Môn: Lập trình Hướng đối tượng (Object Oriented Programming)

Chương 3. Giới thiệu Java

Nội dung

3.1. Lịch sử phát triển của Java
3.2. Đặc trưng của Java
3.3. Tổng quan lập trình Java
    3.3.1. Kiểu dữ liệu cơ bản
    3.3.2. Hằng, biến
    3.3.3. Toán tử, biểu thức
    3.3.4. Các cấu trúc lệnh trên Java (cấu trúc điều khiển, lặp)
3.4. Kiến trúc chương trình xây dựng trên Java
3.4. Case Study I (simple programs & language)

3.1. Lịch sử phát triển của Java

- 1991: được Sun Microsystems phát triển nhằm mục đích viết phần mềm điều khiển (phần mềm nhúng) cho các sản phẩm gia dụng
  - lúc đầu được đặt tên là Oak
- 1995: được phổ cập với sự phát triển mạnh mẽ của Internet thị trường phần mềm nhúng không phát triển mạnh
  - WWW bùng nổ (1993~)
- Hiện nay, được chấp nhận rộng rãi với tư cách là một ngôn ngữ (công nghệ) đa dụng
  - khả chuyển, an toàn
  - hướngđốitượng, hướng thành phần

3.1. Lịch sử phát triển của Java (tt)

- Java là một công nghệ
- Java bao gồm
  - Ngôn ngữ lập trình
  - Môi trường phát triển
  - Môi trường thực thi và triển khai

3.1. Lịch sử phát triển của Java (tt)

Mục tiêu của Java

- Ngôn ngữ dễ dùng
  - Khắc phục nhiều nhược điểm của các ngôn ngữ trước đó
  - Hướng đối tượng
  - Rõ ràng
- Môi trường thông dịch
  - Tăng tính khả chuyển
  - An toàn
- Cho phép chạy nhiều tiến trình (threads)
- Nạp các lớp (classes) động vào thời điểm cần thiết từ nhiều nguồn khác nhau
  - Cho phép thay đổi động phần mềm trong khi hoạt động
- Tăng độ an toàn

3.1. Lịch sử phát triển của Java (tt)

Biên dịch và thông dịch

- Chương trình nguồn được biên dịch sang mã đích (bytecode)
- Mã đích (bytecode) được thực thi trong môi trường thông dịch (máy ảo)

Các dạng ứng dụng củaJava

- Desktop applications - J2SE
  - Java Applications: ứng dụng Java thông thường trên desktop
  - Java Applets: ứng dụng nhúng hoạt động trong trình duyệt web
- Server applications - J2EE
  - JSP và Servlets
- Mobile (embedded) applications – J2ME

3.2. Đặc trưng của Java

- JVM (Java Virtual Machine) – máy ảo Java
- Cơ chế giải phóng bộ nhớ tự động
- Bảo mật chương trình

3.2. Đặc trưng của Java (tt)

JVM (Java Virtual Machine) – máy ảo Java

- Máy ảo phụ thuộc vào platform (phần cứng, OS)
- Cung cấp môi trường thực thi cho chương trình Java (độc lập với platform)
- Máy ảo đảm bảo an toàn cho hệ thống
- Máy ảo thông thường được cung cấp dưới dạng phần mềm
  - JRE - Java Runtime Environment
- Java platform: JVM + APIs

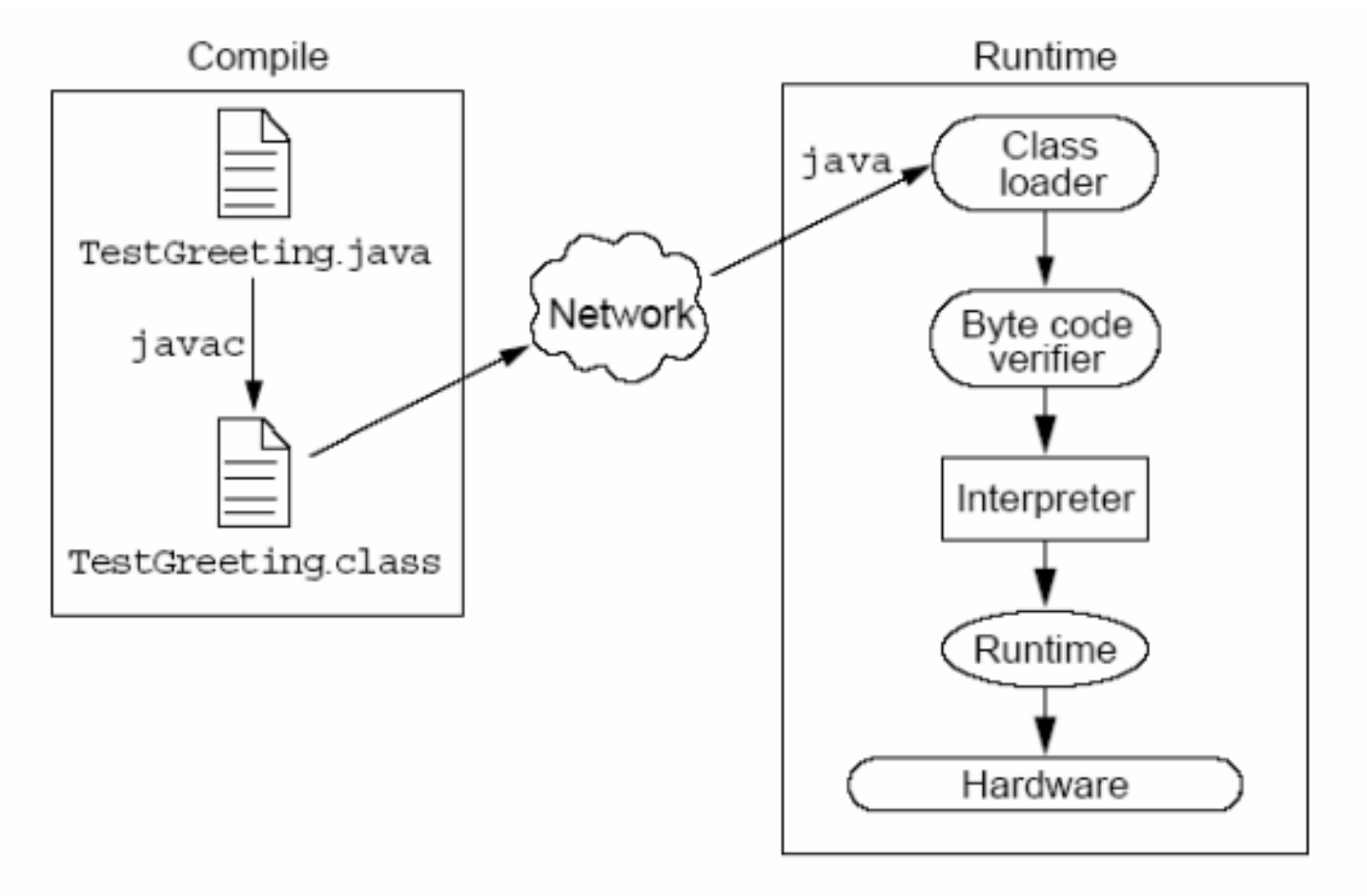
3.2. Đặc trưng của Java (tt)

Cơ chế giải phóng bộ nhớ tự động

- Java cung cấp một tiến trình mức hệ thống để theo dõi việc cấp phát bộ nhớ
- Garbage Collection
  - Đánh dấu và giải phóng các vùng nhớ không còn được sử dụng
  - Được tiến hành tự động
  - Cơ chế hoạt động phụ thuộc vào các phiên bản máy ảo

3.2. Đặc trưng của Java (tt)

Bảo mật chương trình – chống sao chép



3.2. Đặc trưng của Java (tt)

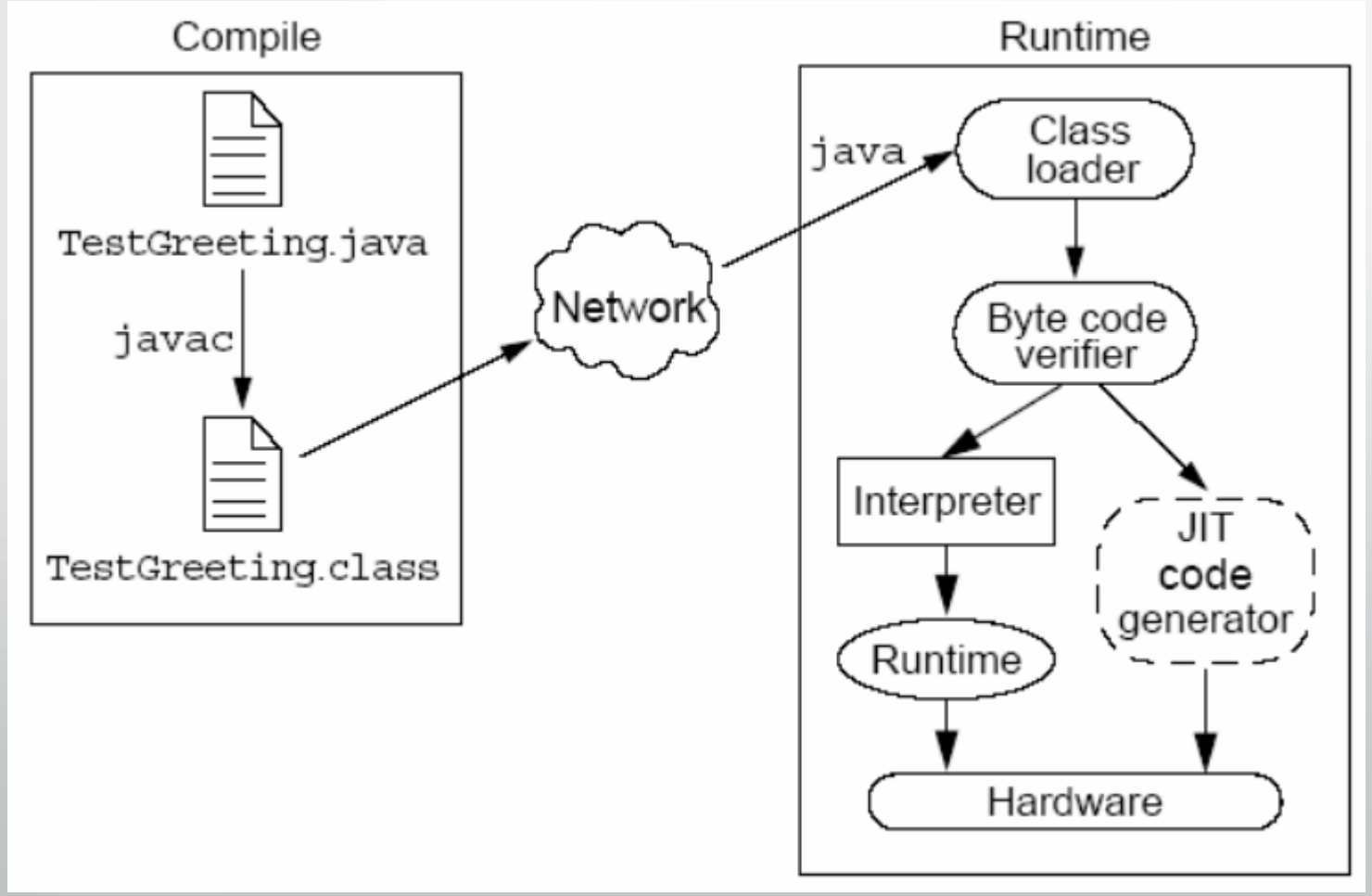
JDK Java Development Kit

- Môi trường phát triển và thực thi do Sun Microsystems cung cấp (http://oracle.com/java)
- Phiên bản hiện tại J2SDK 8.0 (1.8)
- Bao gồm
  - javac Chương trình dịch chuyển mã nguồn sang bytecode
  - java Bộ thông dịch: Thực thi java application
  - appletviewer Bộ thông dịch: Thực thi java applet mà không cần sử dụng trình duyệt như Nestcape, hay IE, v.v.
  - javadoc Bộ tạo tài liệu dạng HTML từ mã nguồn và chú thích
  - jdb Bộ gỡ lỗi (java debuger)
  - javap Trình dịch ngược bytecode

3.2. Đặc trưng của Java (tt)

Công nghệ JIT

Just-In-Time Code Generator



3.2. Đặc trưng của Java (tt)

- Java Applications
  - Chương trình ứng dụng hoàn chỉnh
  - Giao diện dòng lệnh hoặc đồ họa
  - Được bắt đầu bởi phương thức (hàm) main() là phương thức public static
- Java Applets
  - Được nhúng trong một ứng dụng khác (web browser)
  - Có giao diện hạn chế (đồ họa)
  - Không truy cập được tài nguyên của client
  - Code nhúng vào trang Web: Welcome.html:

```
<html>
<applet code = ”Welcome.class” width = ”300”
height = ”45”></applet>
</html>
```

3.3. Tổng quan lập trình Java

3.3.1. Kiểu dữ liệu cơ bản
3.3.2. Hằng, biến
3.3.3. Toán tử, Biểu thức
3.3.4. Các cấu trúc lệnh trên Java (cấu trúc điều khiển, lặp)

3.3.1. Kiểu dữ liệu cơ bản

Có 8 kiểu dữ liệu cơ bản trong Java.

- 4 kiểu biểu diễn số nguyên:
  - `byte`, `short`, `int`, `long`
- 2 kiểu biểu diễn số thực:
  - `float`, `double`
- 1 biểu diễn các ký tự:
  - `char`
- Và 1 biểu diễn cho giá trị luận lý *(true, false)*:
  - `boolean`

3.3.2. Hằng, biến

Từ khóa

- Từ khóa cho các kiểu dữ liệu cơ bản : byte, short, int, long, float, double, char, boolean
- Từ khóa cho phát biểu lặp: do, while, for, break, continue
- Từ khóa cho phát biểu rẽ nhánh: if, else, switch, case, default, break
- Từ khóa đặc tả đặc tính một method: private, public, protected, final, static, abstract, synchronized, volatile..
- Literal value: true, false, null
- Từ khóa liên quan đến method: return, void
- Từ khoá liên quan đến package: package, import
- Từ khóa cho việc quản lý lỗi: try, catch. finally, throw, throws
- Từ khóa liên quan đến đối tượng: new, extends, implements, class, instanceof, this, super

3.3.2. Hằng, biến (tt)

Cách đặt tên

- Bắt đầu bằng ký tự, ký tự gạch dưới (underscore ‘_’) hay ký tự ‘$’
- Sau đó là các ký tự ký số hay ‘_’, ‘$’ , không dùng các ký tự khác như: khoảng trống, ký hiệu phép toán
- Tên có tính chất phân biệt chữ thường chữ hoa (case-sensitive)

3.3.2. Hằng, biến (tt)

Khai báo và sử dụng các biến

- Biến là một giá trị có thể thay đổi khi chương trình thực thi.
- Khi biến được tạo sẽ xuất hiện một vùng nhớ để lưu trữ giá trị của biến.
- 3 đặc điểm của biến: *tên biến, giá trị khởi tạo, tầm vực (scope)*
- Scope của biến: khối chương trình mà biến có ý nghĩa (tham khảo được)
- Một biến phải được khai báo trước khi sử dụng (tên biến và kiểu dữ liệu). Một biến có thể được khởi tạo giá trị khi khai báo biến.

Kiểu dữ liệu

Tên biến

```
int total;

int count, temp, result;
```

```
int sum = 0;
int base = 32, max = 149;
```

3.3.2. Hằng, biến (tt)

- Hằng số (constant)
  - Một hằng tương tự như biến như giá trị của nó luôn luôn không đổi.
  - Trình biên dịch sẽ phát sinh lỗi nếu ta cố tình thay đổi giá trị của hằng.
  - Trong Java, ta dùng `final` để khai báo hằng.

```
final int MIN_HEIGHT = 69;
```

- Phép gán làm thay đổi giá trị của một biến. Toán tử gán (=).

```
total = 55;
```

- Toán tử gán mở rộng +=, -=, *=, /=, %=

3.3.3. Toán tử, biểu thức (tt)

- *Các toán tử số học:*

Cộng	+
Trừ	-
Nhân	*
Chia	/
Chia lấy số dư	%

- *Toán tử tăng / giảm*
  - Toán tử tăng (++)
  - Toán tử giảm (--)
  - Câu lệnh **count++;** tương đương với **count = count + 1;**

3.3.3. Toán tử, biểu thức (tt)

- Toán tử so sánh (quan hệ)

==	bằng
!=	không bằng
<	nhỏ hơn
>	lớn hơn
<=	nhỏ hơn hoặc bằng
>=	lớn hơn hoặc bằng

- *Toán tử luận lý*

! :	not
&& :	and
|| :	or

3.3.3. Toán tử, biểu thức (tt)

- Toán tử điều kiện

  Cú pháp: ***điều_kiện? biểu_thức1:biểu_thức2***

- Nếu ***điều_kiện*** là true, ***biểu_thức1*** được thực thi; Nếu là false, ***biểu_thức2*** được thực thi.

- Các biểu thức
  - Một biểu thức là một sự kết hợp giữa các toán tử và các toán hạng.
  - Nếu trong biểu thức có chứa số thực thì kết quả trả về số thực.
  - Lưu ý độ ưu tiên của các toán tử trong 1 biểu thức

3.3.3. Toán tử, biểu thức (tt)

- instanceof : toán tử kiểm tra 1 đối tượng có thuộc 1 lớp nào đó → true | false

```
class InstanceOfDemo
{
  public static void main (String args[])
  {
       InstanceOfDemo t = new InstanceofDemo();
       if  ( t instanceof InstanceOfDemo)
         System.out.println(" Yes");
       else
         System.out.println(" NO");
  }
}
```

3.3.4. Các cấu trúc lệnh trên Java

Cấu trúc điều khiển – Rẽ nhánh

Cấu trúc if

```
if (Condition)
{
    Statements;
}
else
{
    Statement;
}
```

Cấu trúc switch

```
switch (Expression)
{ case Cons1: Statements;
    break;
  case Cons2: Statements;
    break;
  . . .
  default : Statements;
 }
```

3.3.4. Các cấu trúc lệnh trên Java

Cấu trúc điều khiển – Rẽ nhánh (tt)

- Thường một lệnh *break* được dùng ở cuối danh sách lệnh của mỗi case. Một cấu trúc switch có thể có một case *default*
- Kết quả của ***`biểu_thức`*** trong `switch` phải là kiểu số nguyên (`byte`, `short`, `int`, `long`) hoặc `char`.
- Không thể là `boolean` hoặc số thực (`float`, `double`)

3.3.4. Các cấu trúc lệnh trên Java (tt)

- Cấu trúc lặp

```
while (condition)
{
      Statements;
}
```

```
do
{
      Statements;
} while (condition);
```

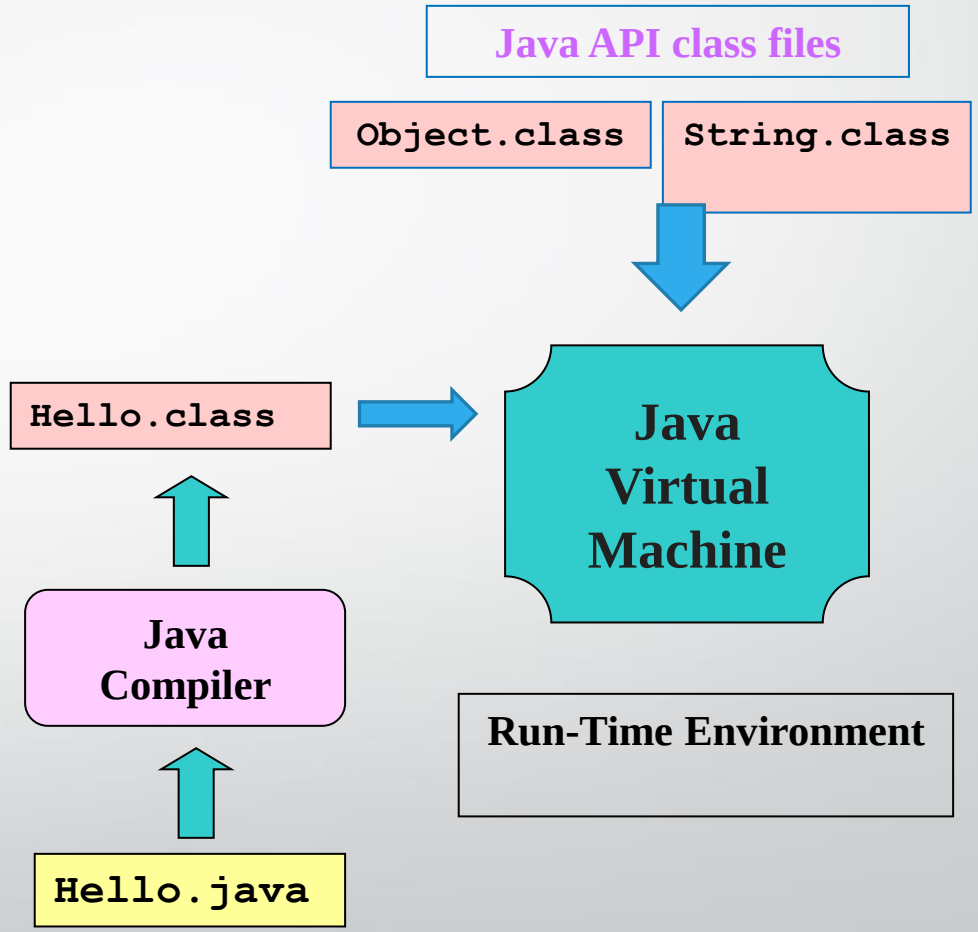
```
for ( varInit ; Condition ; Statements)
{
   Statements1;
}
```

3.4. Kiến trúc chương trình xây dựng trên Java

1. Chương trình nguồn (source code) được viết bằng ngôn ngữ Java

2. Các chương trình được biên dịch thành các file dạng lớp (*. class)

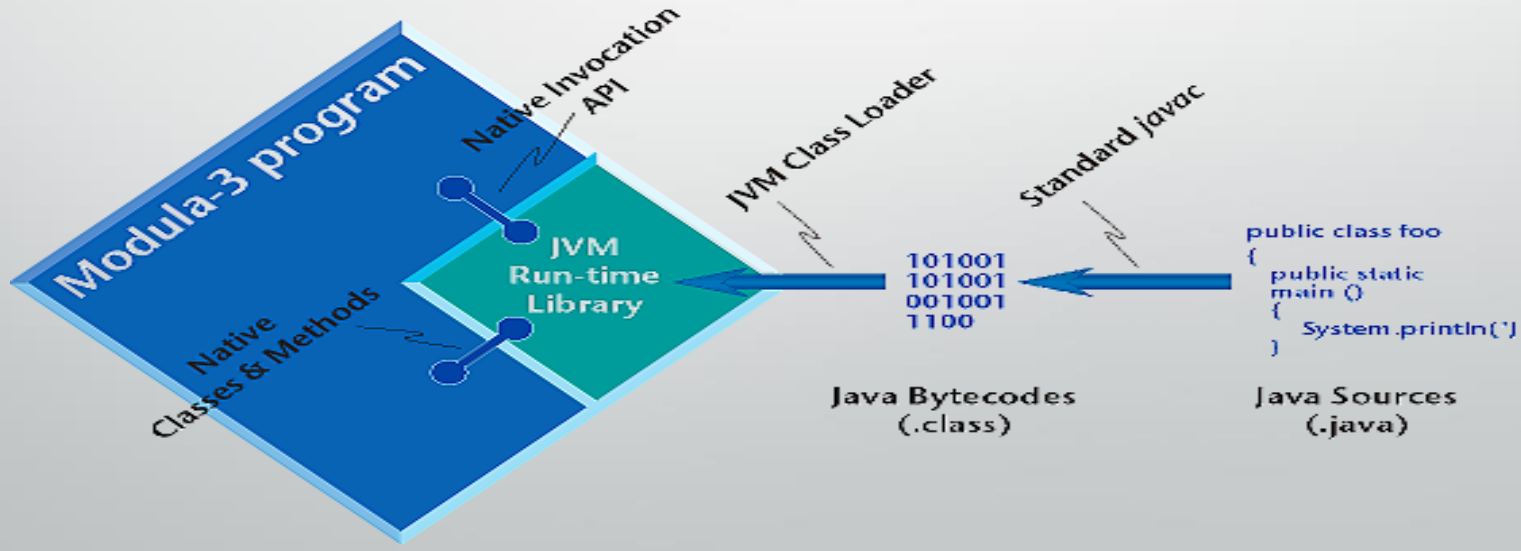
3. Các file .class được nạp vào bộ nhớ và thực thi bởi máy ảo Java (JVM)



3.4. Kiến trúc chương trình … (tt)

JVM và Java "bytecode"

- Chương trình Java không biên dịch mã nguồn thành ngôn ngữ máy đích mà biên dịch thành file dạng "bytecode" – file *.class
- Mỗi HĐH sẽ có thể hiện riêng của máy ảo Java – JVM
- Mã bytecode làm việc với JVM và JVM làm việc với HĐH



3.4. Kiến trúc chương trình … (tt)

- Các thành phần chính của một tập tin mã nguồn java
  - Gói – Package, Thư viện có sẵn – Library, Lớp – Class

Package

- Các class được lưu trong một khối thống nhất gọi là package
- Cú pháp: package <tên gói>;
- Các gói có thể được lồng vào nhau như cấu trúc thư mục
- Ưu điểm:
  - Các lớp được tổ chức riêng trong các gói riêng
  - Trong một gói không các lớp không trùng tên, trong một dự án lớn việc trùng tên các gói khác nhau.
  - Dễ quản lý trong các dự án lớn
  - Tên lớp có thể dùng định danh đối tượng

3.4. Kiến trúc chương trình … (tt)

Library

- Dùng thư viện có sẵn
- Cú pháp: import <ten goi>;
- Ví dụ: import java.util.*; import java.io.File, …

Class

- Cú pháp

```
class ClassName
{
    //fields, constructors
    // method declaration
}
```

3.4. Kiến trúc chương trình … (tt)

```
public class Rectangle
{
  private float length;
  private float width;

  public Rectangle()
  {
   length = 0;
   width = 0;
  }

  public Rectangle(float l, float w)
  {
   length = l;
   width = w;
  }

  public float area()
  {
   return length * width;
  }
}
```

Tên của lớp

1. Sử dụng quy tắc đặt tên
2. Luôn viết hoa chữ cái đầu tiên
3. Dùng danh từ để đặt tên

Dữ liệu thành phần

- Là những dữ liệu cần phải có

Khởi tạo

- Định nghĩa cách thức thể hiện 1 đối tượng
- Có tên giống tên lớp
- Giống như hàm trong C nhưng không có kiểu dữ liệu trả về

Các phương thức (method)

- **Những hành vi có thể thực hiện**

1. Như hàm trong C
2. Sử dụng động từ để đặt tên
3. Luôn viết thường chữ cái đầu tiên

3.4. Kiến trúc chương trình … (tt)

- Để thực thi chương trình, trình ứng dụng Java (Java application) bắt buộc phải có 1 lớp mà trong đó định nghĩa phương thức main.

- Phương thức main() trong lớp public được triệu hồi bởi JVM để bắt đầu thực thi ứng dụng.

```
public class RectangleDemo
{
    public static void main(String[] args)
    {
        Rectangle rec = new Rectangle(3, 4);

        float area = rec.area();

        System.out.println("Area: " + area);
    }
}
```

3.4. Kiến trúc chương trình … (tt)

- Đối tượng phải được tạo trước khi được sử dụng trong chương trình.
  1. Khai báo 1 biến để lưu giữ tham chiếu đến đối tượng (đối tượng chỉ có thể được thao tác thông qua tham chiếu)
  2. Tạo đối tượng: bằng cách sử dụng toán tử new (ngầm định gọi đến hàm khởi tạo – hàm dựng)

```
public class RectangleDemo
{
    public static void main(String[] args)
    {
        Rectangle rec = new Rectangle(3, 4);

        float area = rec.area();

        System.out.println("Area: " + area);
    }
}
```

3.4. Kiến trúc chương trình … (tt)

Gọi phương thức

- Sử dụng toán tử dấu chấm (the '.' operator)
- Cú pháp (Syntax):
  - <tên biến đối tượng tham chiếu>‘.’<tên phương thức được gọi>

```
public class RectangleDemo
{
    public static void main(String[] args)
    {
        Rectangle rec = new Rectangle(3, 4);

        float area = rec.area();

        System.out.println("Area: " + area);
    }
}
```