دومین موضوعی که در برنامه نویسی امن با زبان جاوا مورد توجه قرار می‌گیرد مربوط به اعلان‌ها و مقدار دهی اولیه است. در ادامه به اولین قانون (DCL00-J) ذیل این موضوع پرداخته خواهد شد.

در بخش اول مقاله، این قانون توضیح داده شده و به نمونه کد ناسازگار با قانون و راه حل آن اشاره می‌شود. در بخش بعدی مقاله، دو نمونه دیگر از کد ناسازگار خواهد آمد.

قانون DCL00-J – جلوگیری از دورهای مقدار دهی اولیه کلاس

با توجه به *مشخصات زبان جاوا* (JLS) (مشخصه شماره ۱۲,۴ "مقدار دهی اولیه کلاس‌ها و واسطها"):

مقدار دهی اولیه کلاس شامل اجرای تخصیص دهنده‌های استاتیک آن و اجرای تخصیص دهنده‌ها برای فیلدهای استاتیک تعریف شده در کلاس (متغیرهای کلاس) می‌باشد.

بنابراین، وجود فیلد استاتیک مقدار دهی اولیه کلاس را فعال می‌کند. در عین حال، تخصیص دهنده فیلد استاتیک می‌تواند مبتنی بر مقدار دهی اولیه کلاسی دیگر باشد و این کار احتمالاً دور مقدار دهی اولیه ایجاد می‌کند.

علاوه بر این، در مشخصه شماره ۸,۳,۲,۱ "تخصیص دهنده‌ها برای متغیرهای کلاس" آمده است:

در زمان اجرا، ابتدا متغیرهای استاتیک که final بوده و با مقادیر ثابت زمان کامپایل مقدار دهی اولیه شده‌اند مقدار دهی اولیه می‌شوند.

این عبارت درخصوص نمونه‌هایی که از مقادیر فیلدهای استاتیک final استفاده می‌کنند که بعداً مقدار دهی اولیه می‌شوند، صادق نیست. اعلان فیلد به صورت استاتیک final تضمین کننده مقداردهی اولیه آن قبل از خوانده شدن نیست.

به طور کلی، پیشنهاد می‌شود برنامه‌ها – و برنامه‌های حساس به امنیت باید– تمام دورهای مقدار دهی اولیه کلاس را حذف کنند.

یک نمونه ناسازگار با قانون (دور درون کلاسی)

در این مثال دور مقدار دهی اولیه درون کلاسی وجود دارد:

```
public class Cycle {
  private final int balance;
  private static final Cycle c = new Cycle();
  private static final int deposit = (int) (Math.random() * 100); // Random deposit

  public Cycle() {
    balance = deposit - 10; // Subtract processing fee
  }

  public static void main(String[] args) {
    System.out.println("The account balance is: " + c.balance);
  }
}
```

کلاس Cycle در مثال فوق دارای متغیر خصوصی استاتیک final است که این متغیر به نمونه جدیدی از کلاس Cycle مقدار دهی اولیه می‌شود. تخصیص دهنده‌های استاتیک حتماً قبل از اولین استفاده از عضو استاتیک کلاس یا اولین فراخوانی سازنده، یک بار فراخوانی می‌شود.

هدف برنامه نویس، محاسبه مانده حساب از طریق کسر هزینه پردازش از مقدار سپرده است. اما مقدار دهی اولیه متغیر c (که از نوع کلاس است) قبل از مقداردهی اولیه فیلد deposit در زمان اجرا انجام می‌شود، چراکه ظاهراً این مقدار دهی قبل از مقدار دهی اولیه فیلد depositاست.

در نتیجه، مقدار deposit دیده شده توسط سازنده –در زمان فراخوانی درحین مقدار دهی اولیه متغیر استاتیک c– به جای مقدار تصادفی، دارای مقدار اولیه صفر است. به همین خاطر، مقدار balance همیشه برابر با ۱۰- محاسبه می‌شود.

مرحله سوم رویه مقدار دهی اولیه که در JLS شماره ۱۲,۴,۲ شرح داده شده است، امکان پیاده‌سازی بدون روبرو شدن با چنین دورهای بازگشتی مقدار دهی اولیه را فراهم می‌کند.

راه حل سازگار با قانون (دور درون کلاسی)

این راه حل سازگاری، ترتیب مقداردهی اولیه کلاس Cycle را به گونه ای تغییر می‌دهد که فیلدها بدون ایجاد هرگونه دورهای وابستگی مقدار دهی اولیه شوند. مقدار دهی اولیه c بعد از مقداردهی اولیه deposit قرار گرفته است؛ در نتیجه مقداردهی اولیه آن، بعد از مقداردهی اولیه کامل به deposit خواهد بود.

```
public class Cycle {
  private final int balance;
  private static final int deposit = (int) (Math.random() * 100); // Random deposit
  private static final Cycle c = new Cycle();  // Inserted after initialization of required fields
  public Cycle() {
    balance = deposit - 10; // Subtract processing fee
  }

  public static void main(String[] args) {
    System.out.println("The account balance is: " + c.balance);
  }
}
```

چنین دورهای مقداردهی اولیه زمانی که تعداد فیلدها زیاد باشد، مشکل ساز خواهد بود. بنابراین به طریقی باید از فقدان چنین دورهایی در جریان کنترلی اطمینان حاصل کرد.

اگرچه این راه حل سازگاری از دور مقداردهی اولیه جلوگیری می‌کند، اما وابسته به ترتیب اعلان بوده و راه حل مطمئنی نیست، چراکه ممکن است کسانی که مسئول نگهداری نرم افزار هستند از اهمیت این ترتیب که برای حفظ صحت برنامه است اطلاع نداشته باشند. به همین خاطر چنین مواردی حتماً باید صریحاً در کد مستند شوند.

در بخش بعدی مقاله، به دو نمونه دیگر از کدهای ناسازگار با قانون و راه‌حل‌های سازگاری آن با قانون خواهیم پرداخت.