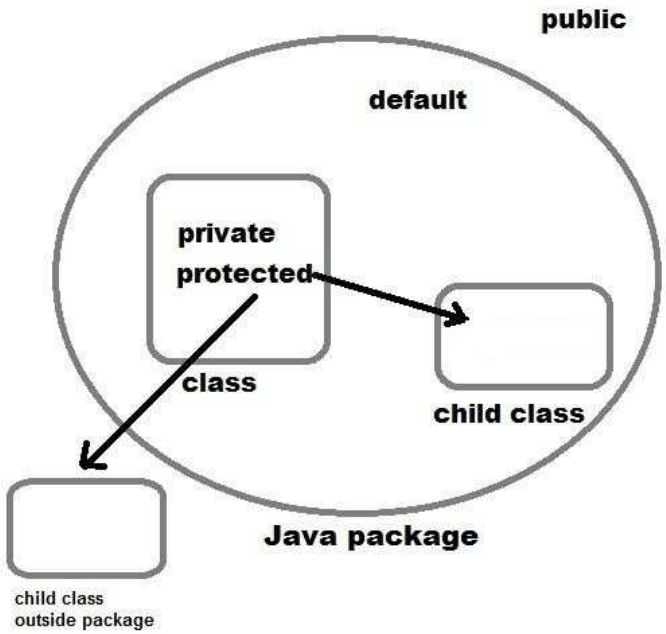
Bài 8
Bổ từ và gói (package)

Mục tiêu

- Các bổ từ truy cập.
- Biến và phương thức static
- Package
- File thư viện Jar

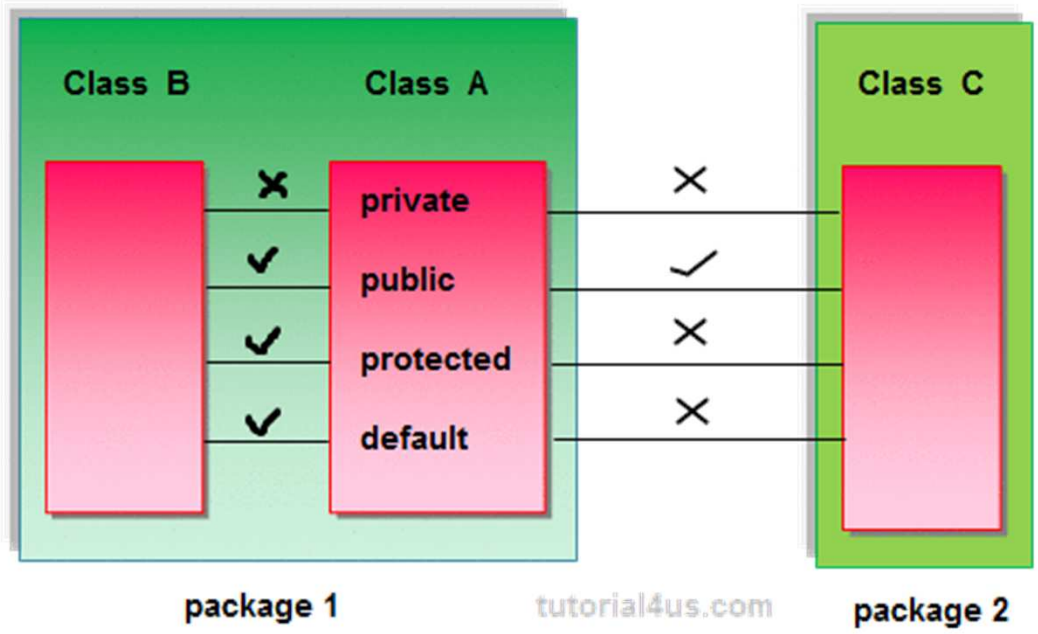


Bổ từ truy cập

- Java là ngôn ngữ đóng gói chặt chẽ.
- Java cung cấp các mô tả truy cập như: ***public***, ***protected***, ***default*** và ***private*** để hỗ trợ giới hạn khả năng truy cập tới lớp và các lớp thành viên.
- Để tăng cường hơn nữa, Java cung cấp trường và phương thức bổ sung để **hạn chế hơn nữa quyền truy cập** vào các thành viên của một lớp, hỗ trợ ngăn ngừa thay đổi mã trái phép.

Bổ từ truy cập

- Trường và phương thức bổ sung là các từ khóa để xác định khu vực và phương thức cung cấp kiểm soát truy cập cho người dùng.
- Một số có thể kết hợp với mô tả truy cập như public, private.



Bổ từ truy cập

Các bổ từ truy cập là:

- **volatile**
- **native**
- **transient**
- **final**

Bổ từ truy cập Volatile

Bổ từ `volatile` cho phép nội dung của biến được đồng bộ qua các luồng.

Luồng là thành phần độc lập thực hiện mã trong chương trình. Có thể có nhiều luồng chạy đồng thời trong chương trình.

Bổ từ `volatile` chỉ áp dụng cho một trường.

Constructors, phương thức, các lớp, và interface không áp dụng với bổ từ này.

Bổ từ `volatile` không thường xuyên được sử dụng.

Khi làm việc với nhiều luồng trong chương trình, bổ từ `volatile` mới có thể được sử dụng.

Bổ từ truy cập Volatile

Khi mà nhiều luồng trong chương trình sử dụng chung một biến, thông thường, mỗi luồng sẽ có một bản sao của biến trong bộ nhớ tạm.

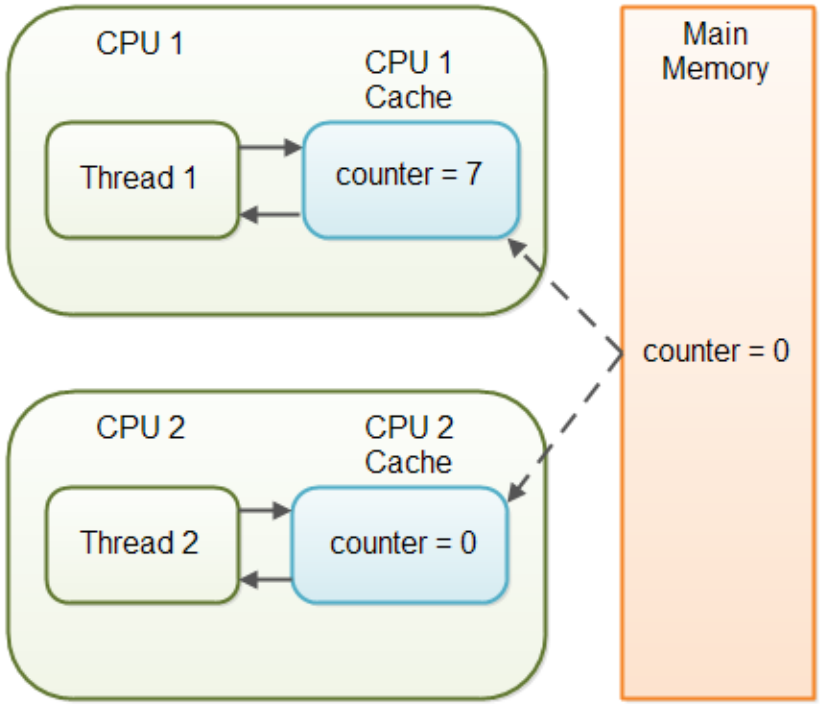
Trong trường hợp như vậy, nếu giá trị của biến được cập nhật, nó chỉ cập nhật bản sao trong bộ nhớ tạm cục bộ của luồng.

Nếu một luồng khác cũng sử dụng giá trị của biến thì không nhận được sự cập nhật đấy.

- Để tránh vấn đề này, một biến sẽ được khai báo với từ khóa volatile.
- Khi một luồng giá trị của biến thay đổi thì sự cập nhật này sẽ được cập nhật vào bộ nhớ biến chính.

Bổ từ truy cập Volatile

Luồng sẽ sao chép giá trị của biến vào bộ nhớ tạm (cache). Nên khi có sự cập nhật thì dữ liệu sẽ không đồng bộ ở tất cả các luồng.



Bổ từ truy cập Volatile

```java
/**
 *
 * @author VuTuanMinh
 */
public class TestVolatile extends Thread {

    volatile boolean keepRunning = true;

    public void run() {
        long count = 0;
        while (keepRunning) {
            count++;
        }

        System.out.println("Thread terminated." + count);
    }

    public static void main(String[] args) throws InterruptedException {
        TestVolatile t = new TestVolatile();
        t.start();
        Thread.sleep(1000);
        t.keepRunning = false;
        System.out.println("keepRunning set to false.");
    }

}
```

Bổ từ truy cập native

Bổ từ native có tính chất sau:

- Chỉ áp dụng với phương thức.
- Nó chỉ ra rằng việc thực hiện phương thức này không phải bằng mã nguồn Java mà bằng mã ngôn ngữ khác (VD: C hay C++).
- Constructors, fields, classes, và interfaces không sử dụng với bổ từ này.
- Phương thức khai báo với bổ từ `native` có thể gọi phương thức native.
- Mã nguồn Java thường chỉ chứa khai báo của phương thức native và không thực hiện.
- Việc thực hiện phương thức này sẽ ở thư viện bên ngoài JVM.

Bổ từ truy cập native

File Java khai báo phương thức với bổ từ native, load thư viện bên ngoài:

Main.java:

```
public class Main {
    public native int square(int i);
    public static void main(String[] args) {
        System.loadLibrary("Main");
        System.out.println(new Main().square(2));
    }
}
```

Bổ từ truy cập native

Thư viện C:

Main.c:

```
#include <jni.h>
#include "Main.h"

JNIEXPORT jint JNICALL Java_Main_square(
    JNIEnv *env, jobject obj, jint i) {
  return i * i;
}
```

Bổ từ truy cập native

Biên dịch và chạy:

Compile and run:

```
sudo apt-get install build-essential openjdk-7-jdk
export JAVA_HOME='/usr/lib/jvm/java-7-openjdk-amd64'
javac Main.java
javah -jni Main
gcc -shared -fpic -o libMain.so -I${JAVA_HOME}/include \
  -I${JAVA_HOME}/include/linux Main.c
java -Djava.library.path=. Main
```

Output:

```
4
```

Bổ từ truy cập transient

Khi ứng dụng Java thực thi, đối tượng (object) sẽ được tải vào bộ nhớ Random Access Memory (RAM).

Objects có thể được lưu trữ bên ngoài JVM để sử dụng sau này

Điều này xác định phạm vi và tuổi thọ đối tượng.

Quá trình lưu trữ một đối tượng trong một lưu trữ liên tục gọi là serialization.

Đối với bất kỳ đối tượng được tuần tự hóa, các lớp phải thực thi interface Serializable.

Nếu bổ từ `transient` sử dụng với biến, nó sẽ không lưu trữ và sẽ không trở thành một phần của trạng thái liên tục của đối tượng.

Bổ từ `transient` hữu ích trong việc bảo mật, ngăn chặn dữ liệu nhạy cảm bị sao chép vào một nguồn không có cơ chế an ninh nào thực thi.

Bổ từ `transient` làm giảm dữ liệu được tuần tự hóa, cải thiện hiệu năng và giảm chi phí.

Bổ từ truy cập transient

Bổ từ chỉ sử dụng với biến

Ví dụ:

```
class Circle {

    transient float PI; // transient variable that will not persist
    float area; // instance variable that will persist

}
```

Bổ từ truy cập final

Bổ từ `final` sử dụng khi muốn hạn chế sự thay đổi đối với thành viên lớp hay dữ liệu.

Bổ từ `final` có thể được sử dụng với biến, phương thức, và lớp.

Final Variable

Biến sẽ trở thành hằng số và không thể chỉnh sửa, thay đổi.

Biến `final` phải được gán giá trị ngay khi nó khởi tạo.

Sẽ phát sinh lỗi khi biến `final` bị cố ý thay đổi giá trị sau khi gán lần đầu lúc khởi tạo.

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

Bổ từ truy cập final

Final Method

Không thể bị ghi đè hay ẩn đi trong lớp con.

Lý do là để ngăn chặn lớp con thay đổi ý nghĩa của phương thức gốc.

Thường được dùng để tạo ra hằng số ngẫu nhiên trong ứng dụng toán học

VD:

```
final float getCommission(float sales){
  System.out.println("A final method. . .");
}
```

Bổ từ truy cập final

Final Class

Không thể có kế thừa hay lớp con.

Xây dựng lên lớp chuẩn.

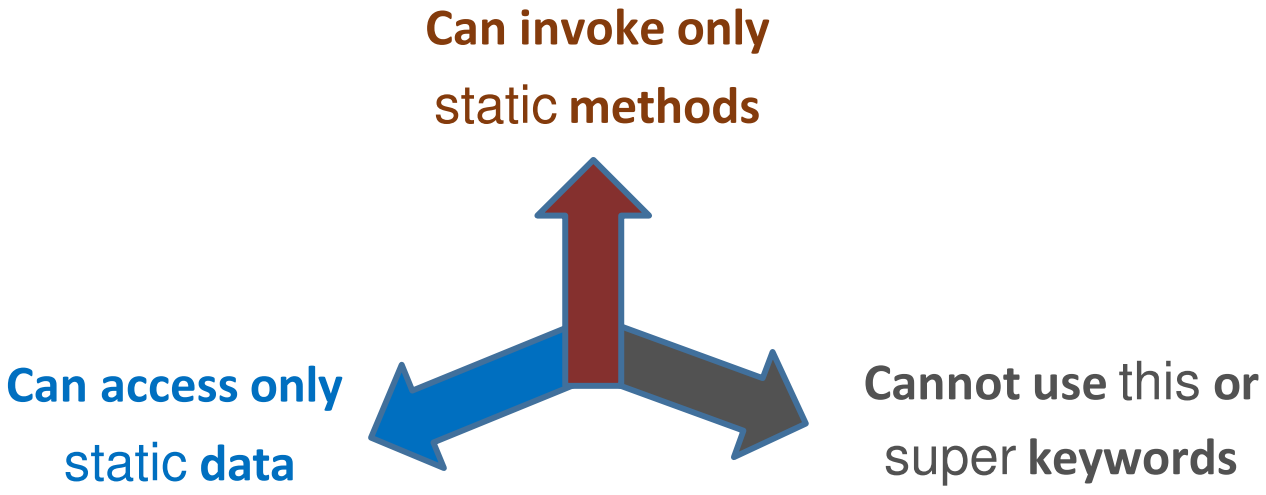
Các biến và phương thức trong class cũng hiểu ngầm ở dạng thức final

VD:

```
public final class Stock {
...
}
```

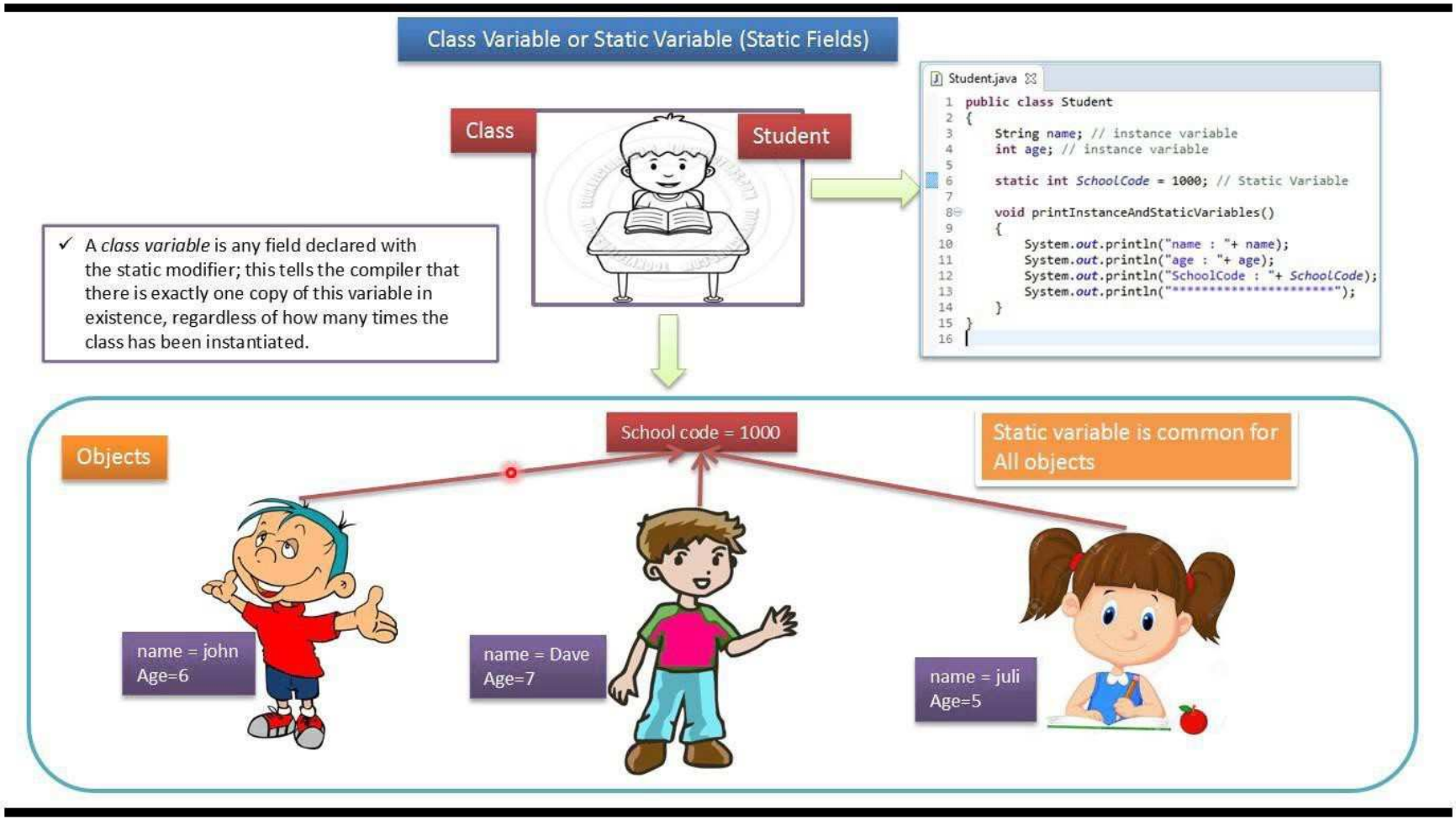
Biến và phương thức static

- Xem xét tình huống khi mà muốn tạo một truy cập theo dõi số lượng các đối tượng truy cập một phương thức cụ thể.
- Với yêu cầu này, một biến sẽ phải có khả năng chia sẻ giữa tất cả các đối tượng (tạo từ cùng class), bất kỳ sự thay đổi nào ở đối tượng bản sao sẽ được cập nhật ở tất cả đối tượng còn lại.



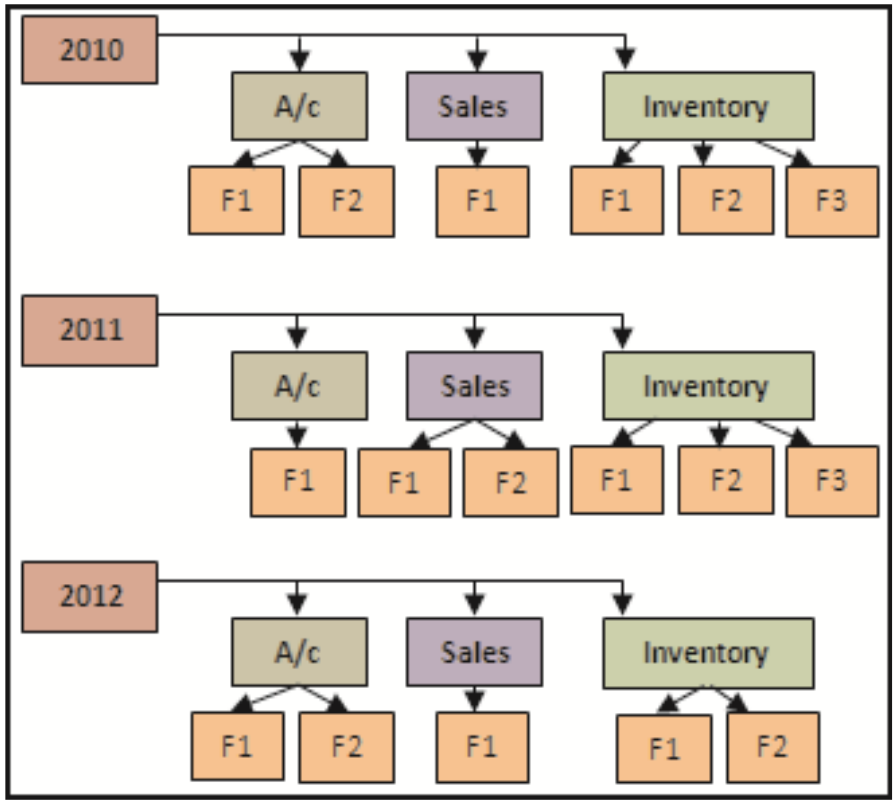
Biến và phương thức static

Xem xét vd sau để hiểu ý nghĩa static



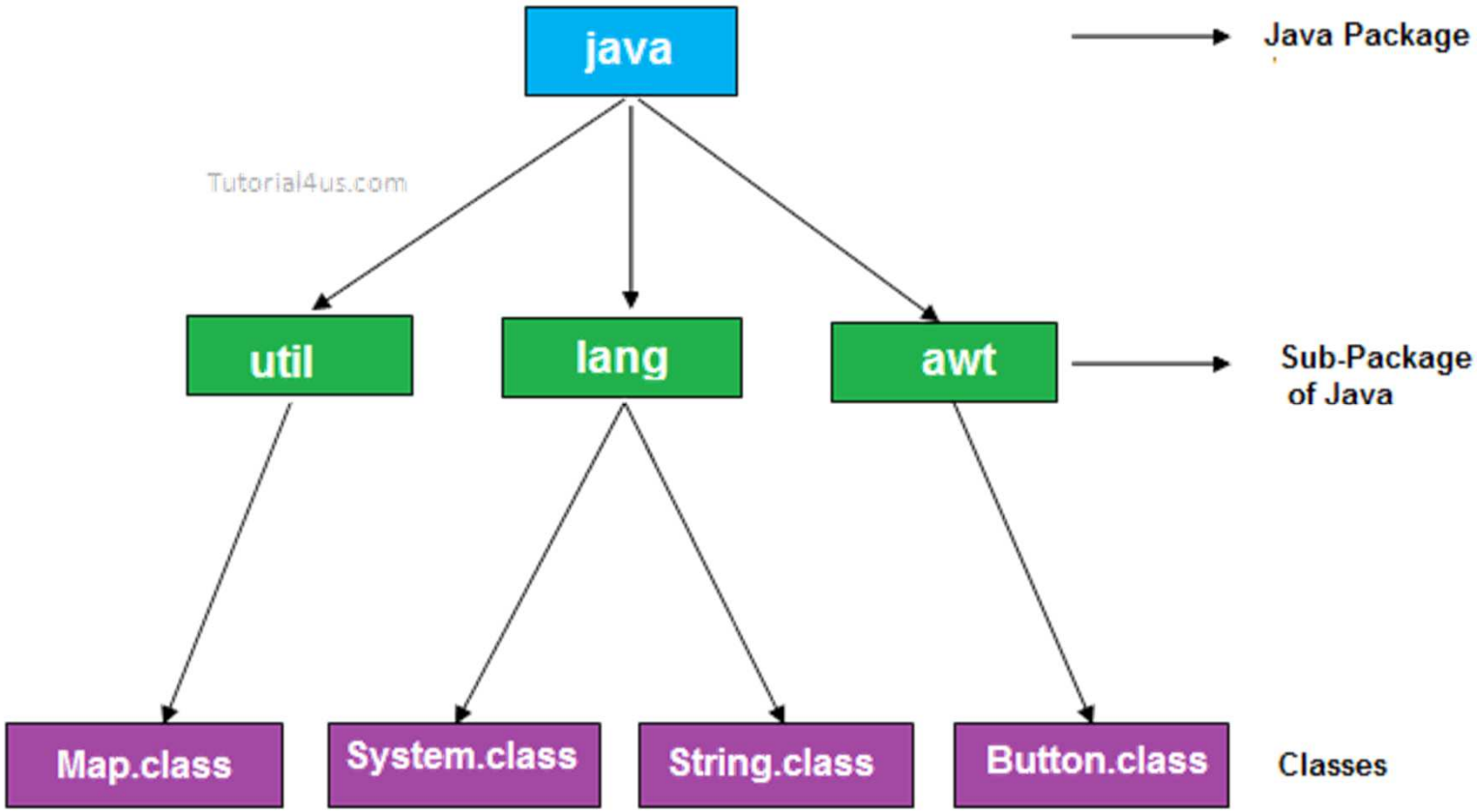
Package

- **Java** tổ chức file nguồn vào các gói (**package**).
- Package là một **namespace** có tác dụng **nhóm** các **class, interface**... và **tổ chức** chúng như một đơn vị.



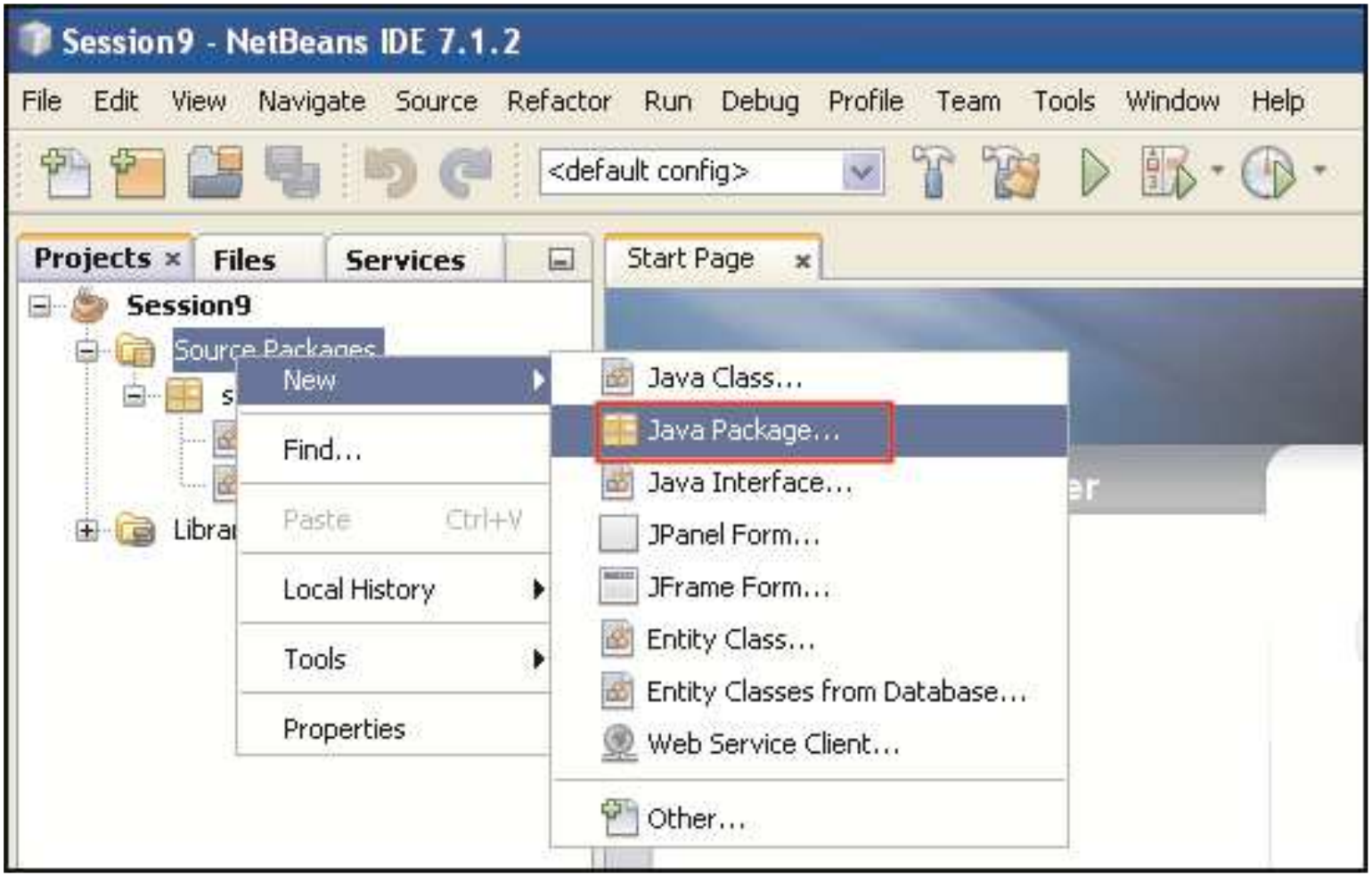
Package

Java tổ chức chính thư viện của mình cũng vào các gói (**package**).



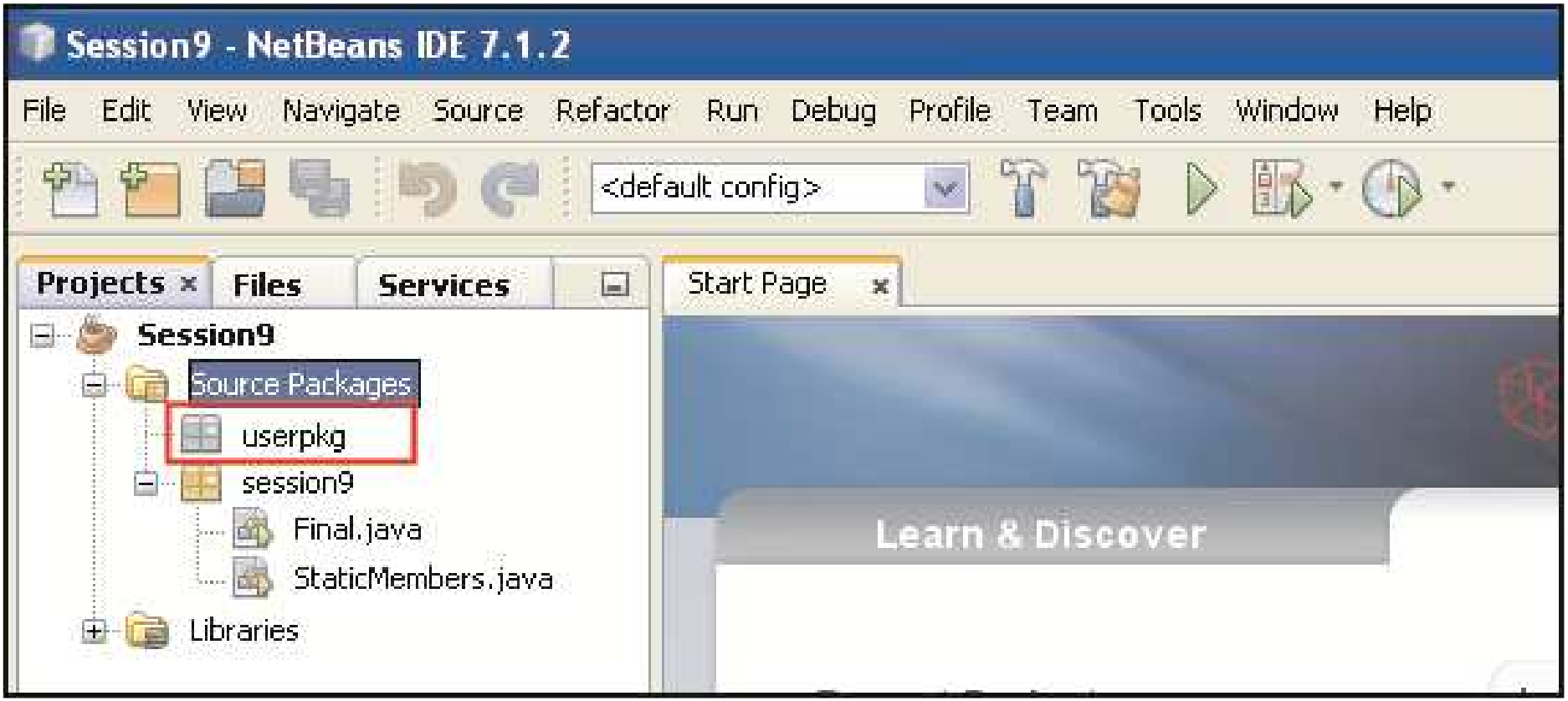
Package

Tạo package.

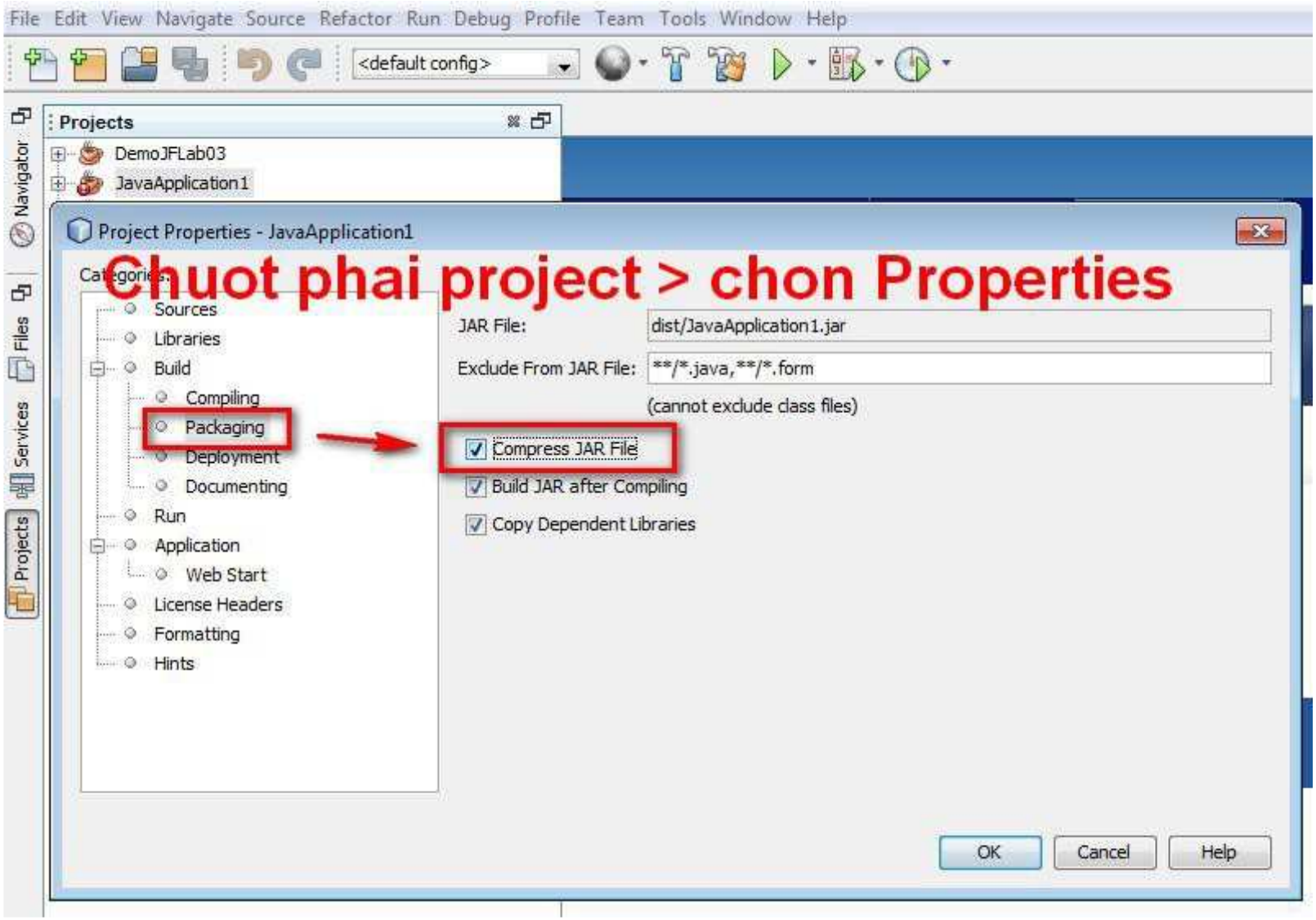


Package

Chuột phải vào package để tạo/thêm file.

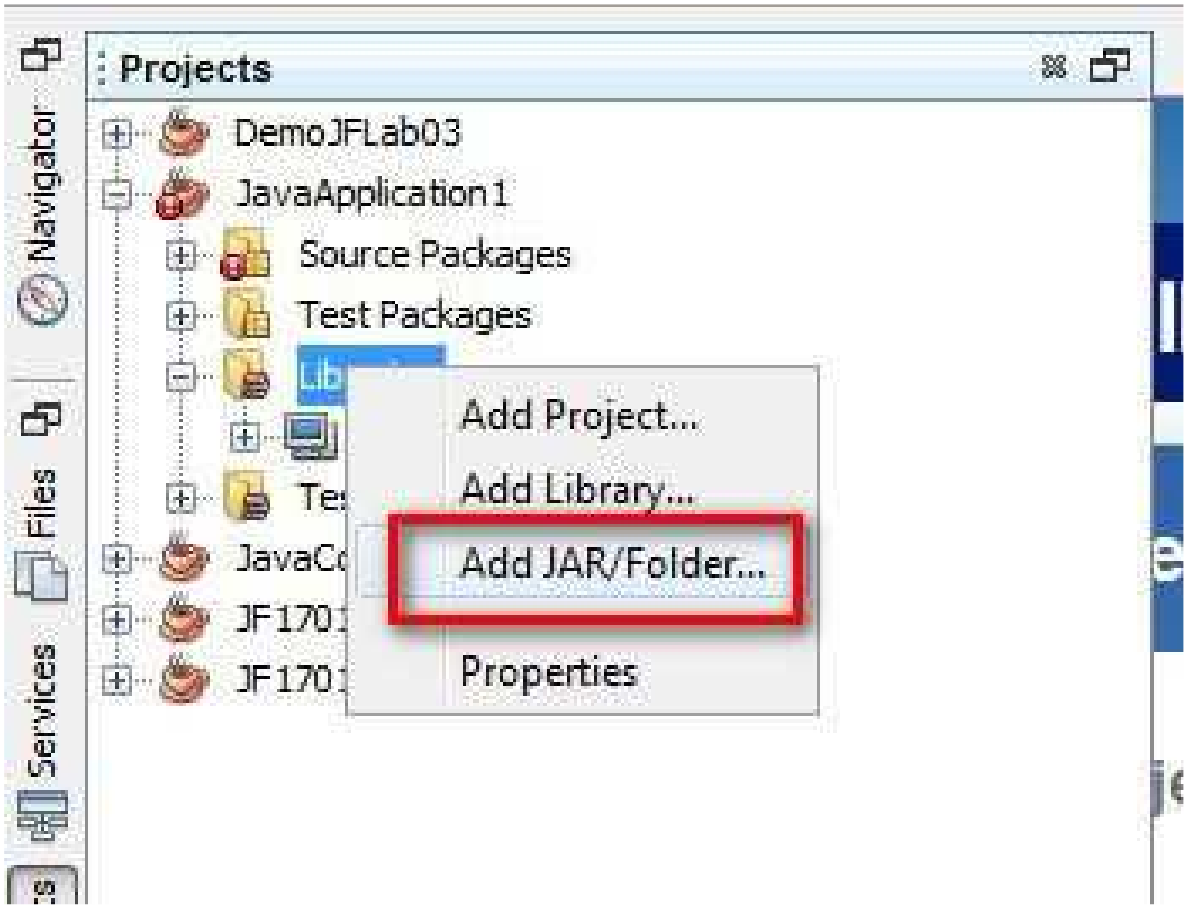


File thư viện Jar



File thư viện Jar

Thêm thư viện jar



Tóm tắt bài học

- ✓ Các **bổ từ truy cập** hỗ trợ thêm khả năng **kiểm soát giới hạn**.
- ✓ Biến hay phương thức **static** có thể gọi trực tiếp từ class mà không cần khởi tạo đối tượng.
- ✓ **Package** là **namespace** giúp việc tổ chức mã nguồn khoa học, quy củ, tiện việc phát triển, bảo trì, nâng cấp.
- ✓ Mã nguồn java có thể được đóng gói thành thư viện (file **jar**).