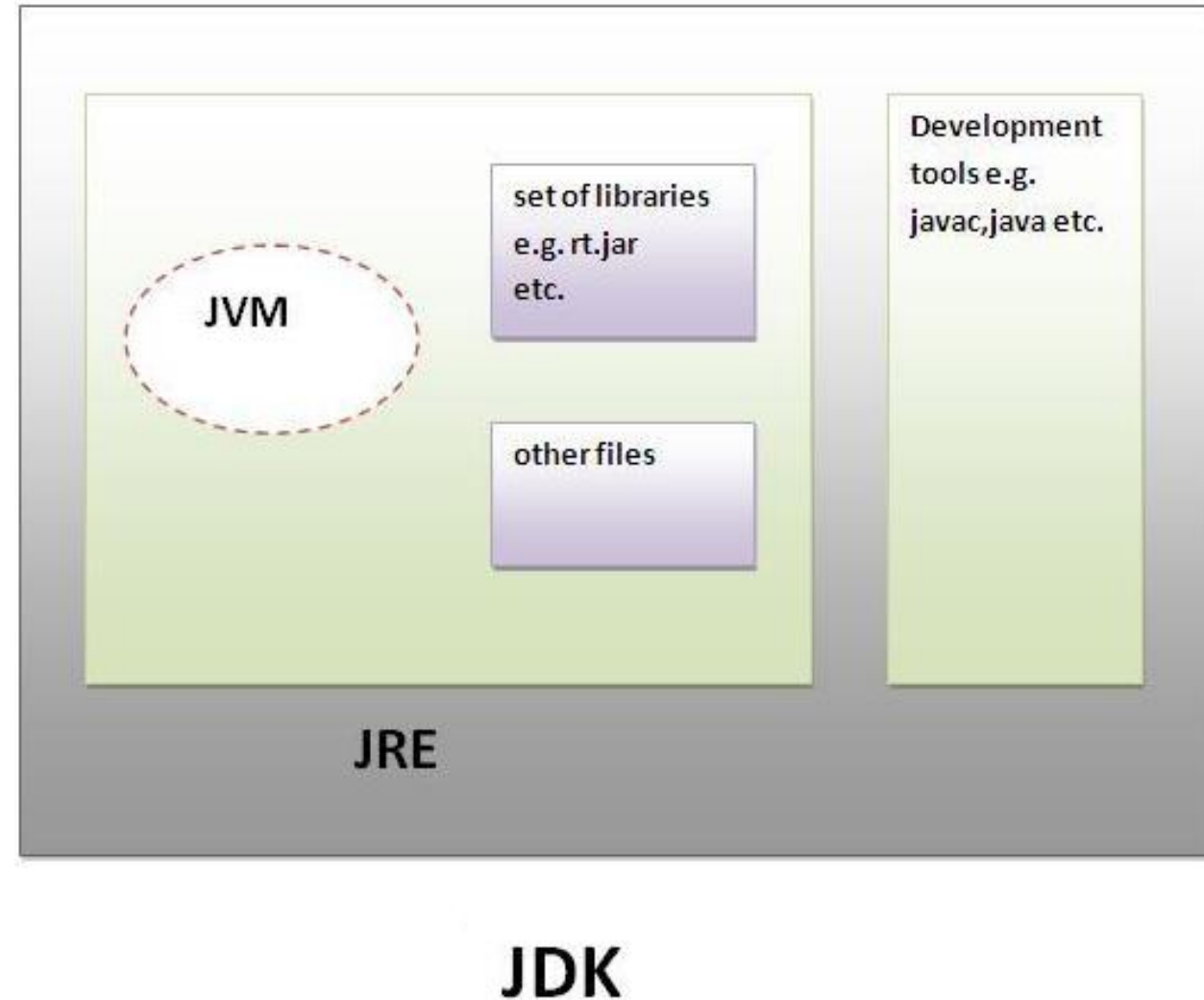


Image link: <https://www.javatpoint.com/images/jdk2.JPG>

Các dạng sản phẩm

■ Console applications

- *Chỉ đơn thuần là xử lý & giao tiếp với người dùng dựa trên text*

■ Applets

- *Hoạt động trên các trang web, không còn phổ biến từ những năm 2005, hạn chế về bảo mật & tốn tài nguyên khi trình duyệt nạp trang web*

■ Frame-Based Applications

- *Desktop application*

■ Servlets

- *Các chương trình chạy trên Web servers*

Một số quy ước

- Mỗi Java class nên lưu trữ trong một tập tin riêng biệt có phần mở rộng là “**.java**”
- Tên của tập tin “**.java**” nên trùng tên với class name đã được định nghĩa trong mã nguồn
- Tất cả các tập tin mã nguồn của một chương trình nên được quản lý bởi các Directory / Folder có cấu trúc “phù hợp”

IDE

(Integrated Development Environment)

Setup JDK & Eclipse

- **JDK 8u161**

(<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>)

- **Eclipse IDE**

(<http://www.eclipse.org/downloads/>)



- **NetBeans IDE**

(<https://netbeans.org/downloads/>)



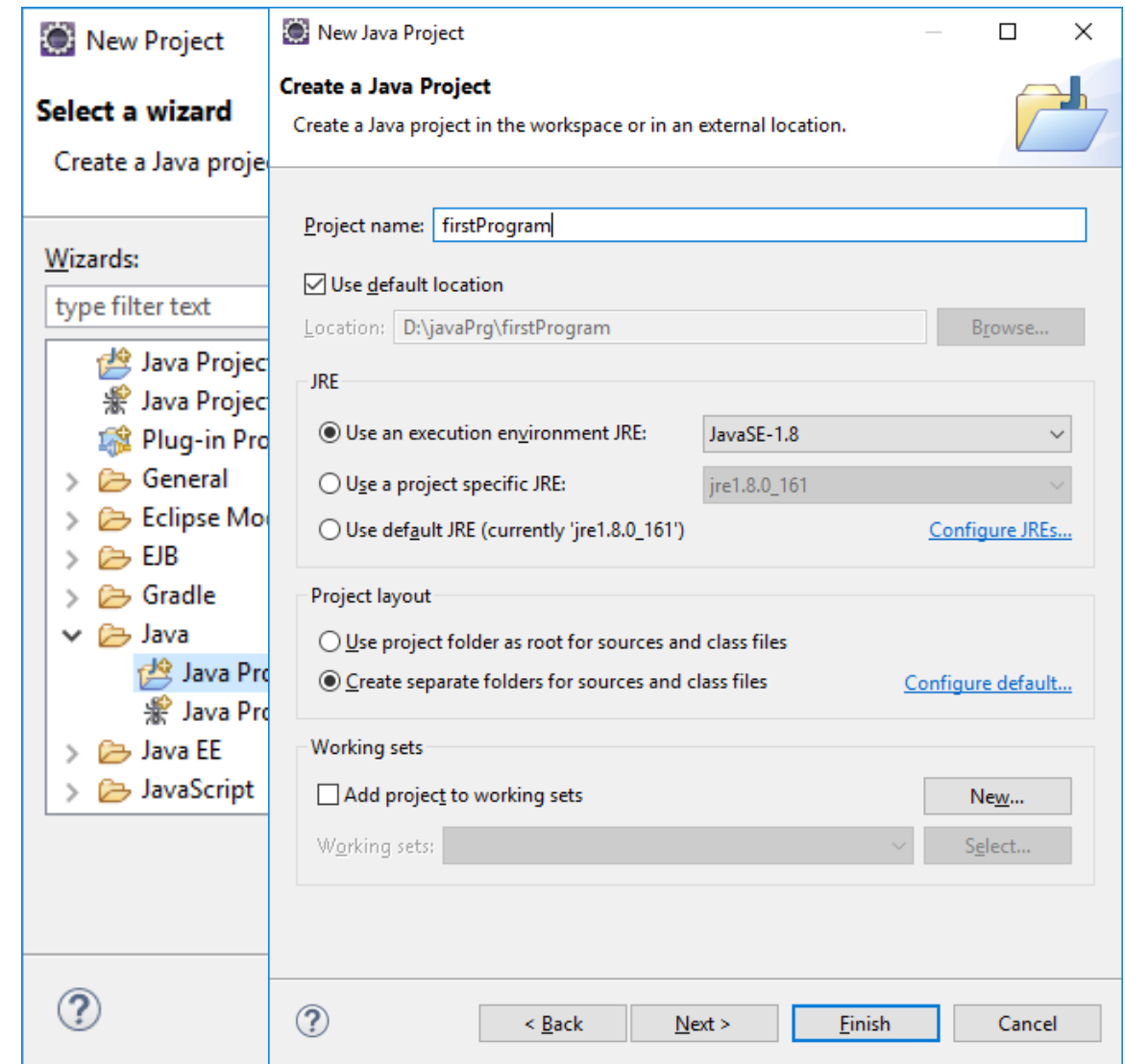
Một số thuật ngữ

- **J2SE** (*Java 2 Standard Edition*) là một nền tảng thực thi (*bao gồm cả phát triển ứng dụng và triển khai*) cho các ứng dụng Java. **J2SE** cung cấp các API, các kiến trúc chuẩn, các thư viện lớp và các công cụ cần thiết để xây các ứng dụng Java. **J2SE** vẫn được xem là nền tảng thiên về phát triển các sản phẩm chạy trên máy tính để bàn. **J2SE** gồm 2 thành phần chính: JRE & JDK
(https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html#A1096936)
 - **JRE** – Java 2 Runtime Environment
 - **JDK** – Java 2 Software Development Kit
- **J2EE** (*Java 2 Platform, Enterprise Edition*, hay *Java EE*) là một nền tảng cho phép lập trình để phát triển ứng dụng phân tán trên kiến trúc đa tầng (nLayer), chủ yếu dựa vào các module chạy trên các máy chủ ứng dụng.
- **J2ME** được phát triển từ kiến trúc Java Card, Embedded Java và Personal Java của phiên bản Java 1.1. Đến sự ra đời của Java 2 thì Sun quyết định thay thế Personal Java và được gọi với tên mới là Java 2 Micro Edition, hay viết tắt là **J2ME**. Đúng với tên gọi, J2ME là nền tảng cho các thiết bị có tính chất nhỏ gọn. J2ME is a computing platform for development and deployment of portable code for embedded and mobile devices (*micro-controllers, sensors, gateways, mobile phones, personal digital assistants, TV set-top boxes, printers*)

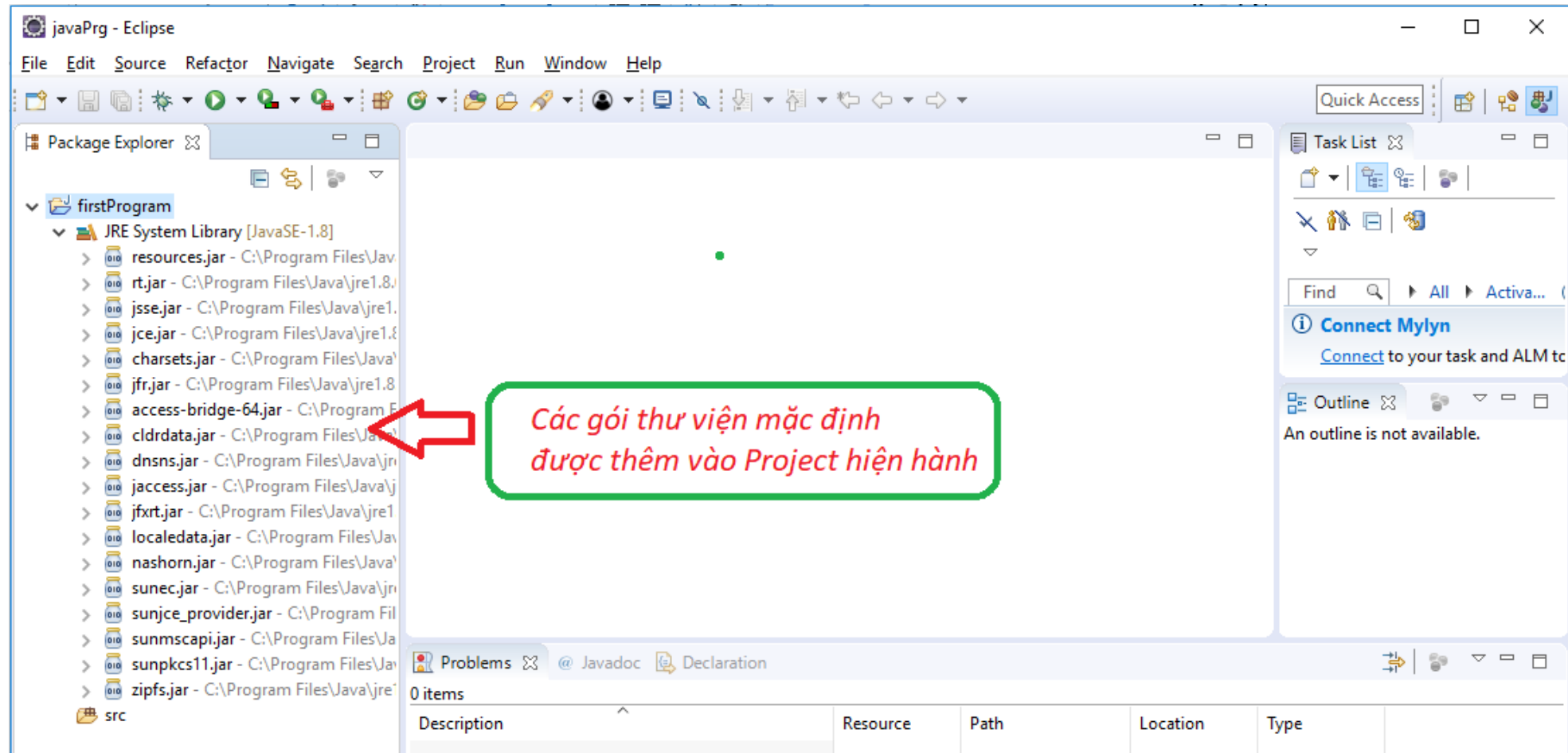
First program

■ Tạo Java project

[Menu] File \ New -> Project

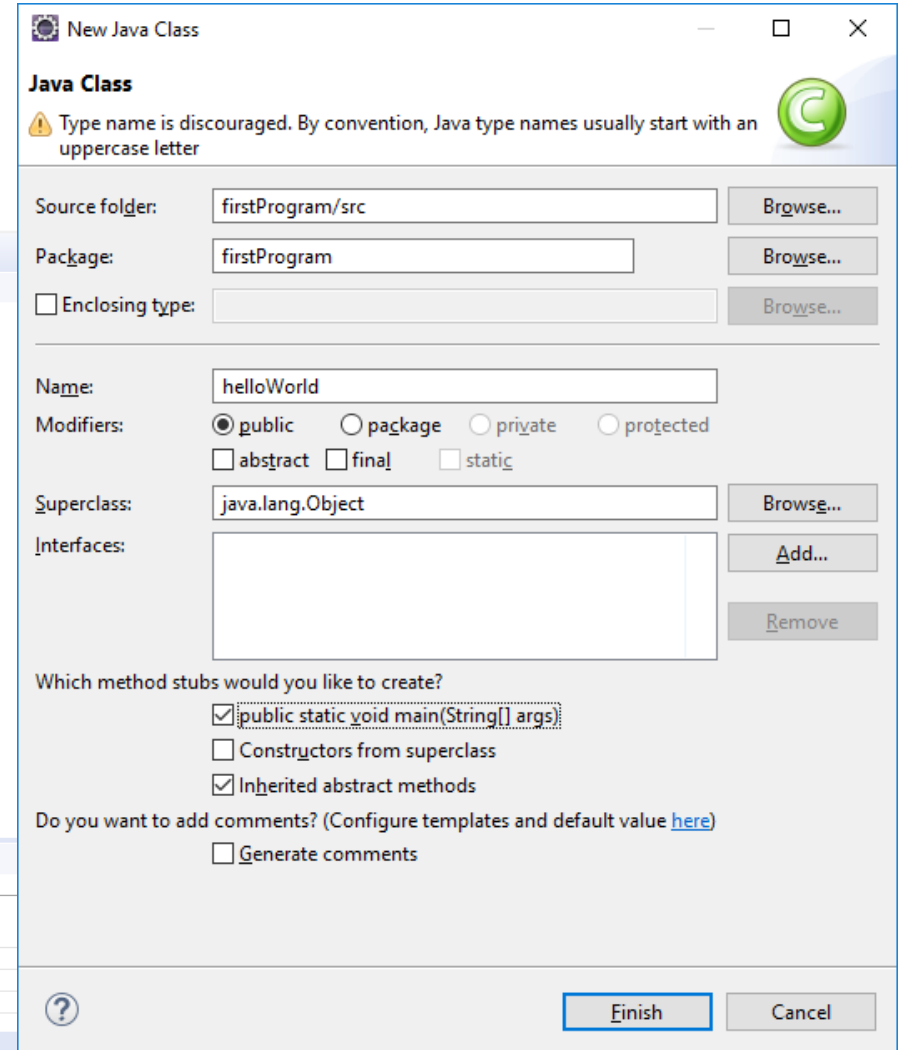
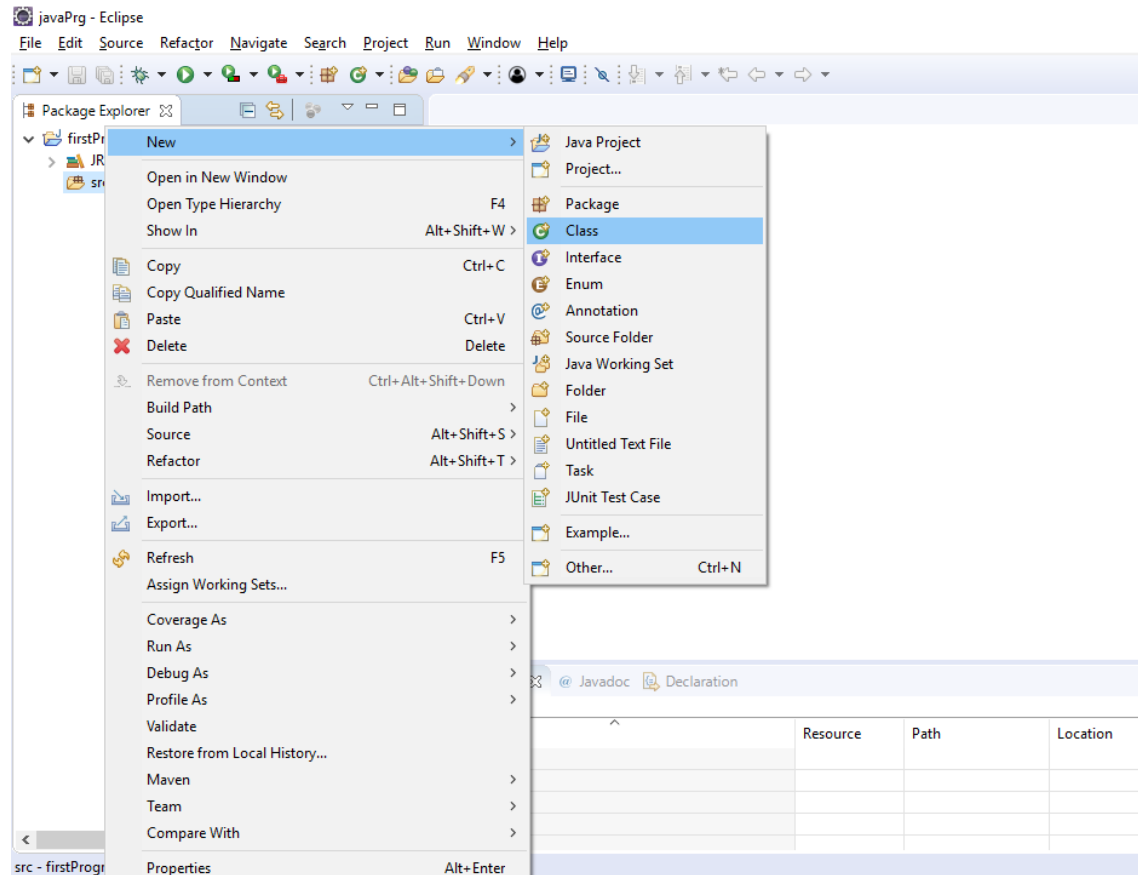


■ Package explorer & các thư viện mặc định

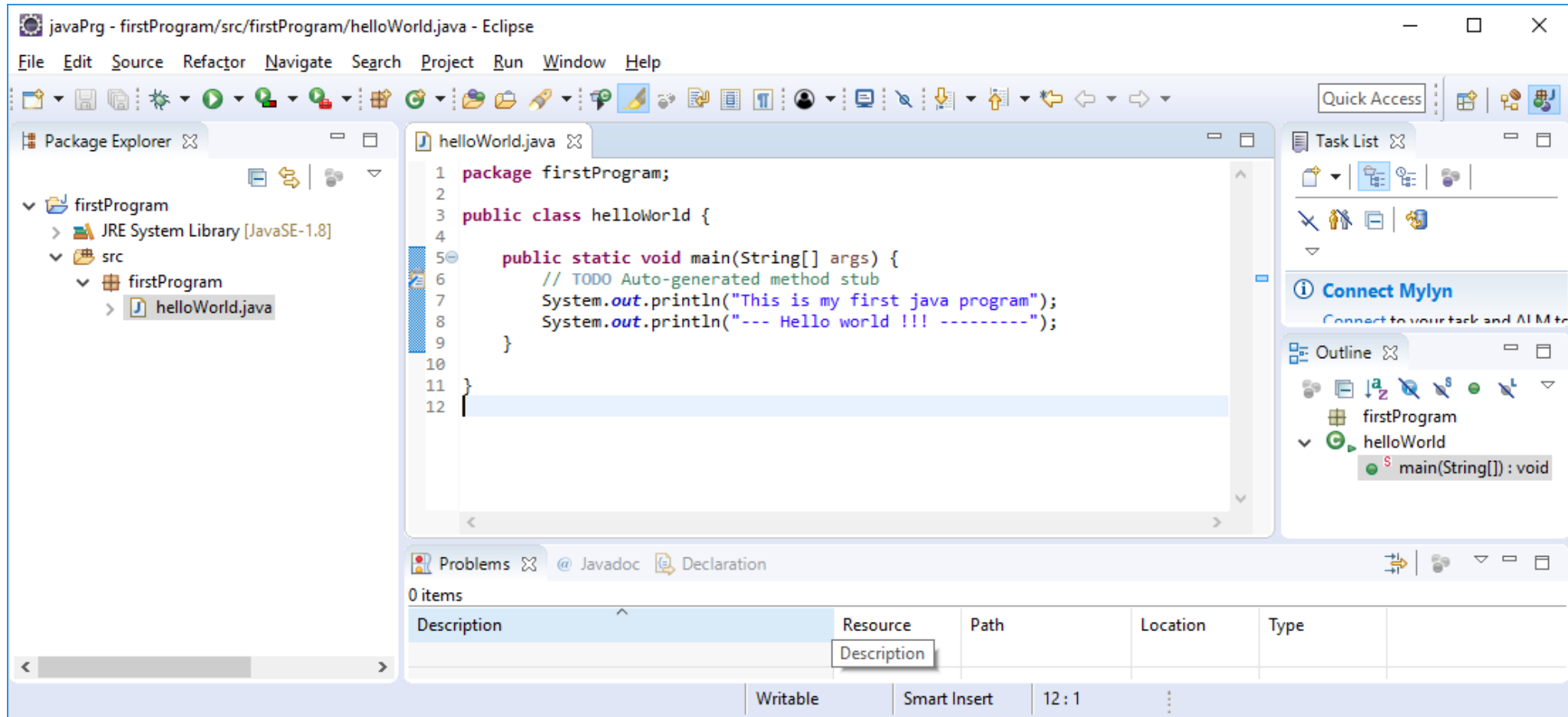


■ Tạo class đầu tiên cho chương trình

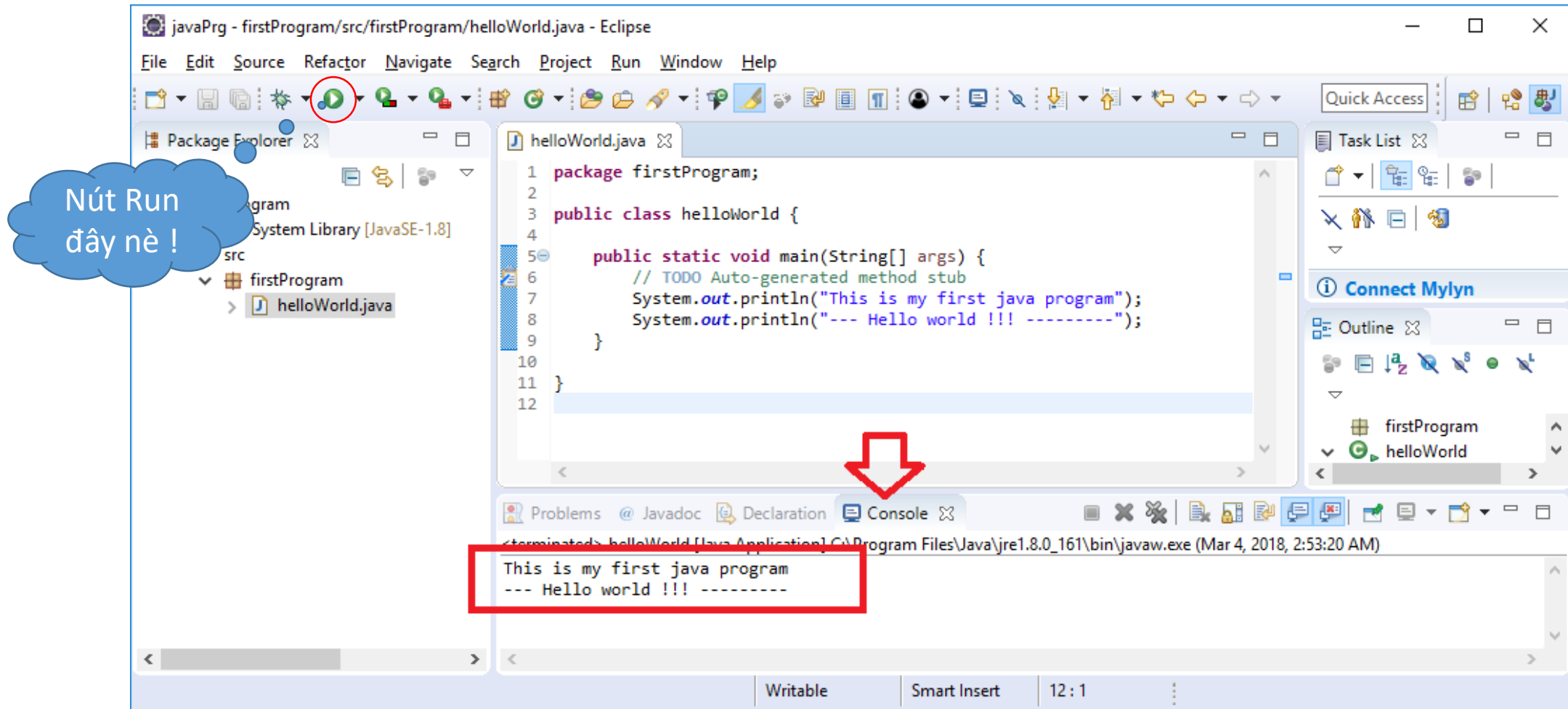
right mouse -> thư mục src, chọn new -> class



▪ Xuất dữ liệu với **System.out.println**(“Chuỗi dữ liệu”)



- Thi hành chương trình & quan sát thông tin trên vùng cửa sổ **Console**



Nhớ gì ?!!!

- Các khái niệm thuật toán, lưu đồ, ngôn ngữ lập trình
- Những khái niệm chính liên quan đến ngôn ngữ Java
- Khởi động Eclipse IDE & tạo Java Project, Tạo class
- **System.out.print** hoặc **System.out.println** là các phương thức cho phép xuất dữ liệu ra màn hình của Java
- Hàm main của 1 class sẽ tự động được gọi thi hành và hàm này bắt buộc phải có khai báo “**public static void**”

Tài liệu tham khảo

- Jose M. Garrido, “**Object-Oriented Programming: From Problem Solving to Java**”
- Paul Deitel, Harvey Deitel, “**Java : How to program**”, 9th edition, 2012
- Oracle, “**The Java™ Tutorials**”,
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>,
6:20PM, 18/01/2018
- Java tutorial, <https://www.javatpoint.com/java-tutorial> , 6:20PM,
18/01/2018