FB.USM\_USELESS\_SUBCLASS\_METHOD

Этот производный метод просто вызывает тот же метод суперкласса, передавая точные полученные параметры. Этот метод можно удалить, поскольку он не несет никакой дополнительной ценности.

Таблица №1 – Основные характеристики детектора FB.USM\_USELESS\_SUBCLASS\_METHOD

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Design Flaw |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Нарушение принципа подстановки Лисков (LSP) * Пустые или тривиальные переопределения методов * Усложнение поддержки кода |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class BaseOperation {  public void execute() {  System.out.println("Base operation executed");  }  }    // Некорректно: бесполезное переопределение  class SpecificOperation extends BaseOperation {  @Override  public void execute() {  // Пустая реализация (нарушает LSP)  }  }    new SpecificOperation().execute(); // Ничего не делает  } | public static void correctTest() {  abstract class Operation {  public abstract void execute();  }    // Корректно: осмысленное переопределение  class ConcreteOperation extends Operation {  @Override  public void execute() {  System.out.println("Concrete operation executed");  }  }    // Альтернатива: композиция вместо наследования  class OperationDelegate {  private final Runnable operation;    public OperationDelegate(Runnable operation) {  this.operation = operation;  }    public void execute() {  operation.run();  }  }    new ConcreteOperation().execute();  new OperationDelegate(() -> System.out.println("Delegated operation")).execute();  } |