

بخش ۸-الف: مبانی UML

الفبای مهندسی نرم افزار

تهییه شده توسط: سید سجاد پیراهاش



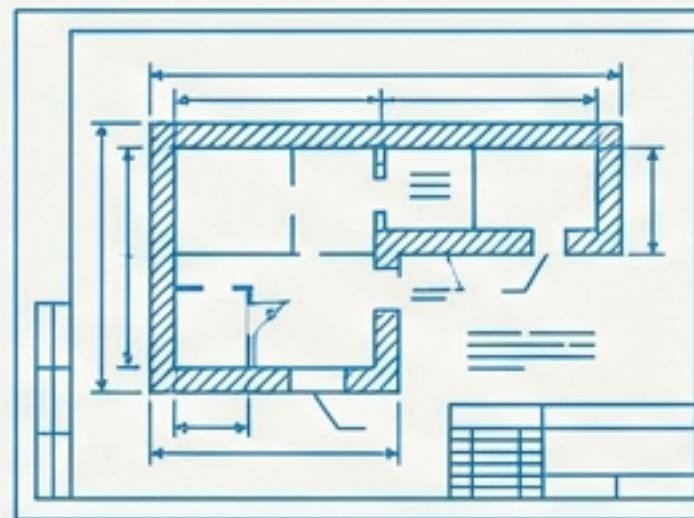
برنامه‌نویس یا مهندس نرم‌افزار؟

فقط کدنویسی



- مانند شروع به چیدن آجر برای ساختن یک آسمان‌خراش بدون داشتن نقشه‌های مهندسی.
- نتیجه: سیستمی شکننده که با اولین تغییر نیازها فرو می‌ریزد.
- هزینه تغییرات بالا و نگهداری دشوار.

طراحی مهندسی



- استفاده از یک نقشه راه دقیق قبل از ساخت.
- نتیجه: سیستمی مستحکم، قابل توسعه و قابل فهم.
- این نقشه راه، **UML** است.

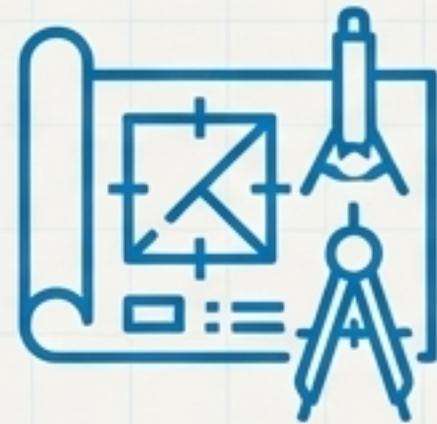
چرا UML حیاتی است؟ (فراتر از یک نقاشی ساده)

UML زبان مدل‌سازی یکپارچه برای تمام مهندسان نرم‌افزار است.



قدرت "دید از بالا" (Abstraction)

UML به شما اجازه می‌دهد ساختار کلان سیستم را بدون گم شدن در جزئیات پیاده‌سازی (مثل حلقه for یا شرط if) ببینید.



طراحی پیش از ساخت (Blueprinting)

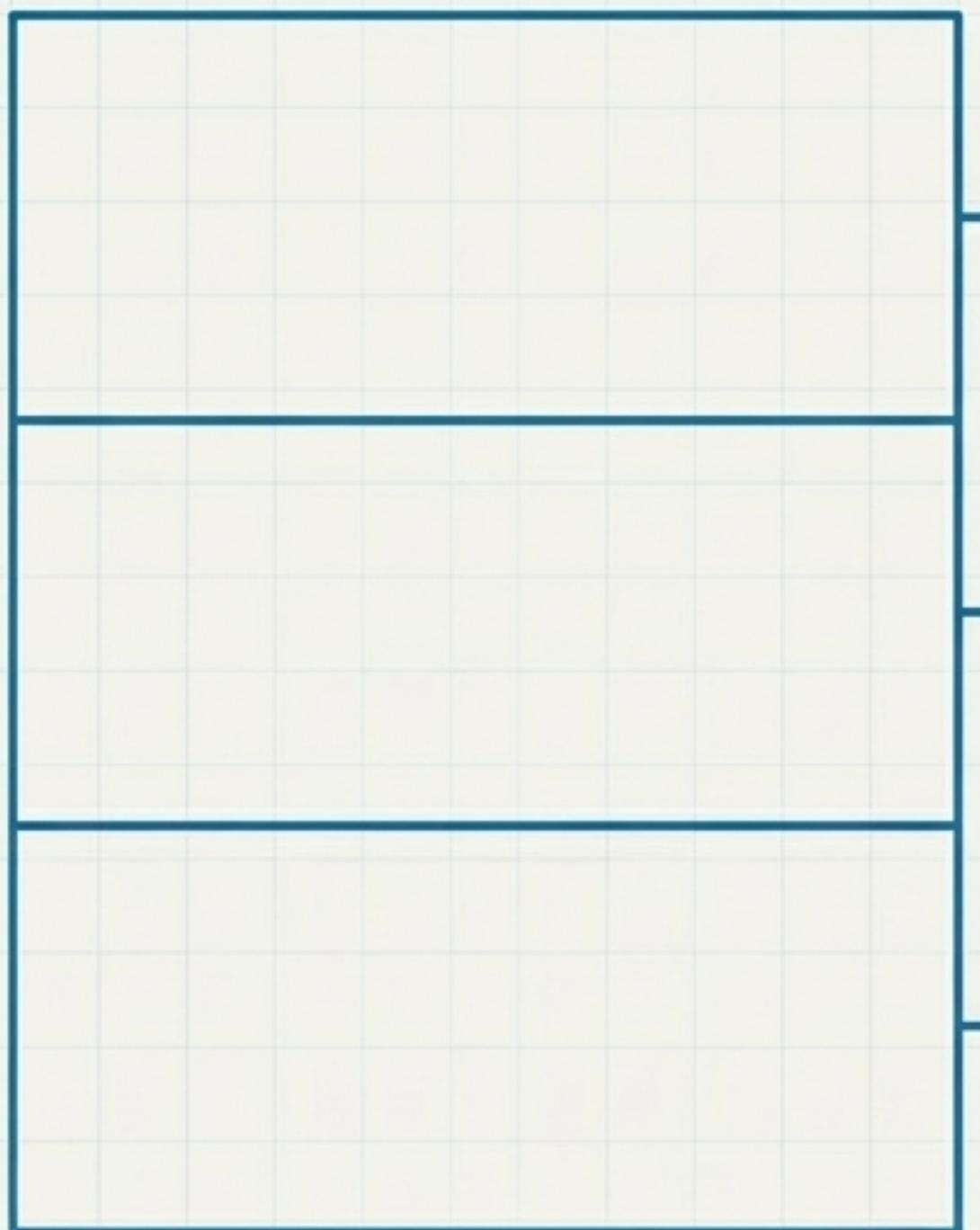
تغییر دادن یک خط در دیاگرام هزینه‌ای نزدیک به صفر دارد. UML اجازه می‌دهد اشتباهات معماري را **قبل از** نوشتن کد پیدا کنید.



زبان مشترک جهانی (Lingua Franca)

دیاگرام UML را مدیر پروژه، تحلیل‌گر سیستم، و برنامه‌نویسان زبان‌های مختلف (Java, Python, C#®, C++) درک می‌کنند.

کالبدشکافی دیاگرام کلاس: مهم‌ترین ابزار شما



طبقه اول: هويت (Name)

- نام کلاس با حروف **Bold** و **PascalCase** نوشته می‌شود.
- نام کلاس‌های *abstract* به صورت *italic* نوشته می‌شود.

طبقه دوم: ویژگی‌ها (Attributes/Fields)

این بخش "وضعیت" (State) یا "دانش" شئ را تعریف می‌کند.

طبقه سوم: عملیات (Operations/Methods)

این بخش "رفتار" (Behavior) شئ را تعریف می‌کند.

دستور زبان UML: سطح دسترسی (Visibility) (Visibility)

نماد UML	معادل جاوا	سطح دسترسی	توضیح کوتاه
	public	عمومی	قابل دسترسی برای تمام کلاس‌های سیستم.
	private	خصوصی	فقط قابل دسترسی در داخل خود کلاس.
	protected	محافظت شده	قابل دسترسی در پکیج و تمام زیرکلاس‌ها.
	(default)	پکیج	قابل دسترسی فقط برای کلاس‌های همان پکیج.

دستور زبان UML: تعریف ویژگی‌ها و عملیات

دستور زبان: فیلدّها

فرمول کلی: [Visibility] name : Type

مثال:

```
private String studentId;
```

```
- studentId : String
```

```
public String getFullName() { ... }
```

```
+ getFullName() : String
```

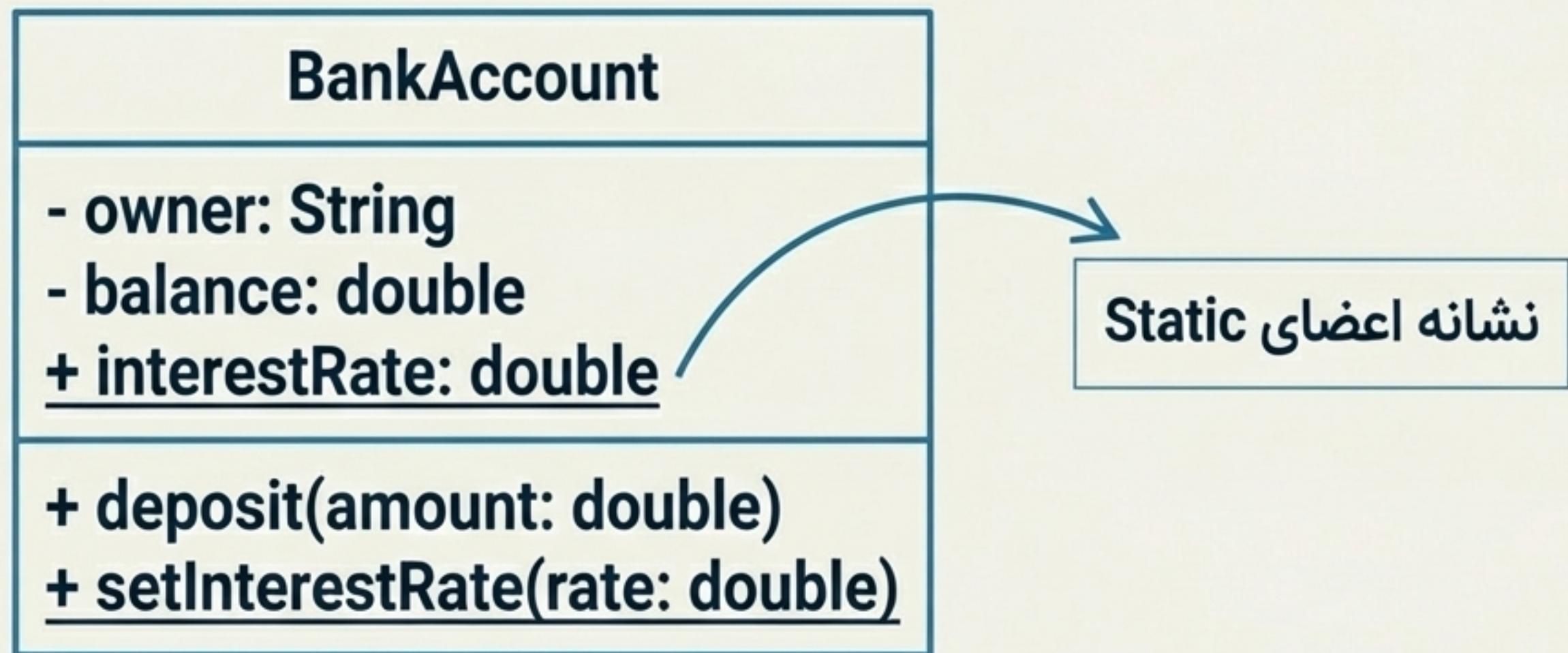
دستور زبان: متدها

فرمول کلی: [Visibility] name(paramName : ParamType) : ReturnType

مثال:

نمادهای خاص: اعضای استاتیک (Static)

برای نمایش فیلدها و متدهای آنها خط کشیده می‌شود. این اعضا متعلق به کلاس هستند، نه به یک شیء خاص.



کارگاه عملی (۱): طراحی کلاس دانشجو - از ایده تا نقشه

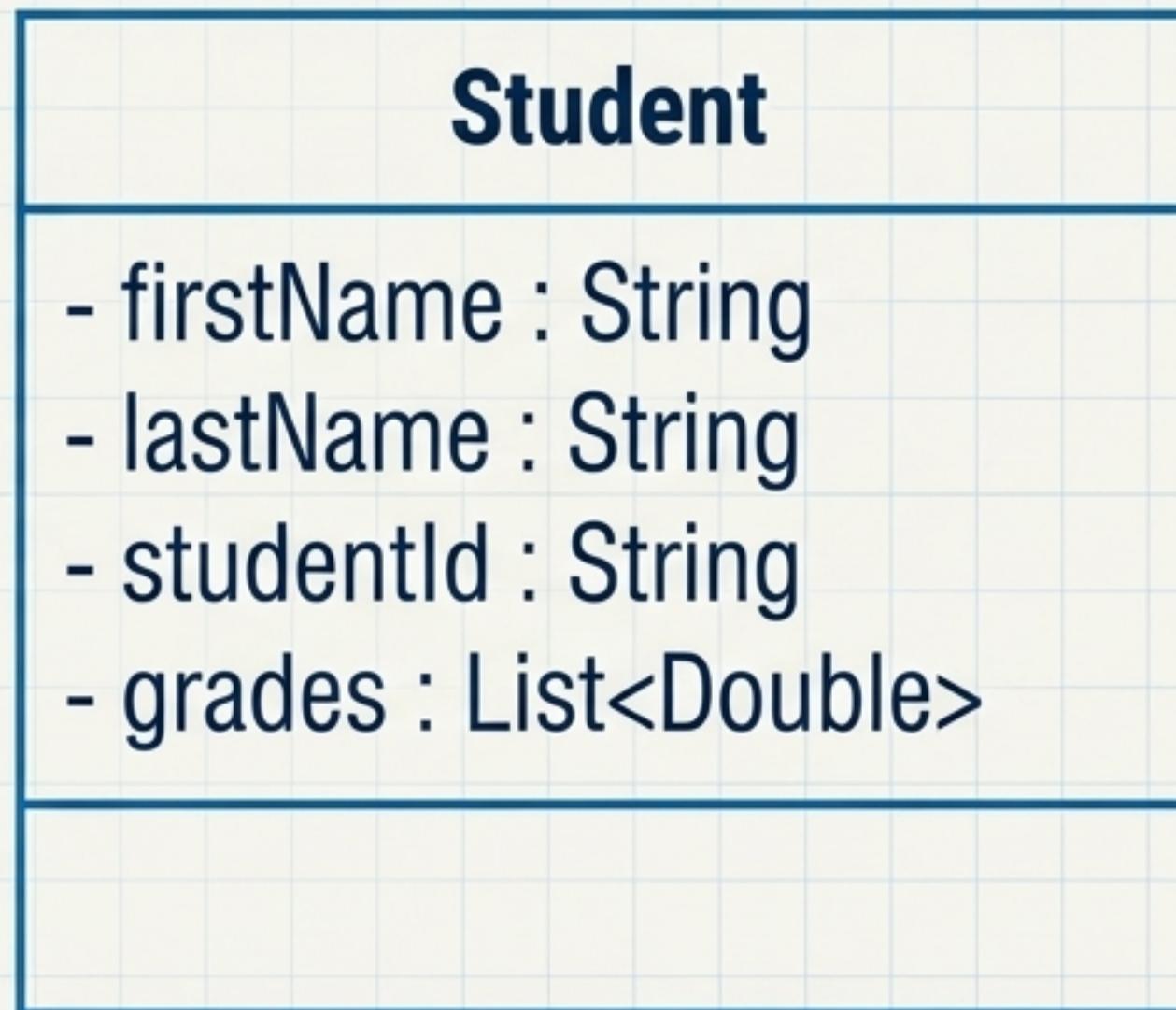
سناریو:

ما نیاز به کلاسی برای مدیریت اطلاعات دانشجویان داریم. هر دانشجو باید نام، نام خانوادگی، شماره دانشجویی و لیستی از نمرات داشته باشد.

گام ۱: هویت (Name)
یک مستطیل با نام `Student` در بالای آن.

گام ۲: ویژگی‌ها (Attributes)
طبق اصول کپسوله‌سازی، فیلد‌ها را به صورت `private` اضافه می‌کنیم:

- firstName : String •
- lastName : String •
- studentId : String •
- grades : List<Double> •



کارگاه عملی (۲): تکمیل نقشه با رفتارها

گام ۳: عملیات (Operations)

رفتارهای مورد نیاز کلاس را تعریف می‌کنیم:
سازنده:

`+ Student(firstName: String,
lastName: String, studentId: String)`

متدها:

- + getFullName() : String
- + addGrade(grade: double) : void
- + calculateGPA() : double

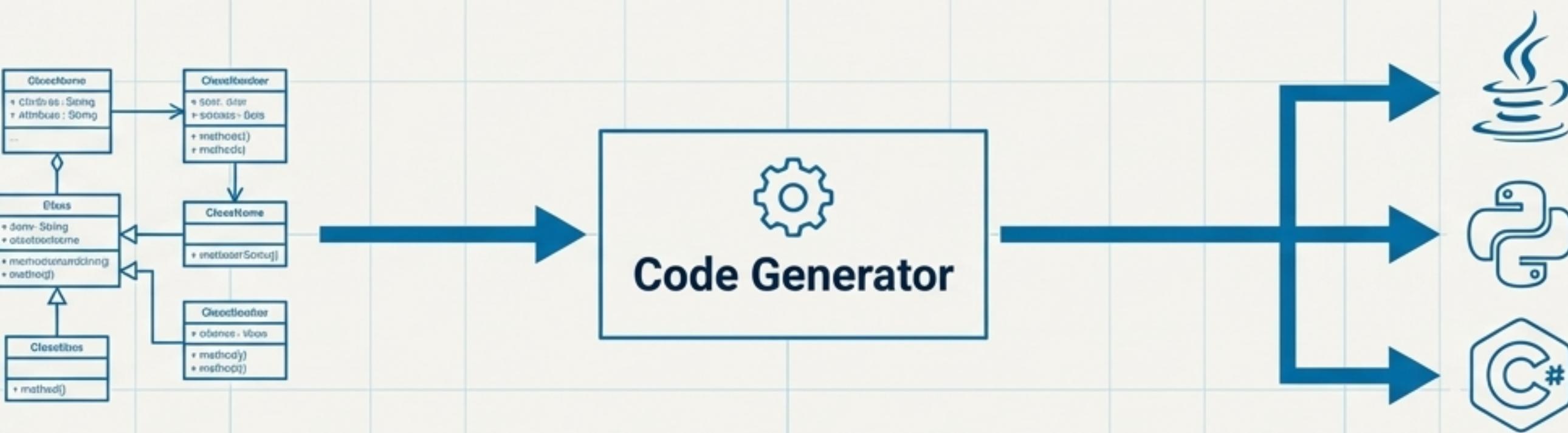
Student

- firstName: String
- lastName: String
- studentId: String
- grades: List~Double~

- + Student(firstName: String,
lastName: String, studentId: String)
- + getFullName(): String
- + addGrade(grade: double): void
- + calculateGPA(): double

(Model-Driven Development) تفکر مبتنی بر مدل

وقتی دیاگرام UML شما آنقدر دقیق و کامل است، خود دیاگرام به منبع اصلی حقیقت (Source of Truth) تبدیل می‌شود.

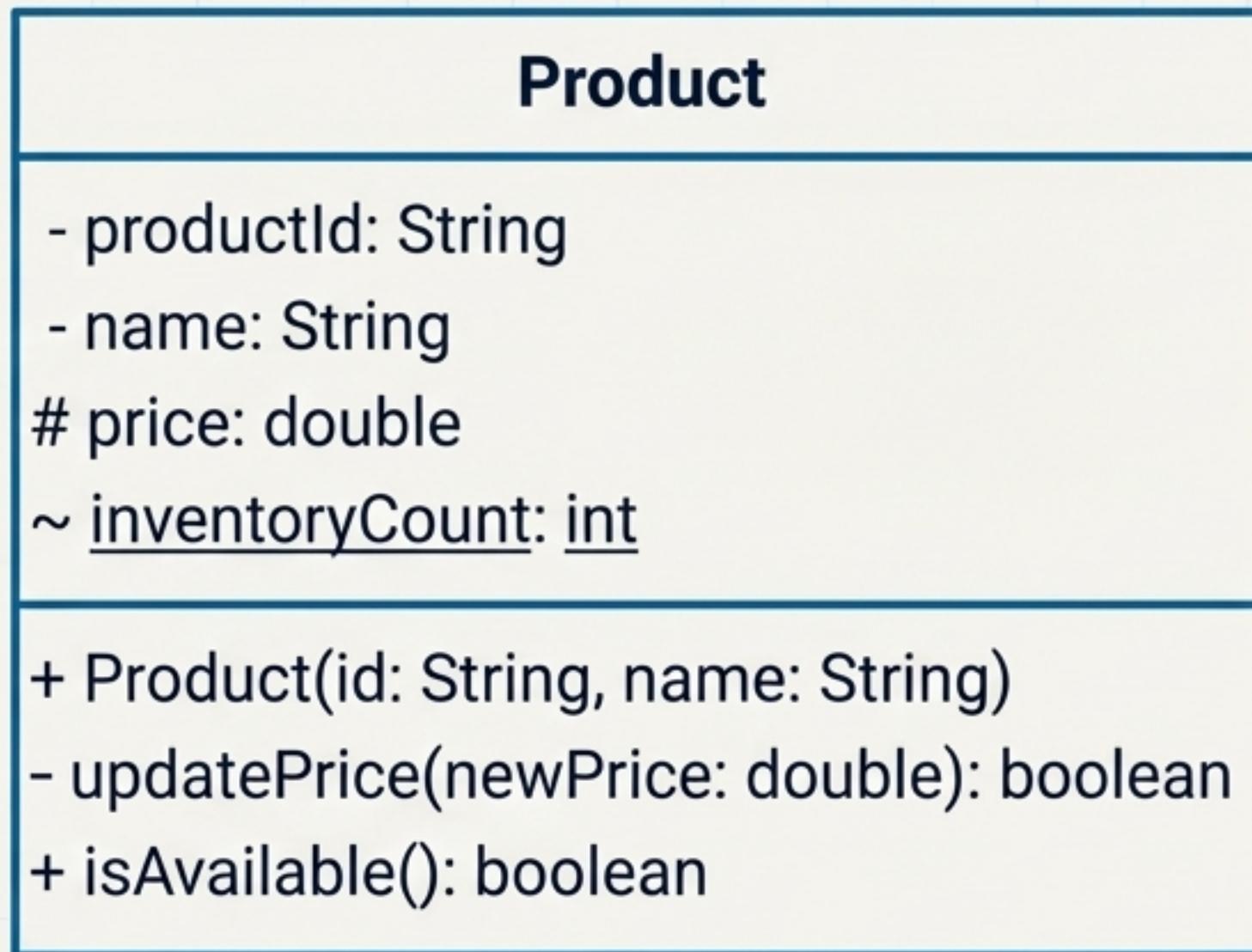


در این رویکرد:

- **مدل (UML)** مهمتر از کد است.
- تغییرات ابتدا در **مدل** اعمال می‌شوند.
- می‌توان به صورت خودکار از روی مدل، **اسکلت کد (Boilerplate Code)** را تولید کرد.
- **نتیجه:** هماهنگی بالا، کاهش خطای سرعت در توسعه.

آزمون اول: خواندن نقشه (Reading UML)

با توجه به دیاگرام زیر، به سوالات پاسخ دهید:



1. سطح دسترسی `price` چیست؟
2. کدام عضو `static` است؟ چگونه متوجه شدید؟
3. متدهای `updatePrice` چه نوع مقداری را برمی‌گرداند؟
4. سازنده این کلاس چند پارامتر ورودی دارد؟

آزمون دوم: مهندسی معکوس (Code to UML)

چالش: کد جاوای زیر را تحلیل کرده و دیاگرام کلاس استاندارد آن را رسم کنید.

```
public class SmartPhone {
    private String brand;
    public String model;
    protected int storageGB;
    private static int deviceCount = 0;

    public SmartPhone(String brand, String model) {
        this.brand = brand;
        this.model = model;
        deviceCount++;
    }

    public String getDeviceName() {
        return this.brand + " " + this.model;
    }

    public static int getDeviceCount() {
        return deviceCount;
    }
}
```

Your UML Diagram Here

جمع‌بندی: جعبه ابزار UML شما

Key Symbols

- + : public
- - : private
- # : protected
- ~ : package-private
(default)

Key Notations

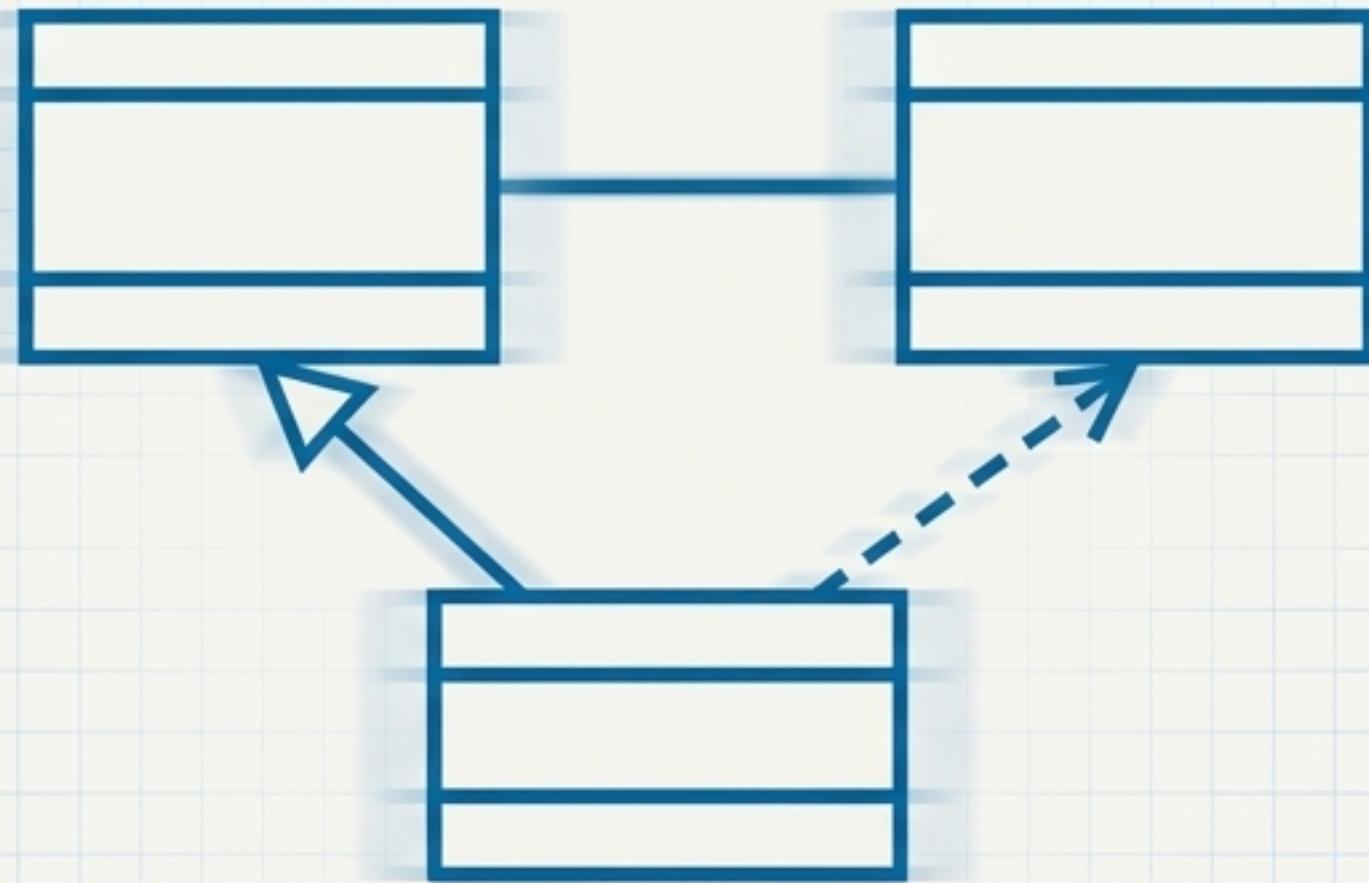
- Underline : static
- `name : Type` : برای فیلدها و متغیرها
- `method()` : برای **ReturnType**` : برای متدها

Core Principle

همیشه قبل از کدنویسی، طراحی کنید. نقشه شما UML است.

پایان بخش الف: الفبای مهندسی

شما اکنون می‌توانید "کلمات" زبان UML را بخوانید و بنویسید (یعنی یک کلاس را مدل کنید).



**در بخش بعدی (۸-ب):

- یاد می‌گیریم چگونه با استفاده از خطوط و فلش‌ها، "جملات" و "داستان‌های" کامل بسازیم.
- روابط بین کلاس‌ها را مدل‌سازی خواهیم کرد (Association). (Association, Inheritance, Aggregation,
- قدرت واقعی OOP در "تعامل" اشیاء آشکار می‌شود.