دومین موضوعی که در برنامه نویسی امن با زبان جاوا مورد توجه قرار می‌گیرد اعلان‌ها و مقدار دهی اولیه است. در ادامه به دومین قانون (DCL01-J) ذیل این موضوع پرداخته خواهد شد.

قانون DCL01-J – عدم استفاده مجدد از شناسه‌های عمومی کتابخانه استاندارد جاوا

از اسامی مرتبط با شناسه‌های عمومی نمایان، کلاس‌های ابزار عمومی، واسط‌ها یا بسته‌ها در کتابخانه استاندارد جاوا استفاده مجدد نکنید.

زمانیکه توسعه دهنده نرم افزار از شناسه‌ای که نامی یکسان با کلاس عمومی مانند Vector دارد استفاده می‌کند، پشتیبانی کننده نرم افزار ممکن است از عدم تعلق این شناسه به java.util.Vector آگاه نبوده و ناخواسته از نوع Vector سفارشی شده به جای کلاس java.util.Vector استفاده کند. نوع سفارشی شده Vector می‌تواند سایه کلاس java.util.Vector بوده و منجر به بروز رفتار ناخواسته در برنامه شود.

توضیح عبارات ورود ساده می‌تواند چنین مواردی را برطرف کند. با این وجود، زمانی که از تعاریف نام هایی که توسط بسته های دیگر وارد شده است استفاده مجدد می شود، استفاده از اعلان type-import-on-demand declaration می تواند تصمیم گیری برنامه نویس برای استفاده از تعریف خاصی که مورد نظر است را پیچیده کند. علاوه بر این، کار متداولی که می‌تواند منجر به خطاهایی شود، ایجاد عبارات import بعد از نوشتن کد برنامه (و اغلب از طریق شمول عبارات import بوسیله IDE) است که با توجه به نام‌ها ابهام بیشتری ایجاد می‌کند. زمانی که نوع سفارشی زودتر از نوع درنظر گرفته شده در جاوا بیآید، جستجوهای بعدی انجام نخواهد شد. در نتیجه، نوع اشتباه مورد قبول قرار می‌گیرد.

یک نمونه ناسازگار با قانون (نام کلاس)

در این مثال، از نام کلاس java.util.Vector استفاده مجدد شده است. این کلاس شرط متفاوتی را برای متد ( ) isEmpty تعریف کرده است. پشتیبانی کننده نرم افزار ممکن است متد ( ) isEmpty را با متد ( ) java.util.Vector.isEmpty اشتباه گرفته و به همین خاطر رفتار ناخواسته‌ای از برنامه بروز کند.

```
class Vector {
  private int val = 1;

  public boolean isEmpty() {
    if (val == 1) {   // Compares with 1 instead of 0
      return true;
    } else {
      return false;
    }
  }
  // Other functionality is same as java.util.Vector
}

// import java.util.Vector; omitted
public class VectorUser {
  public static void main(String[] args) {
    Vector v = new Vector();
    if (v.isEmpty()) {
      System.out.println("Vector is empty");
    }
  }
}
```

راه حل سازگار با قانون (نام کلاس)

این راه حل سازگاری، از نامی دیگر برای کلاس استفاده می‌کند تا از اشتباه شدن احتمالی کلاس کتابخانه استاندارد جاوا، با این کلاس جلوگیری کند.

```
class MyVector {
  //Other code
}
```

زمانی که توسعه دهنده، کلاس پنهان شده اصلی را کنترل می کند، ممکن است تغییر استراتژی طراحی با توجه به مورد ۱۶ از Bloch's *Effective Java* "ترجیح واسط ها به کلاس‌های Abstract" ترجیح داشته باشد. تغییر کلاس اصلی به واسط ، منجر به پیاده سازی واسط Vector فرضی توسط کلاس MyVector می‌شود. با این روش، کد سمت کلاینت که از MyVector استفاده می‌کند با کدی که از پیاده سازی اصلی Vector استفاده می‌کند سازگار خواهد ماند.

ارزیابی خطر

استفاده مجدد از شناسه‌های عمومی خوانایی و نگهداری کد را کاهش می‌دهد.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
DCL01-J	Low	Unlikely	Medium	**P2**	**L3**

تشخیص خودکار

یک ابزار خودکار می‌تواند استفاده مجدد از مجموعه نام‌هایی را که بیانگر کلاس‌های عمومی یا واسط هایی از کتابخانه استاندارد جاوا است به راحتی تشخیص دهد.