# Introduction to Programming

Rania Baghernejad

## Java – part 10

## Static & Enum

---

### مباحث

### کلاس‌ها و متغیرهای سراسری (Class-level variables)

وقتی شما برنامه‌ای می‌نویسید که می‌خواهید اطلاعات را ذخیره کنید و این اطلاعات را در قسمت‌های مختلف برنامه استفاده کنید، ممکن است بخواهید این اطلاعات را در **کلاس‌ها** قرار دهید. در واقع، **کلاس‌ها** مکان‌هایی برای ذخیره اطلاعات و انجام عملیات هستند.

در ابتدا، ممکن است شما یک **متغیر** در داخل یک **کلاس** تعریف کنید. این متغیر فقط برای یک شیء از آن کلاس قابل دسترسی خواهد بود. اگر بخواهید که یک متغیر قابل دسترسی برای **تمام اشیاء از آن کلاس** باشد (یعنی تمامی اشیاء بتوانند به آن متغیر دسترسی داشته باشند)، باید از **static** استفاده کنید. (به مفهوم شئ در مباحث بعدی مفصل خواهیم پرداخت.)

در مثالی ساده‌تر، فرض کنید یک کلاس مثل یک دانشگاه است. **متغیرهای معمولی** مثل دانشجویان هستند که هر کدام اطلاعات شخصی خودشان را دارند (مثلاً نام، شماره دانشجویی). **متغیر static** مثل یک تابلوی اعلانات مرکزی دانشگاه است که همه دانشجویان می‌توانند آن را ببینند و تغییر دهند.

---

### Static

در صورتی که بخواهید یک **متغیر یا متد** تنها یکبار برای کل کلاس تعریف شود و تمام اشیاء از آن کلاس از آن استفاده کنند، می‌توانید از کلمه‌کلیدی **static** استفاده کنید. (زمانی که ما مشخص نمی‌کنیم که این متغیر public باشد یا private به صورت پیش فرض public درنظر گرفته می‌شود.)

به عبارت ساده‌تر، اگر متغیر یا متدی به‌صورت **static** تعریف شود، این ویژگی **مربوط به خود کلاس است و نه به هر شئ خاص از آن کلاس.**

**مثال:**

```
class Counter {
    static int count = 0;  // Static variable, accessible throughout the class

    static void increment() {
        count++;  // Static method to increment the static variable
    }
}

public class Main {
    public static void main(String[] args) {
        Counter.increment();
        Counter.increment();
        System.out.println("Count: " + Counter.count);  // Output: Count: 2
```

```
    }
}
```

- در این مثال، count یک **متغیر سراسری** است که می‌تواند توسط تمامی اشیاء از کلاس Counter استفاده شود.
- با استفاده از **static**، این متغیر به‌جای اینکه برای هر شیء جدید از کلاس Counter مقداردهی شود، **برای کل کلاس یکسان باقی می‌ماند**.

---

### static final چیست؟

وقتی یک متغیر هم static باشد و هم final، یعنی:

1. static ⇒ متغیر به کلاس تعلق دارد (نه به شیء خاصی).
2. final ⇒ فقط یک بار مقدار می‌گیرد و بعدش قابل تغییر نیست.

یعنی این متغیر فقط یک‌بار مقداردهی می‌شود و برای همه استفاده‌کنندگان از کلاس، مشترک است.

**مثال:**

```
public class MathConstants {
    public static final double PI = 3.14159;
}
```

**نحوه استفاده در متد main:**

```
public class Main {
    public static void main(String[] args) {
        System.out.println(MathConstants.PI); // دسترسی مستقیم
    }
}
```

---

### private static final چیست؟

- private ⇒ یعنی این متغیر فقط داخل همان کلاس قابل استفاده‌ست، نه بیرون از آن.

**مثال:**

```
public class AppConfig {
    private static final String VERSION = "1.0.0";
```

```
    public static void printVersion() {
        System.out.println("App Version: " + VERSION);
    }
}
```

**نحوه استفاده:**

```
public class Main {
    public static void main(String[] args) {
        // System.out.println(AppConfig.VERSION); // Error: VERSION is private
        AppConfig.printVersion();   // مجاز است چون از طریق متد پابلیک فراخوانی شده
    }
}
```

---

## Enum

در زبان جاوا، **enum** به‌عنوان یک نوع داده‌ای است که مقادیر ثابت را تعریف می‌کند. این مقادیر ثابت به‌طور معمول در سراسر برنامه استفاده می‌شوند و به‌دلیل ویژگی‌های مشابه **static**، می‌توانند از هر نقطه‌ای در برنامه مورد استفاده قرار گیرند.

در واقع، **enum** مجموعه‌ای از **مقادیر ثابت** است که **یک کلاس ویژه به‌طور پیش‌فرض برای آن‌ها ایجاد می‌کند.**

فرض کنید یک رستوران دارید و می‌خواهید منوی غذاها را تعریف کنید.

enum مثل یک **لیست ثابت** از غذاها است که همیشه همان گزینه‌ها را دارد:

```
enum Menu { PIZZA, BURGER, SALAD }
```

نمی‌توانید به این لیست در زمان اجرا غذای جدید اضافه کنید مثلاً PASTA ، چون از قبل تعریف شده است. این لیست در تمام برنامه یکسان است و همه می‌توانند از آن استفاده کنند.

---

**مثال:**

```
enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

public class Main {
    public static void main(String[] args) {
        Day today = Day.MONDAY;
        System.out.println("Today is: " + today);  // Output: Today is: MONDAY
    }
}
```

- در اینجا، Day اسم enum هست، مثل یه نوع داده جدید  است که مقادیر ثابت مربوط به روزهای هفته را تعریف می‌کند.
- این مقادیر ثابت از نوع **Day** هستند و می‌توانند در هر کجا از برنامه به‌راحتی استفاده شوند.
- today یه متغیره از نوع Day.
- مقدار today رو با Day.MONDAY تنظیم کردیم.

---

**نتیجه‌گیری:**

در این بخش، مفهوم **static** و **enum**را با توجه به اینکه این مفاهیم در حقیقت مربوط به اطلاعات و عملیات‌های یک کلاس هستند، معرفی کردیم. برای **static**، نشان دادیم که این متغیرها و متدها برای تمامی اشیاء از آن کلاس قابل دسترسی هستند. برای **enum** نیز، توضیح دادیم که مقادیر ثابت ایجاد شده در یک **enum** مشابه به **static** قابل استفاده در سراسر برنامه هستند.