

بخش ۱۸: انواع شمارشی (Enums) – قدرت ثابت‌های امن

تهییه شده توسط: سید سجاد پیراهاش

کابوس اعداد جادویی: روش قدیمی و خطرناک

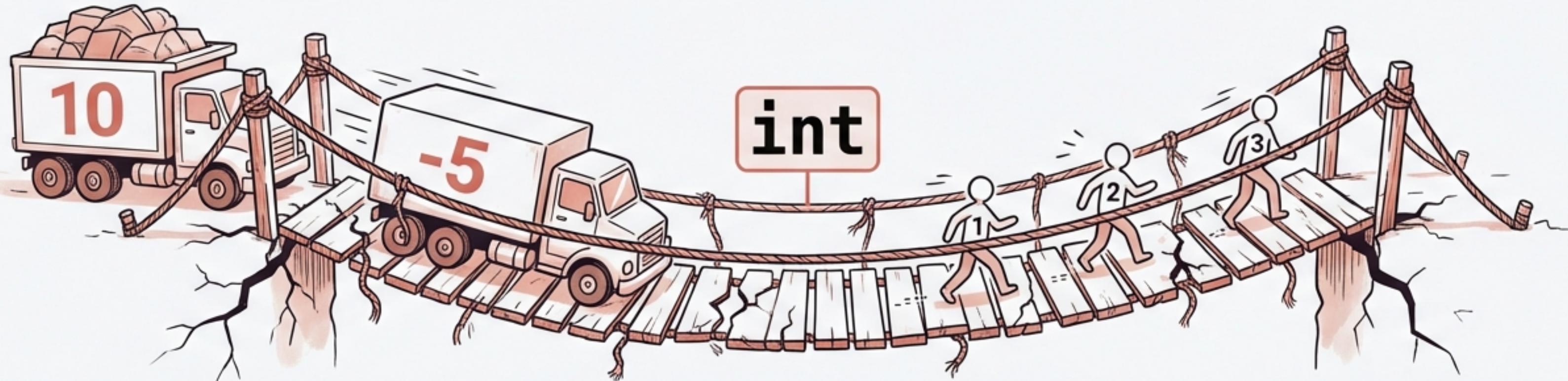
`public static final int`

```
public class OldConstants {  
    public static final int MONDAY = 1;  
    public static final int TUESDAY = 2;  
    // ... and so on  
}  
  
public void processDay(int day) {  
    // What if day is 10 or -5?  
}
```

عدم امنیت نوع (Type-Unsafe): متد `processDay` هر عدد صحیحی را می‌پذیرد، حتی مقادیر بی‌معنی!

خوانایی ضعیف (Poor Readability): چاپ کردن متغیر `day`، عدد `1` را نشان می‌دهد، نه "MONDAY".

شکنندگی (Fragile): تغییر مقدار یک ثابت می‌تواند به طور خاموش باعث بروز باگ در سایر نقاط برنامه شود.



معرفی قهرمان: `enum`، نگهبان منطق برنامه

راه حل مدرن `enum`

```
public enum Day {  
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY,  
    FRIDAY, SATURDAY, SUNDAY;  
}  
  
public void processDay(Day day) {  
    // Only valid Day constants are allowed!  
}
```

امنیت نوع کامل (Type-Safe): کامپایلر تضمین می‌کند که فقط مقادیر تعریف شده در `Day` قابل استفاده هستند.

خوانایی فوق العاده (Highly Readable): متغیر `day` خود را نمایندگی و چاپ می‌کند.

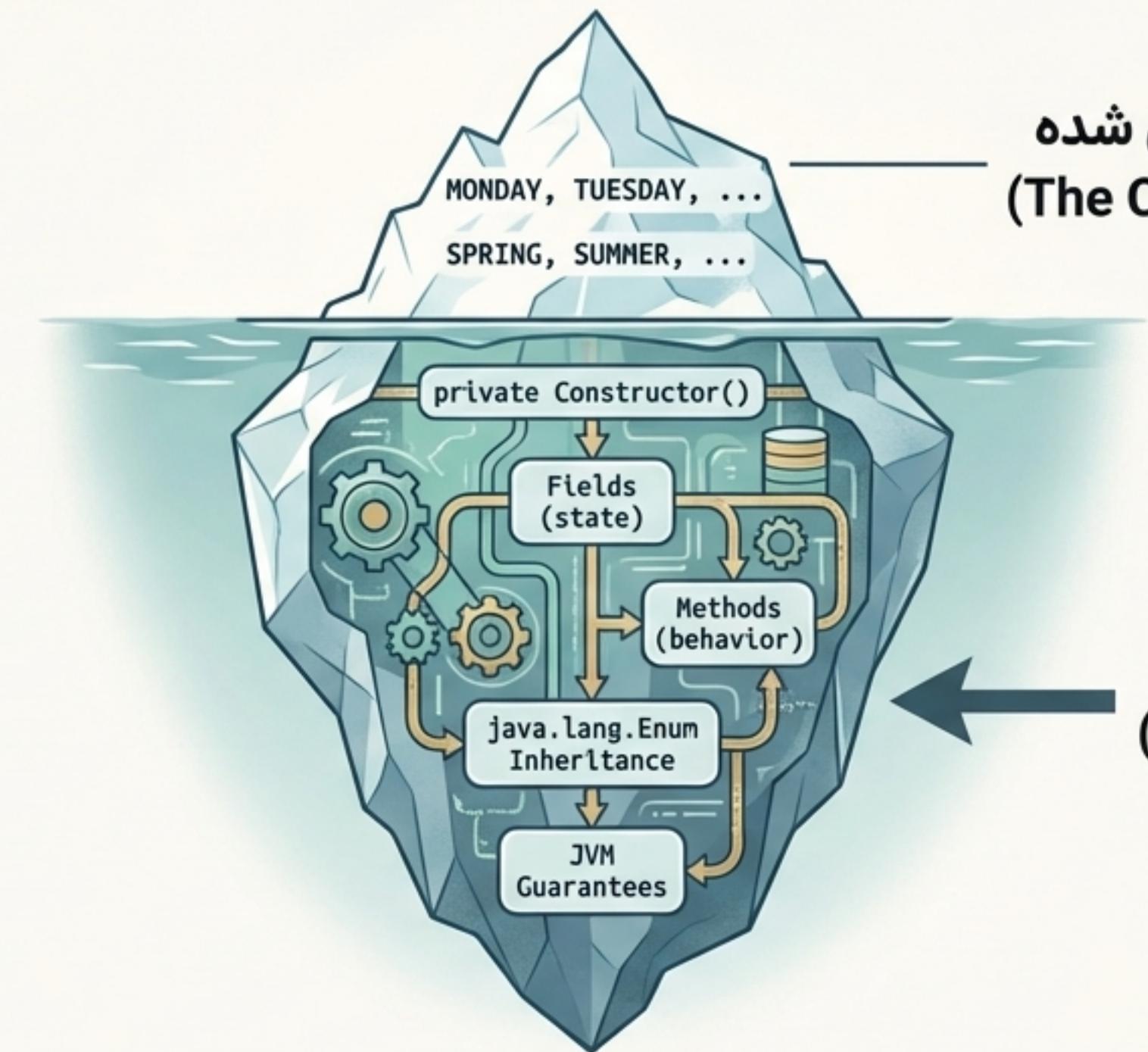
استحکام و پایداری (Robust): هر ثابت یک شیء منحصر به فرد است و با مقادیر دیگر اشتباه گرفته نمی‌شود.



مقایسه رو در رو: `enum` در برابر `static final int`

معیار	`public static final int`	`enum`
امنیت نوع (Type Safety)	✗ ندارد. هر مقدار `int` قابل قبول است.	✓ کامل. فقط نمونه‌های تعریف شده مجاز هستند.
خوانایی (Readability)	▼ ضعیف. مقادیر چاپ شده (مثلاً `(1)` بی‌معنی هستند.	▲ عالی. مقادیر چاپ شده (مثلاً `MONDAY`) خوانا هستند.
افزودن رفتار (Flexibility)	✗ غیرممکن. نمی‌توان به `int` متده اضافه کرد.	✓ قدرتمند. می‌تواند فیلد، سازنده و متده داشته باشد.
استفاده در `switch`	⚠ خطرناک. نیازمند `default case` برای مقادیر نامعتبر.	✓ امن. کامپایلر برای `case`‌های فراموش شده هشدار می‌دهد.
قابلیت پیمایش (Iterable)	✗ دشوار. باید لیست را دستی نگهداری کنید.	✓ آسان. متده `values()` تمام ثابت‌ها را برمی‌گرداند.

فراتر از یک لیست ساده: `enum` به عنوان یک کلاس قدرتمند



ثابت‌های نامگذاری شده
(The Constants You See)

قدرت پنهان یک کلاس واقعی
(The Hidden Power of a Class)

یک `enum` در جاوا فقط یک لیست از اسامی نیست. این یک کلاس ویژه است که می‌تواند حالت (Fields) و رفتار (Methods) داشته باشد، درست مانند هر کلاس دیگری.

کالبدشکافی یک `enum` هوشمند: افزودن فیلد، سازنده و متدها

```
public enum Season {  
    // Each constant is an object initialized via the constructor  
    SPRING(15),  
    SUMMER(30),  
    AUTUMN(18),  
    WINTER(-5);  
  
    // 1. فیلد (Field) برای نگهداری داده  
    private final int averageTemp; ← 1. فیلد  
  
    // 2 (Constructor) - سازنده است  
    private Season(int temp) { ← 2. سازنده (همیشه private)  
        this.averageTemp = temp;  
    }  
  
    // 3. متدها (Method) برای تعریف کار  
    public int getAverageTemp() { ← 3. متدها  
        return this.averageTemp;  
    }  
  
    public void printExpectedActivity() {  
        // ... logic based on season ...  
    }  
}
```

کنترل جریان کد به شکلی خوانا و امن با `switch`



با `int` `switch` (خطرنگ)

```
int day = 2; // TUESDAY
switch (day) {
    case 1: // MONDAY
        System.out.println("Start of the week.");
        break;
    // ...
    default:
        System.out.println("Invalid day!");
    // Needed for safety
}
```



با `enum` `switch` (امن و خوانا)

```
Day day = Day.TUESDAY;
switch (day) {
    case MONDAY:
        System.out.println("Start of the week.");
        break;
    case TUESDAY:
        System.out.println("Team meeting day.");
        break;
    // No need for Day.MONDAY
}
```



کامپایلر جاوا به شما هشدار می‌دهد اگر یک `case` برای یکی از مقادیر `enum` را فراموش کرده باشید! این یک تور ایمنی بزرگ است.

ابزارهای داخلی: پیمایش و تبدیل با `values()` و `valueOf()

- پیمایش تمام ثابت‌ها `values()

متدهای آرایه‌ای از تمام ثابت‌های enum را به ترتیب تعریف‌شان برمی‌گرداند.

```
// Print all seasons and their average temperatures
for (Season s : Season.values()) {
    System.out.printf("%s has an average temp of %d°C\n",
                      s, s.getAverageTemp());
}
```

- تبدیل رشته به ثابت `valueOf()

متدهای valueOf() یک ثابت enum را بر اساس نام رشته‌ای آن پیدا می‌کند.

```
String seasonName = "SUMMER";
Season summer = Season.valueOf(seasonName);

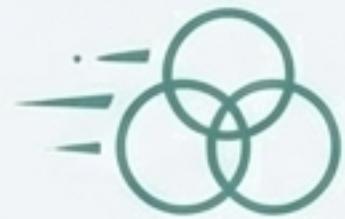
System.out.println(summer); // Prints: SUMMER
```

توجه: اگر رشته ورودی با هیچ ثابتی مطابقت نداشته باشد، یک IllegalArgumentException پرتاب می‌شود.



ابزارهای حرفه‌ای: کلکسیون‌های بهینه و `EnumMap` و `EnumSet`

برای کار با مجموعه‌ها یا مپ‌هایی از `enum`‌ها، همیشه از پیاده‌سازی‌های تخصصی و بسیار بهینه استفاده کنید.



`EnumSet`

یک پیاده‌سازی `Set` بسیار سریع و کم‌صرف که از بیت‌ها برای نگهداری مقادیر استفاده می‌کند.

به جای `HashSet<Day>` استفاده شود.

```
Set<Day> weekend =  
    EnumSet.of(Day.SATURDAY, Day.SUNDAY);
```



`EnumMap`

یک پیاده‌سازی `Map` با عملکرد فوق‌العاده بالا که کلیدهای آن از یک نوع `enum` هستند.

به جای `HashMap<Day, String>` استفاده شود.

```
Map<Day, String> tasks =  
    new EnumMap<>(Day.class);
```

چرا بهینه‌تر هستند؟ چون از یک آرایه داخلی ساده به جای ساختارهای داده پیچیده استفاده می‌کنند و محاسبات هش‌کد ندارند.

راز متخصصان: `enum` به عنوان بهترین پیاده‌سازی الگوی Singleton

چگونه می‌توان تضمین کرد که از یک کلاس فقط و فقط یک نمونه (instance) در کل برنامه وجود داشته باشد؟

The Enum Solution

یک `enum` با یک عضو، ساده‌ترین و امن‌ترین راه برای پیاده‌سازی الگوی Singleton در جاوا است.

```
public enum TheOneSingleton {  
    INSTANCE; // The one and only instance  
  
    public void doSomething() {  
        System.out.println("Doing something...");  
    }  
  
    // Usage:  
    TheOneSingleton.INSTANCE.doSomething();
```

امنیت در برابر **Serialization**: جاوا به طور خودکار از ساخته شدن نمونه‌های تکراری هنگام جلوگیری می‌کند.

امنیت در برابر **Reflection**: محافظت شده در برابر حملات reflection که می‌توانند سازنده‌های private را فراخوانی کنند.

садگی و خوانایی: کد بسیار کوتاه و گویا است.

نکات کلیدی و اشتباهات رایج



برای مقایسه enum ها، استفاده از `==` امن، صحیح و سریع‌تر از `equals()` است.
JVM تضمین می‌کند که هر ثابت enum فقط یک نمونه در حافظه دارد.



یک enum نمی‌تواند کلاسی را `extends` کند (چون به طور ضمنی از `Object` ارث می‌برد)، اما می‌تواند هر تعداد `interface` را `implements` کند.



همیشه `HashMap` و `HashSet` را به جای `EnumMap` و `EnumSet` برای کار با enum ها ترجیح دهید تا از حداکثر کارایی بهره‌مند شوید.

نوبت شما: دانش خود را به کار بگیرید

تمرین‌های عملی برای تسلط بر قدرت Enums



چالش ۱ و ۲: ساخت و غنیسازی `enum` فصل‌ها



چالش ۱: `enum` برای فصول
یک `enum` به نام `Season` با چهار مقدار `SPRING`, `SUMMER`, `AUTUMN`, `WINTER` بسازید.



چالش ۲: `enum` با فیلد و متد

تمرين ۱ را طوری تغيير دهيد که هر فصل يک دمای ميانگين (مثلاً int averageTemp) داشته باشد.

- يک فیلد private final int averageTemp اضافه کنید.
- يک سازنده private مقداردهی اوليه این فیلد بنویسید.
- يک متد () public int getAverageTemp() به آن تعریف کنید.

چالش ۳ و ۴: منطق `switch` و بازنویسی کد قدیمی

چالش ۳: `enum` در `switch`

یک متده بنویسید که یک `Season` را به عنوان ورودی گرفته و با استفاده از `switch`، یک فعالیت پیشنهادی برای آن فصل را به صورت `String` برگرداند.

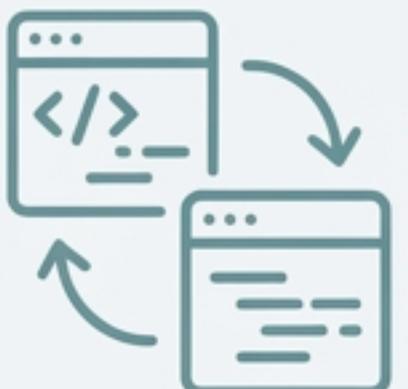


مثال:

`WINTER` → "Skiing"
`SUMMER` → "Swimming"

چالش ۴: ریفکتور کردن `static final`

کد زیر که از ثابت‌های `int` برای کدهای وضعیت HTTP استفاده می‌کند را با استفاده از `enum` بازنویسی کنید.



کد قدیمی:

```
public class HttpStatusCodes {  
    public static final int OK = 200;  
    public static final int NOT_FOUND = 404;  
    public static final int INTERNAL_SERVER_ERROR = 500;  
}
```

راهنمایی: یک `enum` با فیلد `code` و یک سازنده بسازید.

خلاصه بخش هجدهم: قدرت Enums در دستان شما

- امنیت نوع: enum با حذف "اعداد جادویی"، امنیت نوع (Type Safety) را به کد شما هدیه می‌دهد.
- قدرت کلاس: enum ها کلاس‌هایی واقعی با قابلیت داشتن فیلد، سازنده و متدهستند.
- خوانایی و استحکام: کد شما با enum خواناتر، امن‌تر و نگهداری آن آسان‌تر می‌شود، به خصوص در بلوک‌های switch.
- ابزارهای داخلی: متدهای values() و valueOf() ابزارهای قدرتمندی برای پیمایش و تبدیل هستند.
- کارایی بالا: برای عملکرد بهینه، همیشه از EnumMap و EnumSet استفاده کنید.
- کارایی بالا: برای عملکرد بهینه، همیشه از EnumMap و EnumSet استفاده کنید.
- فراتر از ثابت‌ها: enum ها می‌توانند الگوهای طراحی پیشرفته‌ای Singleton را به بهترین شکل پیاده‌سازی کنند.