

# بخش ۶-ب: پکیج‌بندی حرفه‌ای و سطوح دسترسی جامع

تهیه شده توسط: سید سجاد پیراھش



0 5 10 00

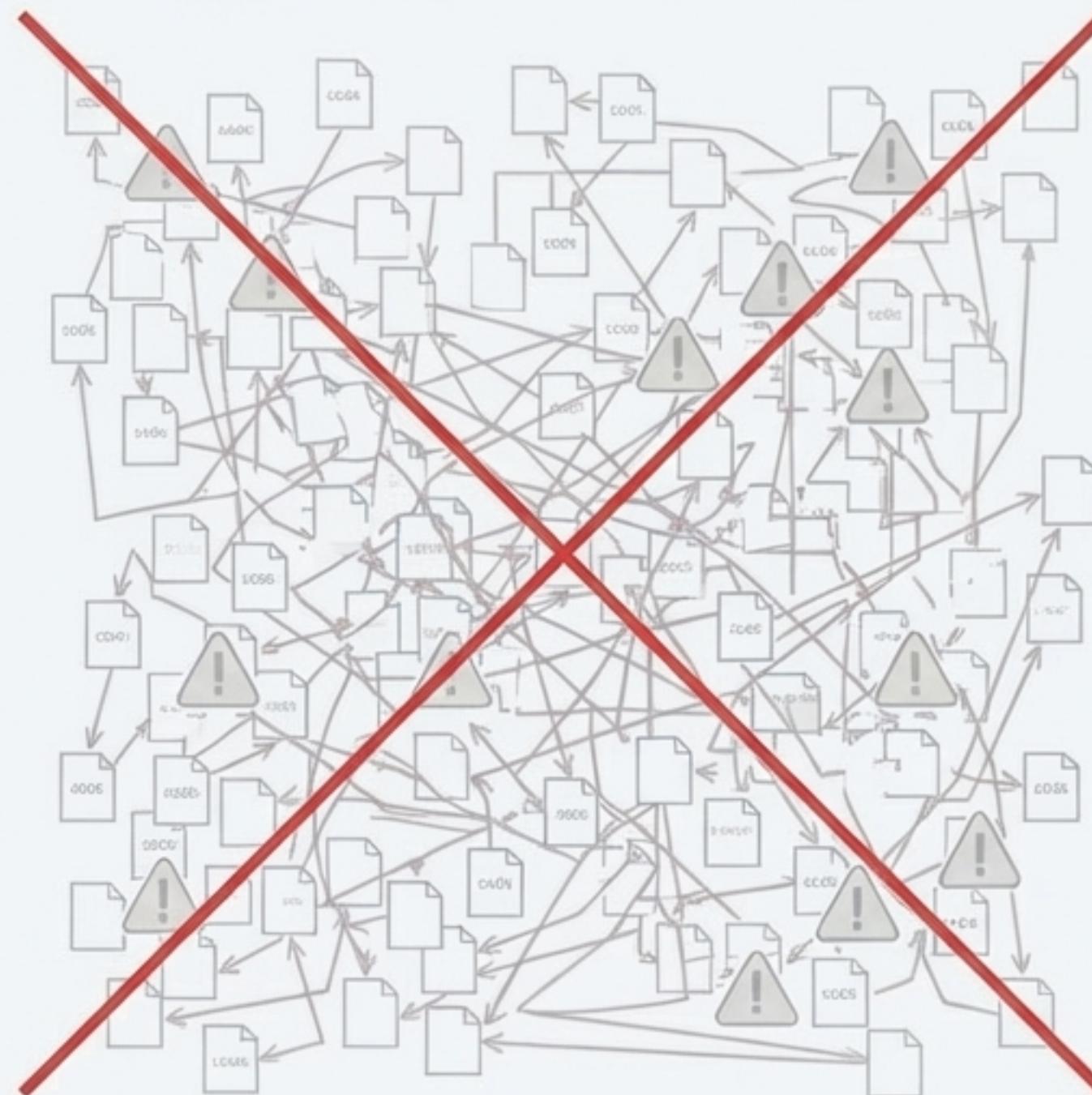
# بدون ساختار، پروژه شما محکوم به شکست است



در پروژه‌های بزرگ، صدها کلاس بدون سازماندهی منجر به فاجعه می‌شود:

- **تداخل نام (Name Collision):** کلاس `User` شما با کلاس `User` یک کتابخانه دیگر تداخل پیدا می‌کند.
- **عدم خوانایی:** پیدا کردن یک کلاس خاص مانند پیدا کردن سوزن در انبار کاه است.
- **نبود امنیت:** تمام جزئیات داخلی پروژه شما برای همه قابل مشاهده و تغییر است.
- **کابوس کار تیمی:** توسعه‌دهندگان دائمًا روی کار یکدیگر پا می‌گذارند.

# دو ستون اصلی برای ساخت نرم افزار پایدار



مهندسان حرفه‌ای از دو ابزار قدرتمند برای کنترل پیچیدگی استفاده می‌کنند:

1. **پکیج بندی استراتژیک (Strategic Packaging)**: مانند نقشه‌کشی و منطقه‌بندی یک شهر. هر چیزی جای مشخص خود را دارد.
2. **کنترل دسترسی جامع (Access Control)**: مانند تعیین سطوح امنیتی برای هر ساختمان. چه کسی اجازه ورود به کجا را دارد؟

# قرارداد دامنه معکوس جهانی (Reverse Domain Name): یک استاندارد جهانی



برای تضمین یکتایی جهانی، از دامنه اینترنتی خود به صورت معکوس استفاده کنید. این یک استاندارد صنعتی است، نه یک پیشنهاد.

**ساختار:** `com.mycompany.onlineshop.usermanagement`

- سطح بالا (Top-Level Domain): `com`
- نام سازمان: `mycompany`
- نام پروژه یا محصول: `onlineshop`
- نام مارک یا لایه: `usermanagement`

**قانون طلایی:** همیشه با حروف کوچک. همیشه معنادار.

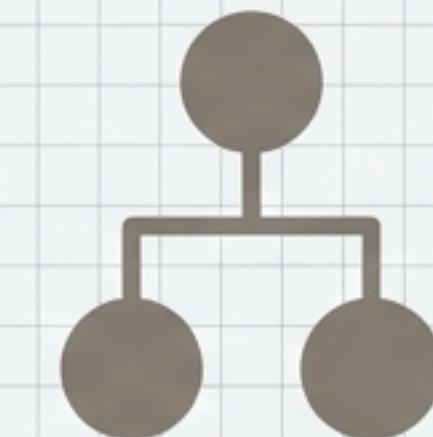
# فرمانروایی کد خود باشید: چهار کلید دسترسی



`private`: قلعه ناکس  
- (Fort Knox)  
فقط خود کلاس.



`default`: نگهبان محله  
- (Neighborhood Watch)  
فقط کلاس‌های همان پکیج.



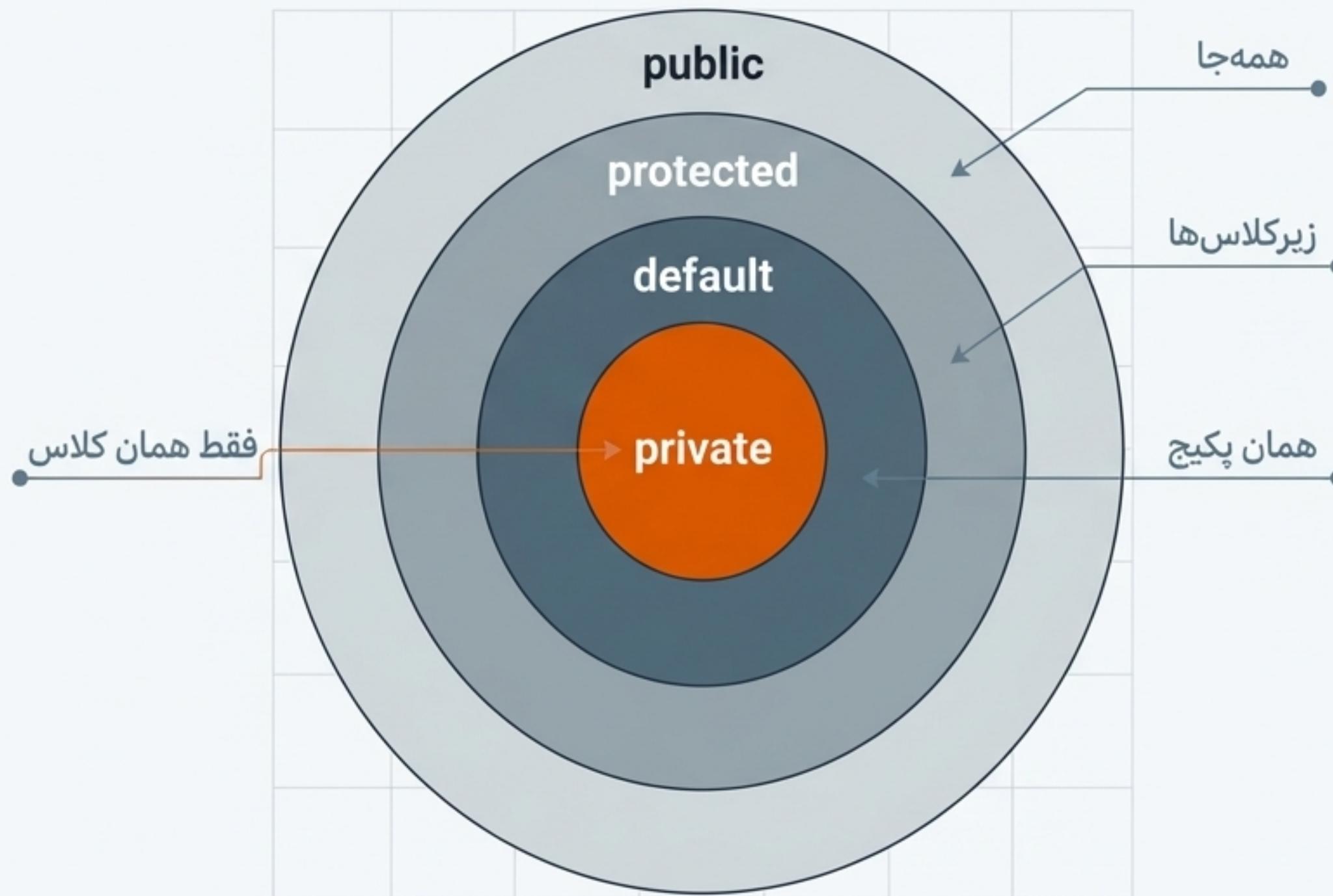
`protected`: حلقه خانوادگی  
- (Family Circle)  
همان پکیج + زیرکلاس‌ها  
در هر کجا.



`public`: پخش جهانی  
- (Global Broadcast)  
همه‌جا، بدون محدودیت.

جاوا به شما چهار سطح کنترل دقیق می‌دهد تا مشخص کنید چه کسی به اعضای کلاس شما (فیلد‌ها و متدها) دسترسی دارد.

# همه چیز در یک نگاه: ماتریس دید (Visibility Matrix)



این نمودار نشان می‌دهد هر سطح دسترسی از از کجا قابل مشاهده است. از داخل به خارج، دسترسی بازتر می‌شود.

- \* `private` ( فقط همان کلاس):
- \* `default` ( همان پکیج):  
    (Package-Private)
- \* `protected` ( زیرکلاس‌ها):
- \* `public` ( همه‌جا):

# مرجع سریع: مقایسه سطوح دقیق دسترسی

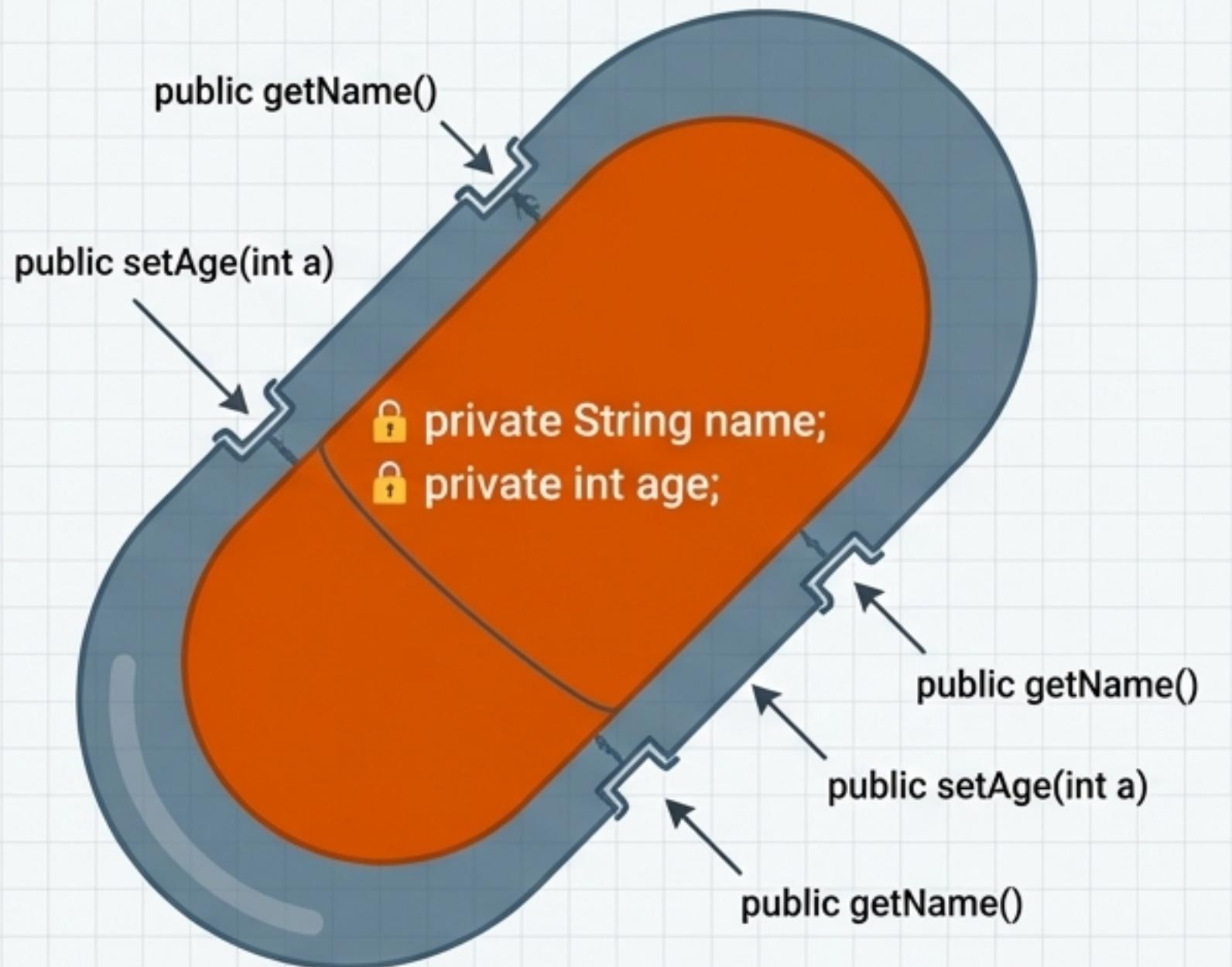
سطح دسترسی	کلمه کلیدی	همان کلاس	همان پکیج	زیرکلاس (پکیج دیگر)	کلاس دیگر (پکیج دیگر)
خصوصی	`private`	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
پکیج-خصوصی	(بدون کلمه)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
محفظت شده	`protected`	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
عمومی	`public`	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



## اصل کمترین دسترسی (Principle of Least Privilege)

همیشه با `private` شروع کنید و فقط در صورت نیاز دسترسی را افزایش دهید. این این اساس برنامه‌نویسی تدافعی است.

# چرا `public` کردن فیلدها یک گناه کبیره است؟



پنهانسازی اطلاعات (Encapsulation) یعنی محافظت از وضعیت داخلی (فیلدها) یک شیء و ارائه دسترسی کنترل شده از طریق متدهای عمومی.

چرا؟ چون شما باید یک "کنترل‌گر" (Control Freak) باشید!

- اعتبارسنجی (Validation): می‌توانید داده‌های ورودی را قبل از تنظیم، اعتبارسنجی کنید (مثلاً age نباید منفی باشد).
- منطق داخلی: می‌توانید هنگام دریافت یا تنظیم مقدار، منطق اضافی اجرا کنید (مثلاً لاغ کردن تغییرات).
- انعطاف‌پذیری: می‌توانید نحوه ذخیره‌سازی داخلی فیلد را در آینده بدون شکستن کدهای دیگر تغییر دهید.

فیلدهای public این کنترل را از شما می‌ربايند.

# از کد شکننده تا کد نفوذناپذیر

قبل: دسترسی مستقیم و خطرناک

```
// in package model
public class Student {
    public String name;
    public int studentId;
}

// in package service
Student s = new Student();
s.studentId = -123; // فاجعه! یک مقدار نامعتبر تنظیم شد
```

بعد: کپسوله‌سازی و کنترل کامل

```
// in package model
public class Student {
    private String name;
    private int studentId;

    public void setStudentId(int id) {
        if (id > 0) { // اعتبارسنجی
            this.studentId = id;
        }
    }

    public int getStudentId() {
        return this.studentId;
    }

    // ... other getters/setters
}
```

# چالش ۱: خطاهای دسترسی را شکار کنید!

در کد زیر، کدام خطوط باعث خطای کامپایل به دلیل نقض سطوح دسترسی می‌شوند و چرا؟

```
1 // package com.ecommerce.model;
2 public class Product {
3     private String id;
4     String name; // default access
5     protected double price;
6     public int stock;
7 }
8
9 // package com.ecommerce.service;
10 import com.ecommerce.model.Product;
11 public class ProductService {
12     public void displayProduct() {
13         Product p = new Product();
14         System.out.println(p.id); // خطای ۱: private
15         System.out.println(p.name); // خطای ۲: default
16         System.out.println(p.price); // خطای ۳: protected
17         System.out.println(p.stock);
18     }
19 }
```

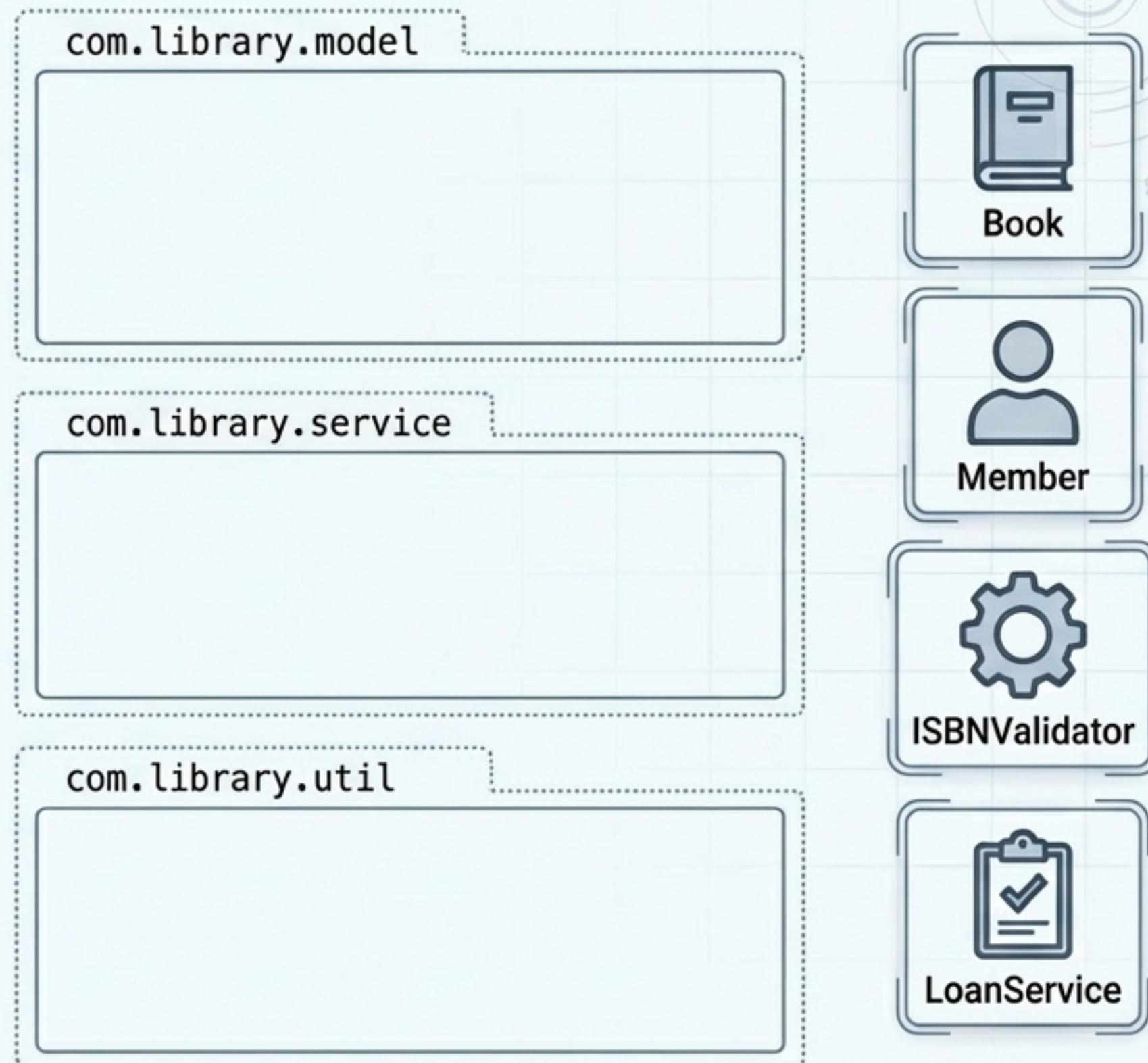
// صحیح: //

\* راهنمایی: ProductService در پکیج متفاوتی قرار دارد و زیرکلاس Product نیست.

# چالش ۲: معمار سیستم کتابخانه شوید!

یک سیستم کتابخانه با نیازمندی‌های زیر طراحی کنید. ساختار پکیج و سطح دسترسی مناسب برای هر کلاس/متدهای کلیدی را مشخص کنید:

- **موجودیت‌ها (Entities) :** (isbn , title) با فیلدۀای private مانند Book •  
(name , memberId) با فیلدۀای private مانند Member •
- **سرویس‌ها (Services) :** LoanService: دارای متدهای public برای امانت دادن کتاب.  
این متدهای جزئیات داخلی Book و Member دسترسی کنترل شده داشته باشد.
- **کلاس‌های کمکی (Helpers) :** ISBNValidator: یک کلاس کمکی که فقط باید توسط سرویس‌های داخل پکیج خودش استفاده شود. سطح دسترسی این کلاس و متدهای آن چه باید باشد؟ ("package-private": پاسخ)



## چالش ۳. چالش ۳: سازماندهی پکیج‌های یک فروشگاه آنلайн

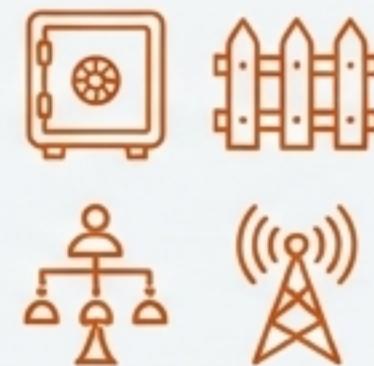
پروژه یک فروشگاه آنلайн را با استفاده از استاندارد دامنه معکوس (‘ir.myonlineshop’) سازماندهی کنید.  
پکیج‌های مناسب برای هر یک از مأموریت‌های زیر را ایجاد کنید:

مدیریت کاربران (User Management)	کاتالوگ محصولات (Product Catalog)	سبد خرید (Shopping Cart)	پردازش سفارش (Order Processing)
User Profile AuthService	Product Category ProductRepository	Cart CartItem CartService	Order Payment OrderProcessor
پکیج پیشنهادی: <code>ir.myonlineshop.user</code>	پکیج پیشنهادی: <code>ir.myonlineshop.catalog</code>	پکیج پیشنهادی: <code>ir.myonlineshop.cart</code>	پکیج پیشنهادی: <code>ir.myonlineshop.order</code>

# جعبه ابزار معمار حرفه‌ای



پکیج (Package) : ابزار شما برای سازماندهی و جلوگیری از تداخل نام.



سطح دسترسی: ابزار شما برای امنیت و پیاده‌سازی .Encapsulation  
**private** < **default** < **protected** < **public** <

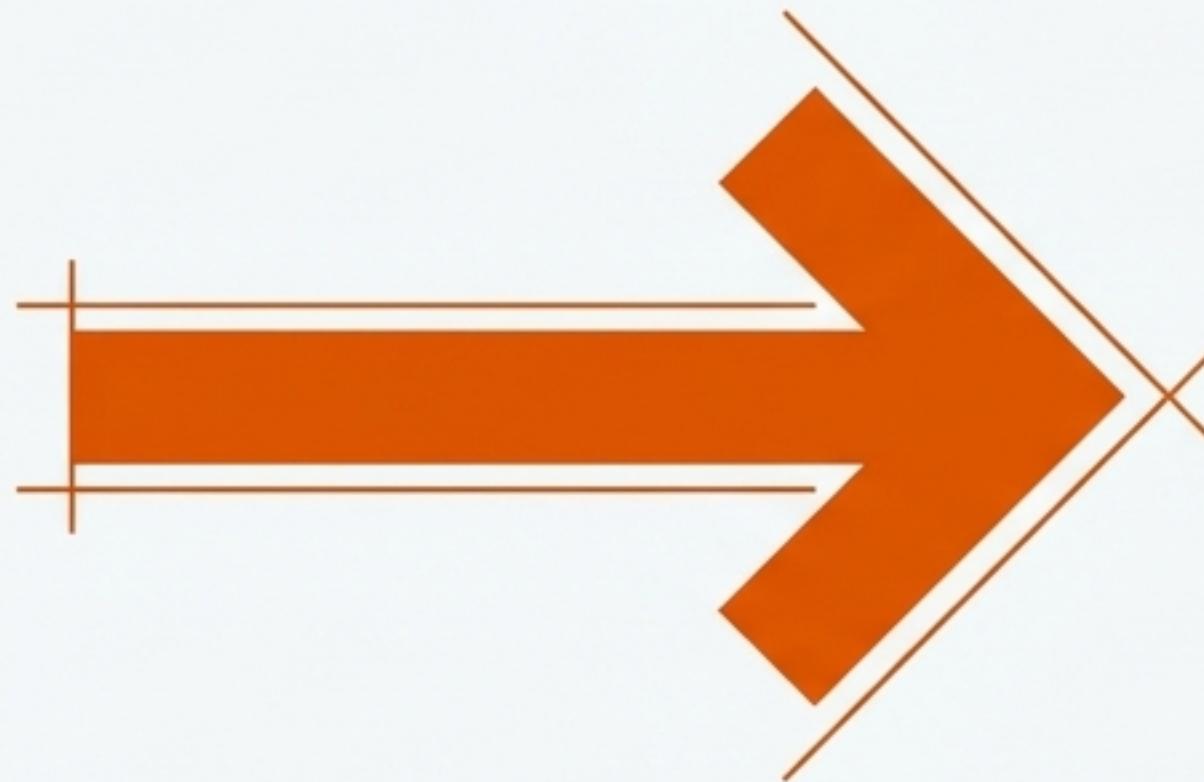


قرارداد نام‌گذاری: همیشه از دامنه معکوس استفاده کنید.  
`(com.company.project)`



اصل طلایی (Golden Rule)  با محدودترین سطح دسترسی (**private**) شروع کنید و فقط در صورت لزوم آن را باز کنید.  
این یعنی **برنامه‌نویسی تدافعی** .(Defensive Programming)

## در بخش بعدی...



**تفکر شیءگرا:** تغییر پارادایم از "چگونه" به "چه کسی"

ما یاد گرفتیم که چگونه کد را ساختاربندی کنیم. اکنون یاد می‌گیریم که چگونه مانند یک **معمار شیءگرا** فکر کنیم و مسئولیت‌ها را به اشیاء مناسب واگذار نماییم.