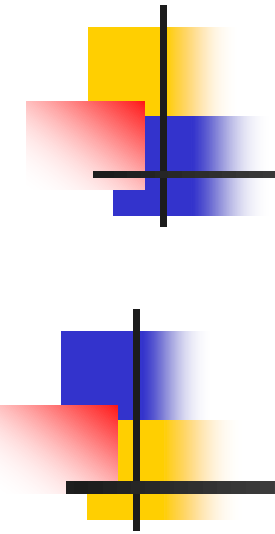
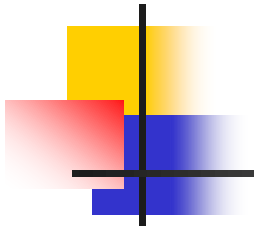


TỪ KHÓA STATIC VÀ ACCESS MODIFIER TRONG JAVA

Từ Khóa Static Trong Java

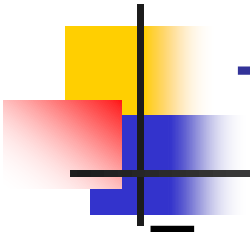
Access Modifier Trong Java



Từ Khóa Static Trong Java

Từ Khóa Static Trong Java

- Từ khóa static trong Java được sử dụng chính để quản lý bộ nhớ.
- Chúng ta có thể áp dụng từ khóa static với các biến, các phương thức, các khối, các lớp lồng nhau (nested class).
- Từ khóa static thuộc về lớp chứ không thuộc về instance (thể hiện) của lớp.



Từ Khóa Static Trong Java

- Trong java, static có thể là:
  - Biến static: Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.
  - Phương thức static: Khi bạn khai báo một phương thức là static, thì phương thức đó gọi là phương thức static.
  - Khối static: Được sử dụng để *khởi tạo thành viên dữ liệu static {……}*

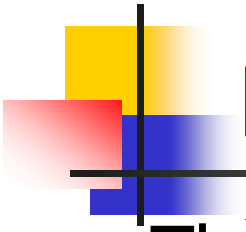
Biến Static Trong Java

- Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.
- Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (không là duy nhất cho mỗi đối tượng)
- Ví dụ: tên công ty của nhân viên, tên trường của các sinh viên, …
- Biến static lấy bộ nhớ chỉ một lần trong Class Area tại thời gian tải lớp đó.

Biến Static Trong Java

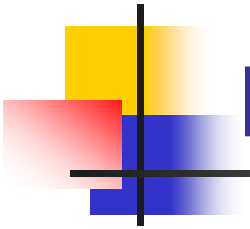
- Sử dụng biến static giúp chương trình sử dụng bộ nhớ hiệu quả hơn.
- Vấn đề khi không sử dụng biến static:

```
class Student{
    int  id;
    String name;
    String college=“Dai hoc Bach Khoa";
    }
```



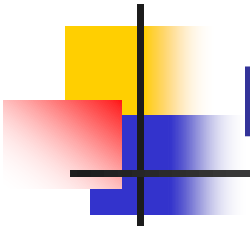
Biến Static Trong Java

- Thành viên dữ liệu sử dụng bộ nhớ mỗi khi đối tượng được tạo.
- Tất cả sinh viên có id và name là thuộc tính riêng, college là thuộc tính chung của tất cả đối tượng.
- Nếu chúng ta tạo nó là static, thì trường này sẽ chỉ sử dụng bộ nhớ một lần để lưu biến này
- Thuộc tính static trong Java được chia sẻ tới tất cả đối tượng *(Student.java)*



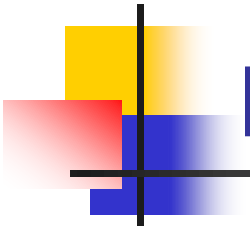
Phương Thức Static Trong Java

- Nếu ta áp dụng từ khóa static với bất cứ phương thức nào, thì phương thức đó được gọi là phương thức static *(Student1.java)*
  - Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.
  - Một phương thức static gọi mà không cần tạo một instance của một lớp.
  - Phương thức static có thể truy cập biến static và có thể thay đổi giá trị của nó.



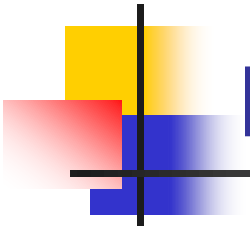
Phương Thức Static Trong Java

- Có hai hạn chế chính đối với phương thức static *(Disadvantage.java)*
  - Phương thức static không thể sử dụng biến non-static hoặc gọi trực tiếp phương thức non-static.
  - Từ khóa this và super không thể được sử dụng trong ngữ cảnh static.



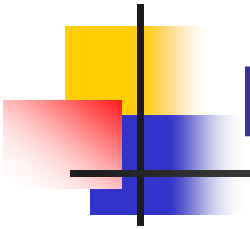
Khối Static Trong Java

- Khối static trong Java
  - Được sử dụng để khởi tạo thành viên dữ liệu static.
  - Nó được thực thi trước phương thức main() tại lúc tải lớp.
- Ví dụ : *Student2.java*



Khối Static Trong Java

- Tại sao phương thức main trong Java là static?
  - Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static
  - Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ phụ



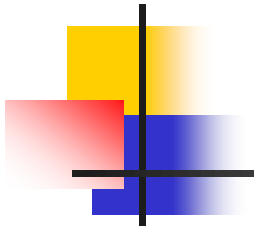
Khối Static Trong Java

- Chúng ta có thể thực thi một chương trình mà không có phương thức main()?
  - Có, một trong các cách đó là khối static trong phiên bản trước của JDK (không phải JDK 1.7)
- Ví dụ: *Student3.java*

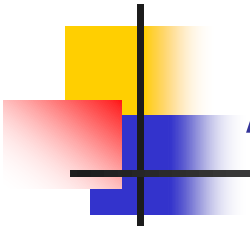
Khối Static Trong Java

- Ví dụ: public class Student3 {

```
    static {
        System.out.println("static block is invoked");
        System.exit(0);
    }}
// TH < jdk7: static block is invoked
// TH >= jdk7: Main method not found in class
// Student3, please define the main method as:
// public static void main(String[] args)
```

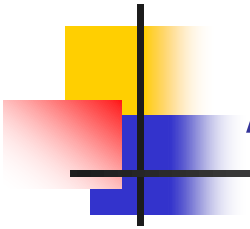


Access Modifier trong Java



Access Modifier trong Java

- Có hai loại Access Modifier trong Java, đó là:
  - Access Modifier
  - Non-access Modifier
- Access Modifer trong Java xác định phạm vi có thể truy cập của biến, phương thức, constructor hoặc lớp.



Access Modifier trong Java

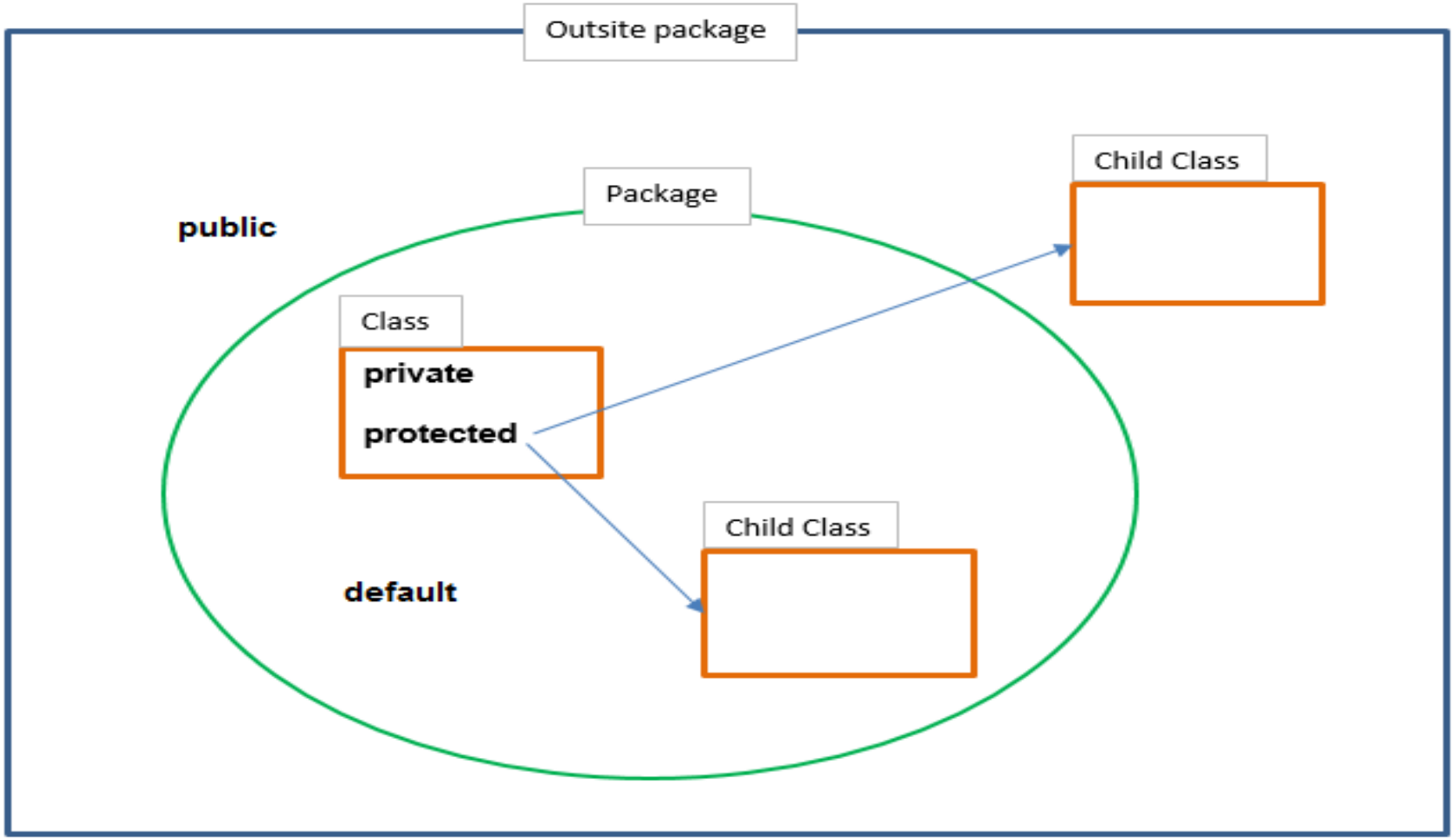
- Trong java, có 4 phạm vi truy cập của Access Modifier như sau:
  - private
  - default
  - protected
  - public
- Ngoài ra, còn có nhiều Non-access Modifier như static, abstract, synchronized, native, volatile, transient,…

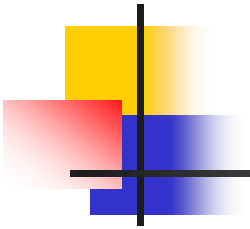
Access Modifier trong Java

- Bảng dưới đây mô tả khả năng truy cập của các Access Modifier trong java:

Access Modifier	Trong lớp	Trong package	Ngoài package bởi lớp con	Ngoài package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

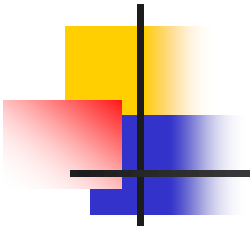
Access Modifier trong Java





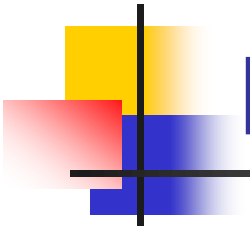
Phạm vi truy cập private

- Private Access Modifier chỉ được truy cập trong phạm vi lớp.
- Ví dụ: *A.java*
  - Ta tạo 2 lớp A và B. Lớp B chứa biến và phương thức được khai báo là private.
  - Ta cố gắng truy cập chúng từ bên ngoài lớp A. Điều này dẫn đến Compile time error



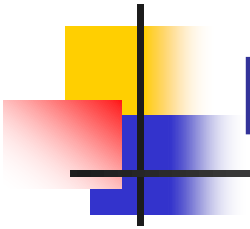
Phạm vi truy cập private

- *Vai trò của Private Constructor*
  - Nếu ta tạo bất kỳ constructor là private trong lớp, ta sẽ không thể tạo instance của class bên ngoài nó *(A1.java)*
- Lưu ý: *Một lớp không thể là private hoặc protected, ngoại trừ lớp lồng nhau*



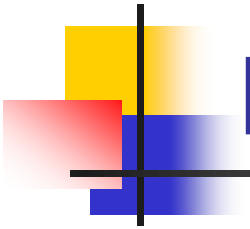
Phạm vi truy cập default

- Nếu bạn không khai báo modifier nào, thì nó chính là trường hợp mặc định.
- Default Access Modifier là chỉ được phép truy cập trong cùng package *(A2.java)*
- Theo ví dụ trên, phạm vi truy cập của lớp B2.java và phương thức method() của nó là mặc định nên ta không thể được truy cập từ bên ngoài package.



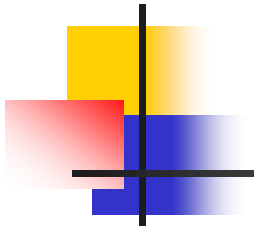
Phạm vi truy cập protected

- Protected access modifier được truy cập bên trong package và bên ngoài package nhưng phải kế thừa *(A3.java)*
- Protected access modifier có thể được áp dụng cho biến, phương thức, constructor
- *Protected access modifier không thể áp dụng cho lớp*



Phạm vi truy cập public

- Public access modifier được truy cập ở mọi nơi *(A4.java)*



Hết!!!