FB.SKIPPED\_CLASS\_TOO\_BIG

Этот класс больше, чем может быть эффективно обработан, и не был полностью проанализирован на наличие ошибок.

Причины такой ошибки могут варьироваться в зависимости от контекста и инструмента анализа, который используется. В некоторых случаях это может быть вызвано неэффективным использованием ресурсов, а в других случаях просто ограничением на размер объектов, которые могут быть обработаны.

Таблица №1 – Основные характеристики детектора FB.SKIPPED\_CLASS\_TOO\_BIG

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Excessive Class Size |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Снижение читаемости и поддерживаемости кода, высокое зацепление, трудности с тестированием, нарушение принципа единой ответственности (SRP) |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| // Проблемный класс (вложен в основной для соответствия шаблону)  class MonsterClass {  private String userData;  private String orderData;  public void processUser() { System.out.println("Processing user"); }  public void validateOrder() { System.out.println("Validating order"); }  // ... 100+ других методов  }  public static void incorrectTest() {  // Пример использования слишком большого класса  MonsterClass monster = new MonsterClass();  monster.processUser();  monster.validateOrder();  } | // Исправленные классы (вложены)  class UserManager {  public void registerUser() { System.out.println("User registered"); }  }  class OrderProcessor {  public void processOrder() { System.out.println("Order processed"); }  class MonsterClass {  private String userData;  private String orderData;  public void processUser() { System.out.println("Processing user"); }  public void validateOrder() { System.out.println("Validating order"); }  // ... 100+ других методов  } |