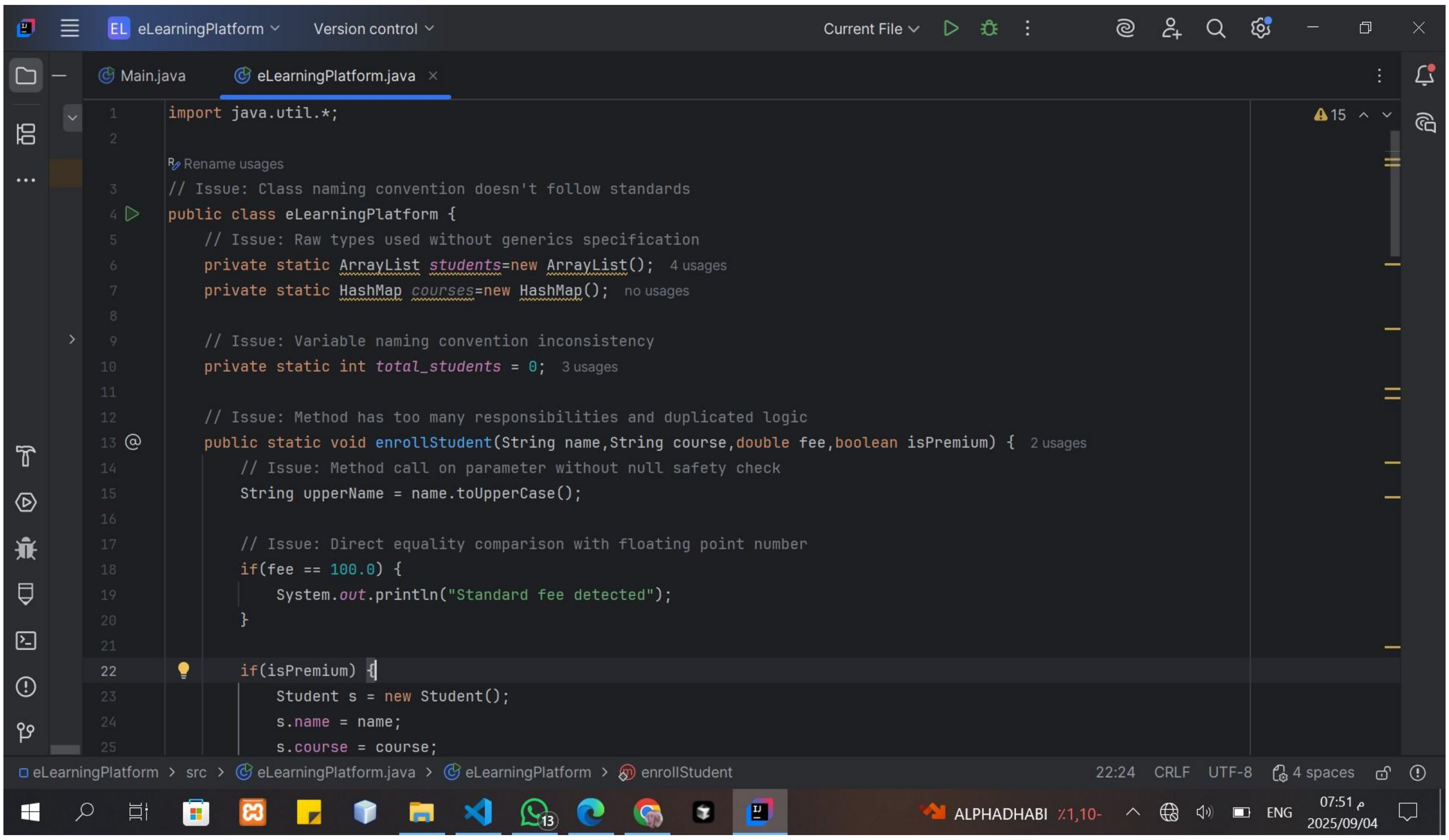
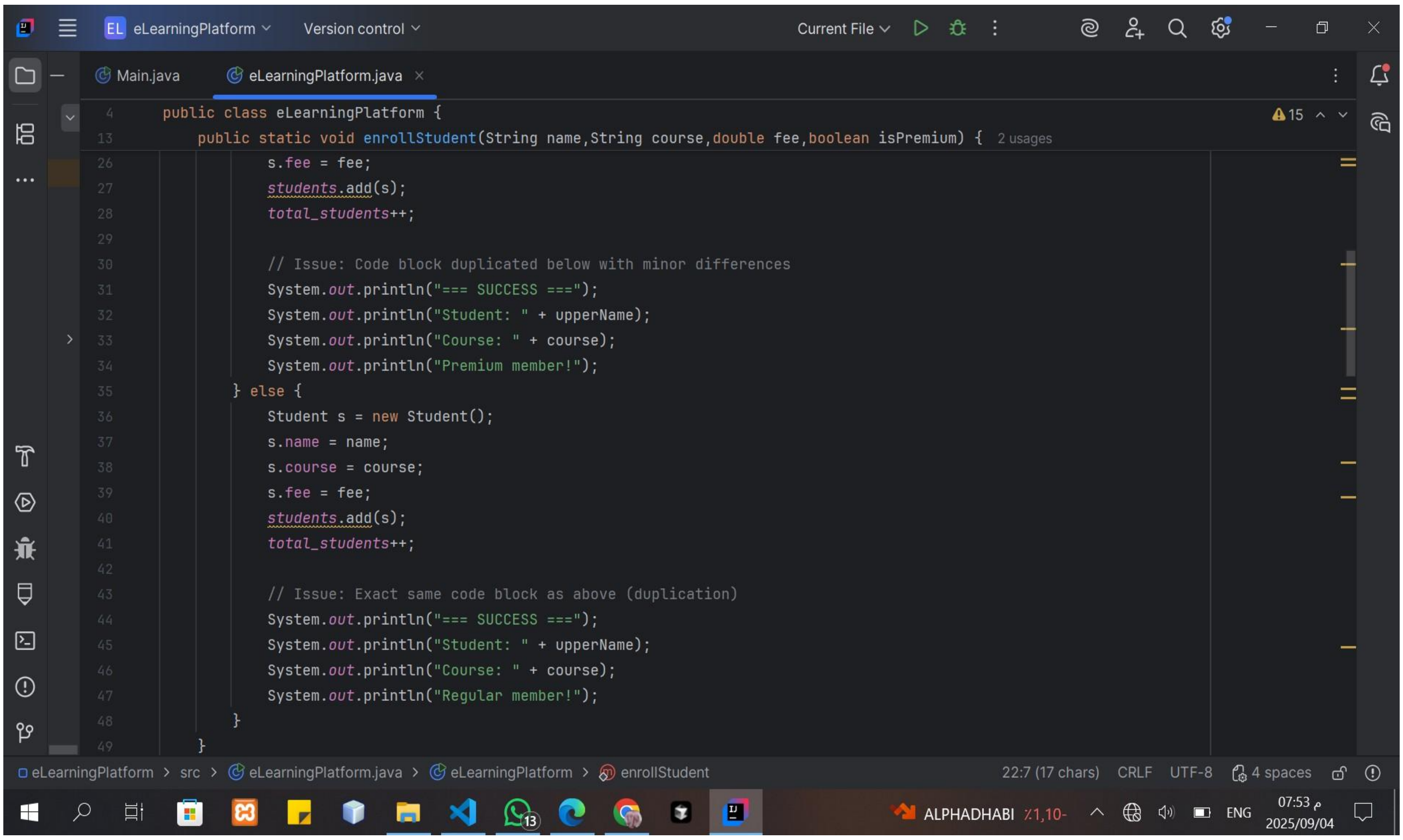
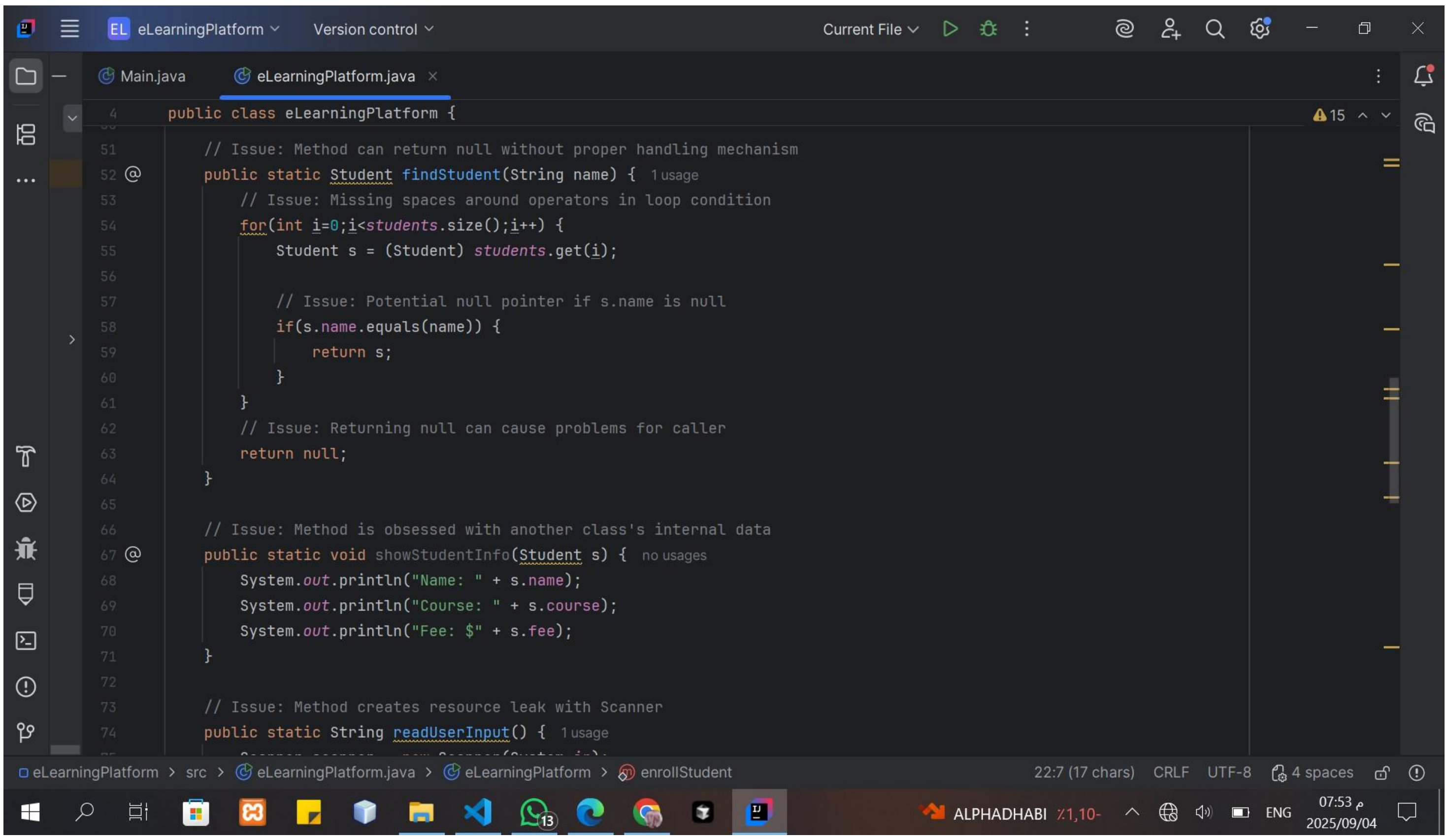
Code :



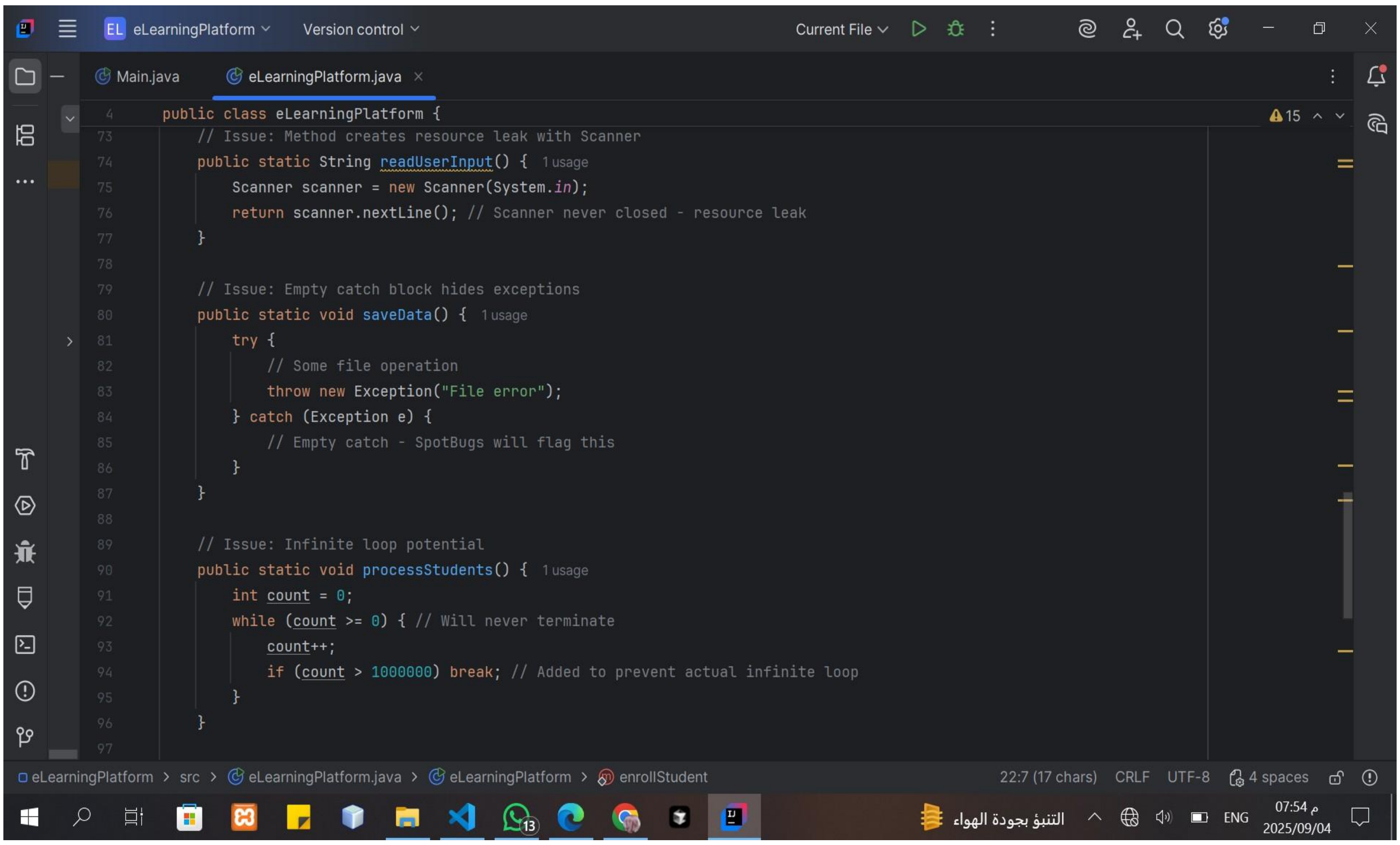
```java
import java.util.*;

// Issue: Class naming convention doesn't follow standards
public class eLearningPlatform {
    // Issue: Raw types used without generics specification
    private static ArrayList students=new ArrayList();
    private static HashMap courses=new HashMap();

    // Issue: Variable naming convention inconsistency
    private static int total_students = 0;

    // Issue: Method has too many responsibilities and duplicated logic
    public static void enrollStudent(String name,String course,double fee,boolean isPremium) {
        // Issue: Method call on parameter without null safety check
        String upperName = name.toUpperCase();

        // Issue: Direct equality comparison with floating point number
        if(fee == 100.0) {
            System.out.println("Standard fee detected");
        }

        if(isPremium) {
            Student s = new Student();
            s.name = name;
            s.course = course;
```



```java
public class eLearningPlatform {
    public static void enrollStudent(String name,String course,double fee,boolean isPremium) {
            s.fee = fee;
            students.add(s);
            total_students++;

            // Issue: Code block duplicated below with minor differences
            System.out.println("=== SUCCESS ===");
            System.out.println("Student: " + upperName);
            System.out.println("Course: " + course);
            System.out.println("Premium member!");
        } else {
            Student s = new Student();
            s.name = name;
            s.course = course;
            s.fee = fee;
            students.add(s);
            total_students++;

            // Issue: Exact same code block as above (duplication)
            System.out.println("=== SUCCESS ===");
            System.out.println("Student: " + upperName);
            System.out.println("Course: " + course);
            System.out.println("Regular member!");
        }
    }
```

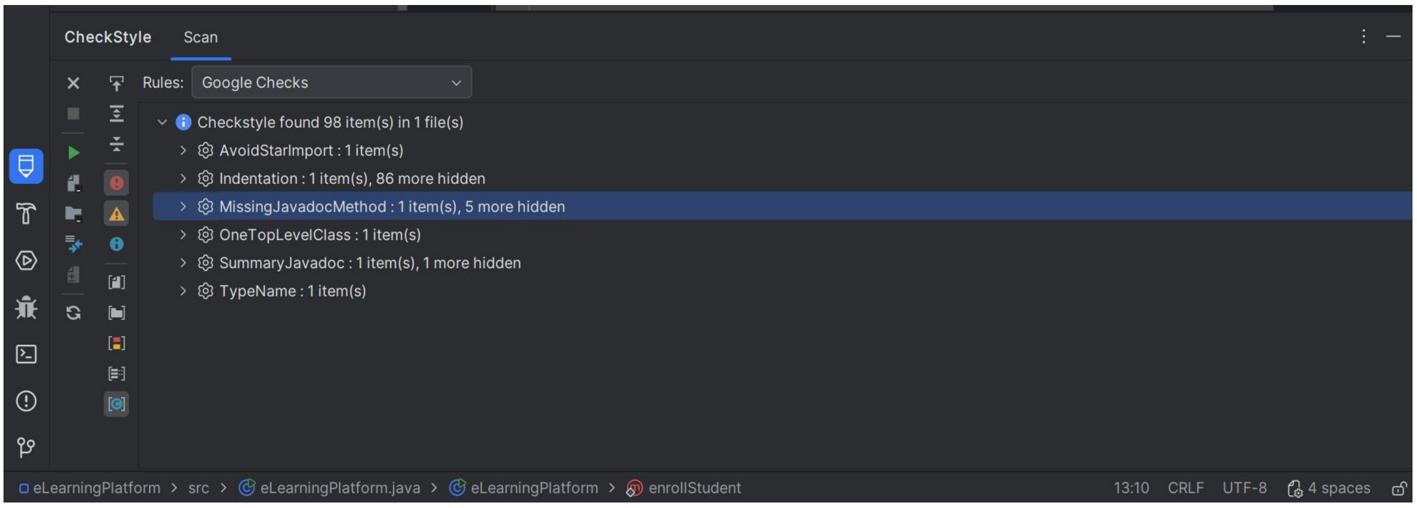


```java
public class eLearningPlatform {
    // Issue: Method can return null without proper handling mechanism
    public static Student findStudent(String name) {
        // Issue: Missing spaces around operators in loop condition
        for(int i=0;i<students.size();i++) {
            Student s = (Student) students.get(i);

            // Issue: Potential null pointer if s.name is null
            if(s.name.equals(name)) {
                return s;
            }
        }
        // Issue: Returning null can cause problems for caller
        return null;
    }

    // Issue: Method is obsessed with another class's internal data
    public static void showStudentInfo(Student s) {
        System.out.println("Name: " + s.name);
        System.out.println("Course: " + s.course);
        System.out.println("Fee: $" + s.fee);
    }

    // Issue: Method creates resource leak with Scanner
    public static String readUserInput() {
```



```java
public class eLearningPlatform {
    // Issue: Method creates resource leak with Scanner
    public static String readUserInput() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextLine(); // Scanner never closed - resource leak
    }

    // Issue: Empty catch block hides exceptions
    public static void saveData() {
        try {
            // Some file operation
            throw new Exception("File error");
        } catch (Exception e) {
            // Empty catch - SpotBugs will flag this
        }
    }

    // Issue: Infinite loop potential
    public static void processStudents() {
        int count = 0;
        while (count >= 0) { // Will never terminate
            count++;
            if (count > 1000000) break; // Added to prevent actual infinite loop
        }
    }

```

Apply checkstyle to detect style problems and correct them

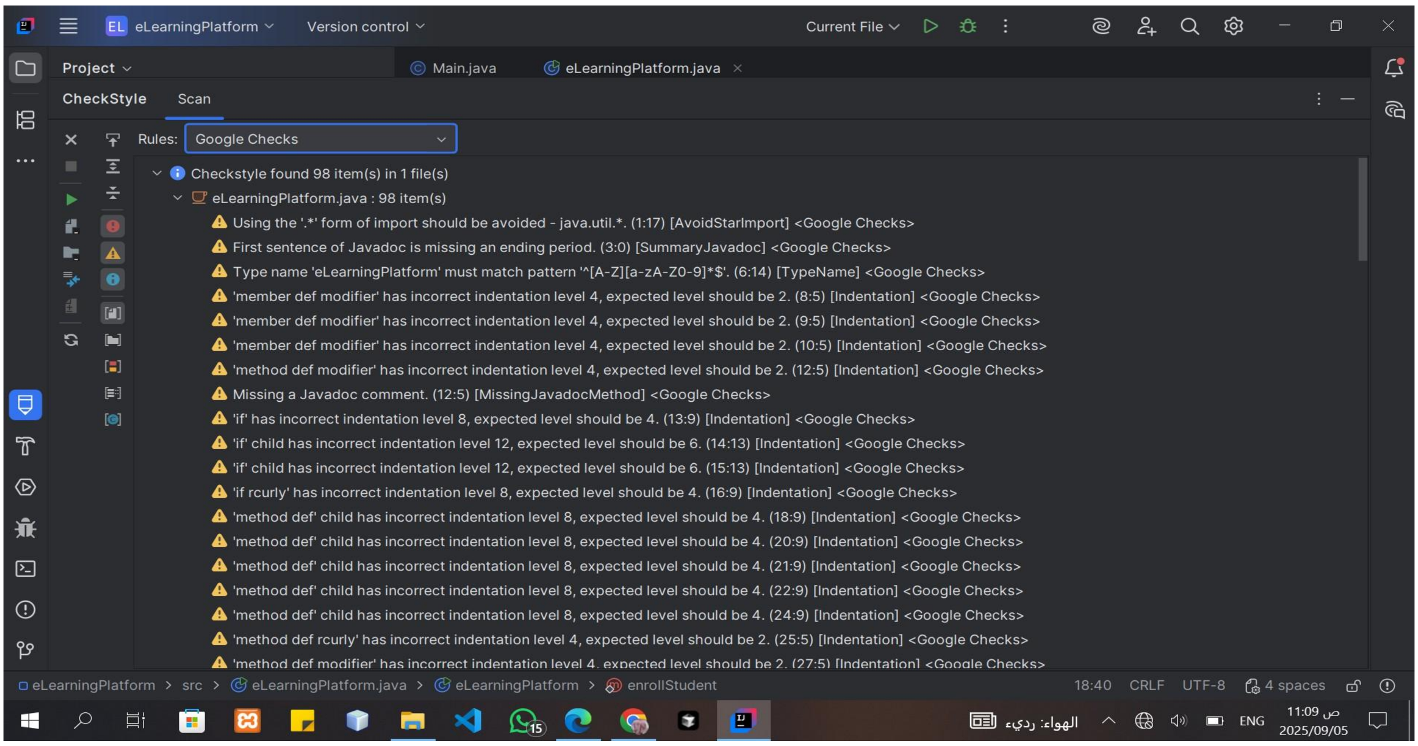


CheckStyle Scan

Rules: Google Checks

```
Checkstyle found 98 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  Indentation : 1 item(s), 86 more hidden
  MissingJavadocMethod : 1 item(s), 5 more hidden
  OneTopLevelClass : 1 item(s)
  SummaryJavadoc : 1 item(s), 1 more hidden
  TypeName : 1 item(s)
```

eLearningPlatform > src > eLearningPlatform.java > eLearningPlatform > enrollStudent   13:10 CRLF UTF-8 4 spaces



eLearningPlatform Version control Current File

Project Main.java eLearningPlatform.java

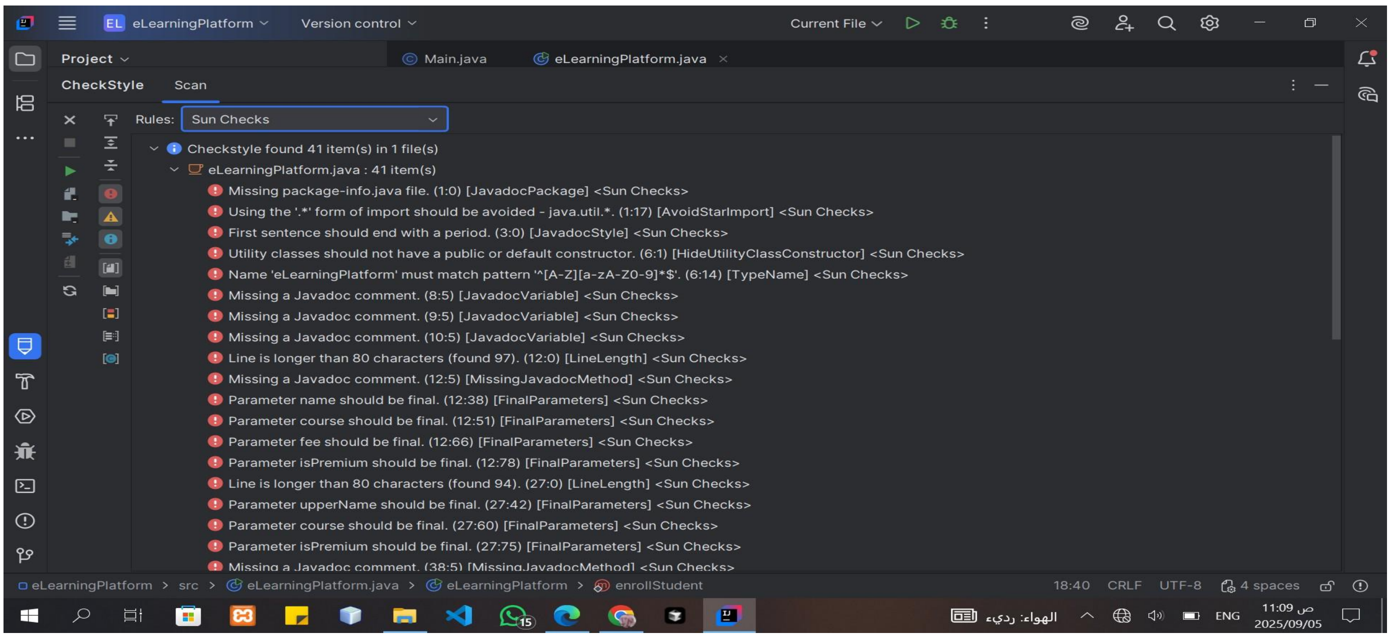
CheckStyle Scan

Rules: Google Checks

```
Checkstyle found 98 item(s) in 1 file(s)
  eLearningPlatform.java : 98 item(s)
    Using the '.*' form of import should be avoided - java.util.*. (1:17) [AvoidStarImport] <Google Checks>
    First sentence of Javadoc is missing an ending period. (3:0) [SummaryJavadoc] <Google Checks>
    Type name 'eLearningPlatform' must match pattern '^[A-Z][a-zA-Z0-9]*$'. (6:14) [TypeName] <Google Checks>
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (8:5) [Indentation] <Google Checks>
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (9:5) [Indentation] <Google Checks>
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (10:5) [Indentation] <Google Checks>
    'method def modifier' has incorrect indentation level 4, expected level should be 2. (12:5) [Indentation] <Google Checks>
    Missing a Javadoc comment. (12:5) [MissingJavadocMethod] <Google Checks>
    'if' has incorrect indentation level 8, expected level should be 4. (13:9) [Indentation] <Google Checks>
    'if' child has incorrect indentation level 12, expected level should be 6. (14:13) [Indentation] <Google Checks>
    'if' child has incorrect indentation level 12, expected level should be 6. (15:13) [Indentation] <Google Checks>
    'if rcurly' has incorrect indentation level 8, expected level should be 4. (16:9) [Indentation] <Google Checks>
    'method def' child has incorrect indentation level 8, expected level should be 4. (18:9) [Indentation] <Google Checks>
    'method def' child has incorrect indentation level 8, expected level should be 4. (20:9) [Indentation] <Google Checks>
    'method def' child has incorrect indentation level 8, expected level should be 4. (21:9) [Indentation] <Google Checks>
    'method def' child has incorrect indentation level 8, expected level should be 4. (22:9) [Indentation] <Google Checks>
    'method def' child has incorrect indentation level 8, expected level should be 4. (24:9) [Indentation] <Google Checks>
    'method def rcurly' has incorrect indentation level 4, expected level should be 2. (25:5) [Indentation] <Google Checks>
    'method def modifier' has incorrect indentation level 4, expected level should be 2. (27:5) [Indentation] <Google Checks>
```

eLearningPlatform > src > eLearningPlatform.java > eLearningPlatform > enrollStudent   18:40 CRLF UTF-8 4 spaces

الهواء: ردي ENG 11:09 ص 2025/09/05



eLearningPlatform Version control Current File

Project Main.java eLearningPlatform.java

CheckStyle Scan

Rules: Sun Checks

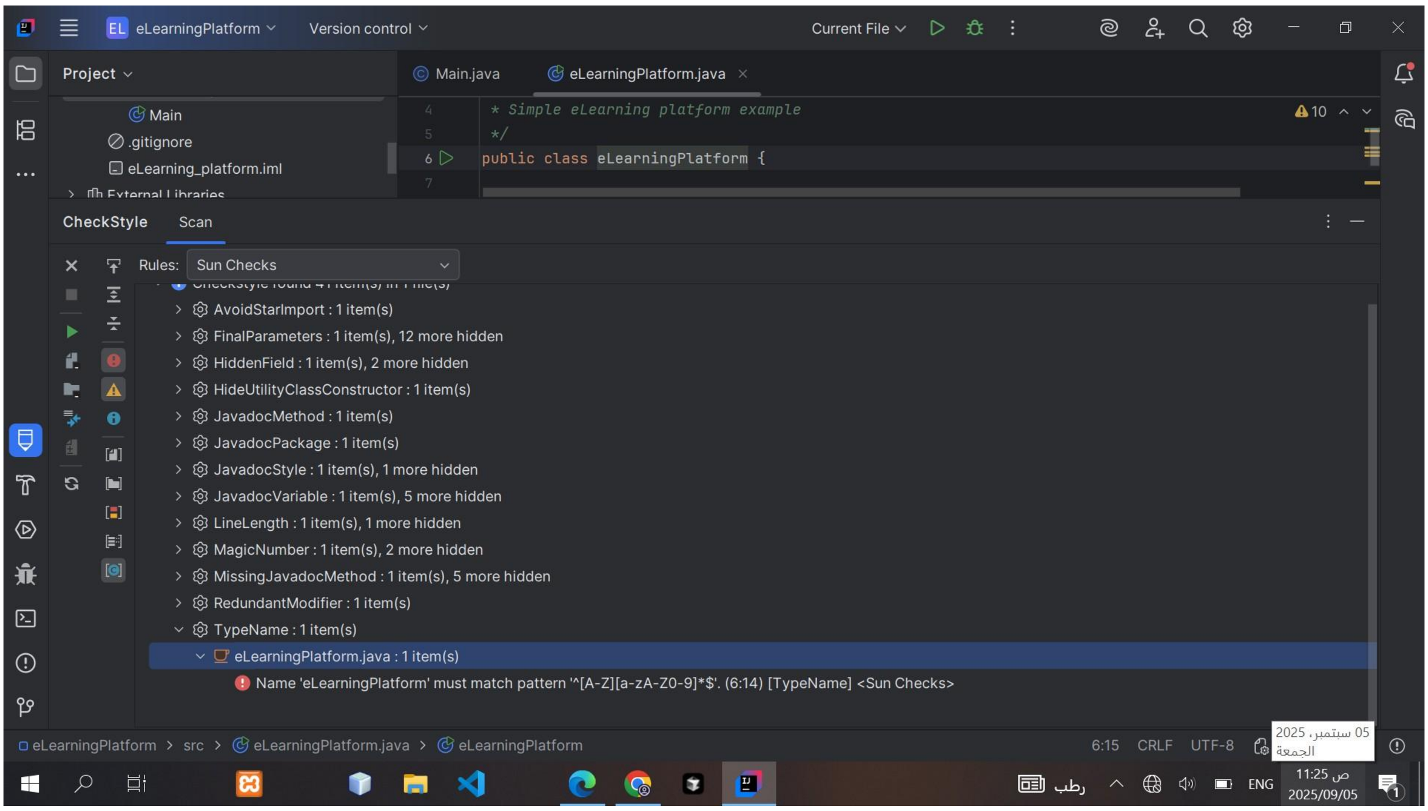
```
Checkstyle found 41 item(s) in 1 file(s)
  eLearningPlatform.java : 41 item(s)
    Missing package-info.java file. (1:0) [JavadocPackage] <Sun Checks>
    Using the '.*' form of import should be avoided - java.util.*. (1:17) [AvoidStarImport] <Sun Checks>
    First sentence should end with a period. (3:0) [JavadocStyle] <Sun Checks>
    Utility classes should not have a public or default constructor. (6:1) [HideUtilityClassConstructor] <Sun Checks>
    Name 'eLearningPlatform' must match pattern '^[A-Z][a-zA-Z0-9]*$'. (6:14) [TypeName] <Sun Checks>
    Missing a Javadoc comment. (8:5) [JavadocVariable] <Sun Checks>
    Missing a Javadoc comment. (9:5) [JavadocVariable] <Sun Checks>
    Missing a Javadoc comment. (10:5) [JavadocVariable] <Sun Checks>
    Line is longer than 80 characters (found 97). (12:0) [LineLength] <Sun Checks>
    Missing a Javadoc comment. (12:5) [MissingJavadocMethod] <Sun Checks>
    Parameter name should be final. (12:38) [FinalParameters] <Sun Checks>
    Parameter course should be final. (12:51) [FinalParameters] <Sun Checks>
    Parameter fee should be final. (12:66) [FinalParameters] <Sun Checks>
    Parameter isPremium should be final. (12:78) [FinalParameters] <Sun Checks>
    Line is longer than 80 characters (found 94). (27:0) [LineLength] <Sun Checks>
    Parameter upperName should be final. (27:42) [FinalParameters] <Sun Checks>
    Parameter course should be final. (27:60) [FinalParameters] <Sun Checks>
    Parameter isPremium should be final. (27:75) [FinalParameters] <Sun Checks>
    Missing a Javadoc comment. (38:5) [MissingJavadocMethod] <Sun Checks>
```

eLearningPlatform > src > eLearningPlatform.java > eLearningPlatform > enrollStudent   18:40 CRLF UTF-8 4 spaces

الهواء: ردي ENG 11:09 ص 2025/09/05

Problem1 :class naming

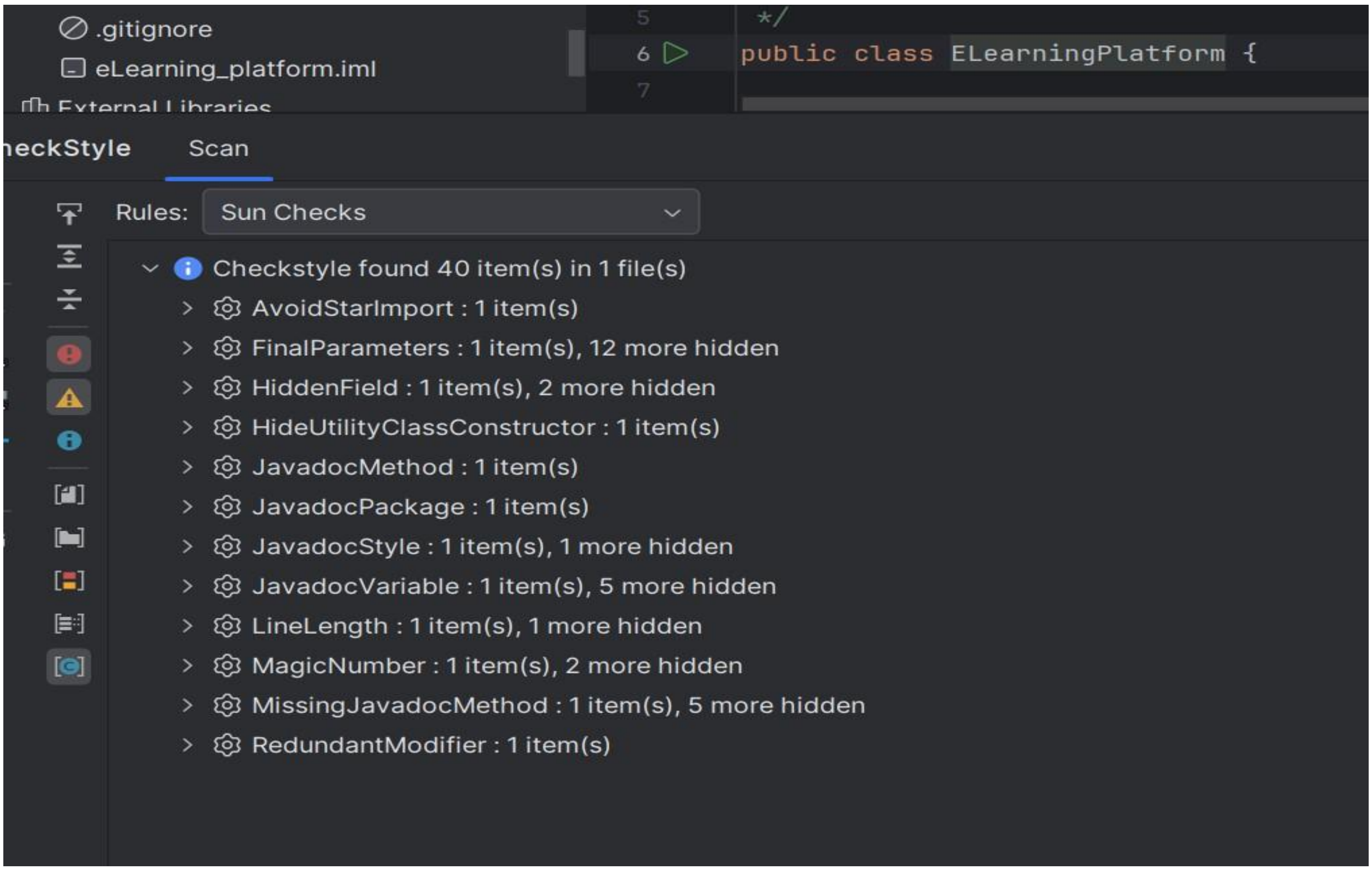
Before: Detected style problem using CheckStyle, class name `elearningPlatform` does not follow Java naming conventions.



```
 * Simple eLearning platform example
 */
public class eLearningPlatform {
```

```
Checkstyle found 41 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  FinalParameters : 1 item(s), 12 more hidden
  HiddenField : 1 item(s), 2 more hidden
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  LineLength : 1 item(s), 1 more hidden
  MagicNumber : 1 item(s), 2 more hidden
  MissingJavadocMethod : 1 item(s), 5 more hidden
  RedundantModifier : 1 item(s)
  TypeName : 1 item(s)
    eLearningPlatform.java : 1 item(s)
      Name 'eLearningPlatform' must match pattern '^[A-Z][a-zA-Z0-9]*$'. (6:14) [TypeName] <Sun Checks>
```

After correct : Changed class name from `elearningPlatform` to `ELearningPlatform`

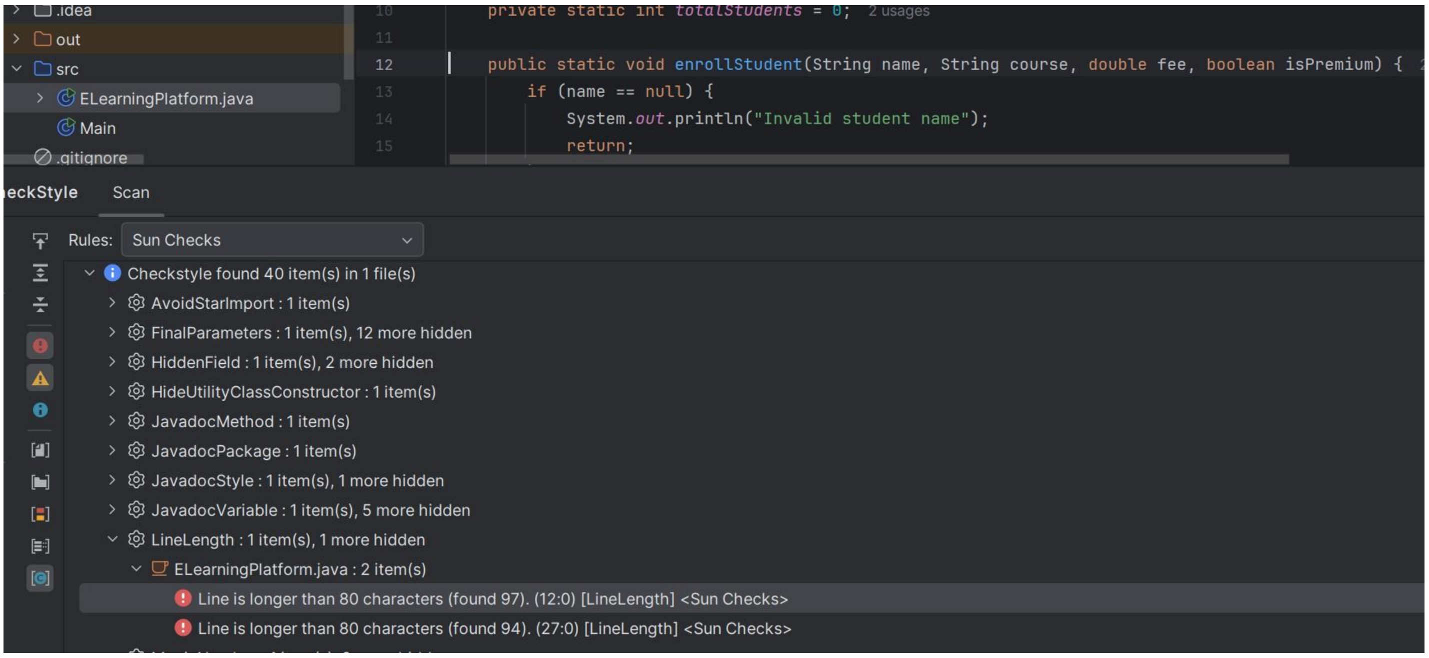


```
 */
public class ELearningPlatform {
```

```
Checkstyle found 40 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  FinalParameters : 1 item(s), 12 more hidden
  HiddenField : 1 item(s), 2 more hidden
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  LineLength : 1 item(s), 1 more hidden
  MagicNumber : 1 item(s), 2 more hidden
  MissingJavadocMethod : 1 item(s), 5 more hidden
  RedundantModifier : 1 item(s)
```

Problem2 : line length is very long

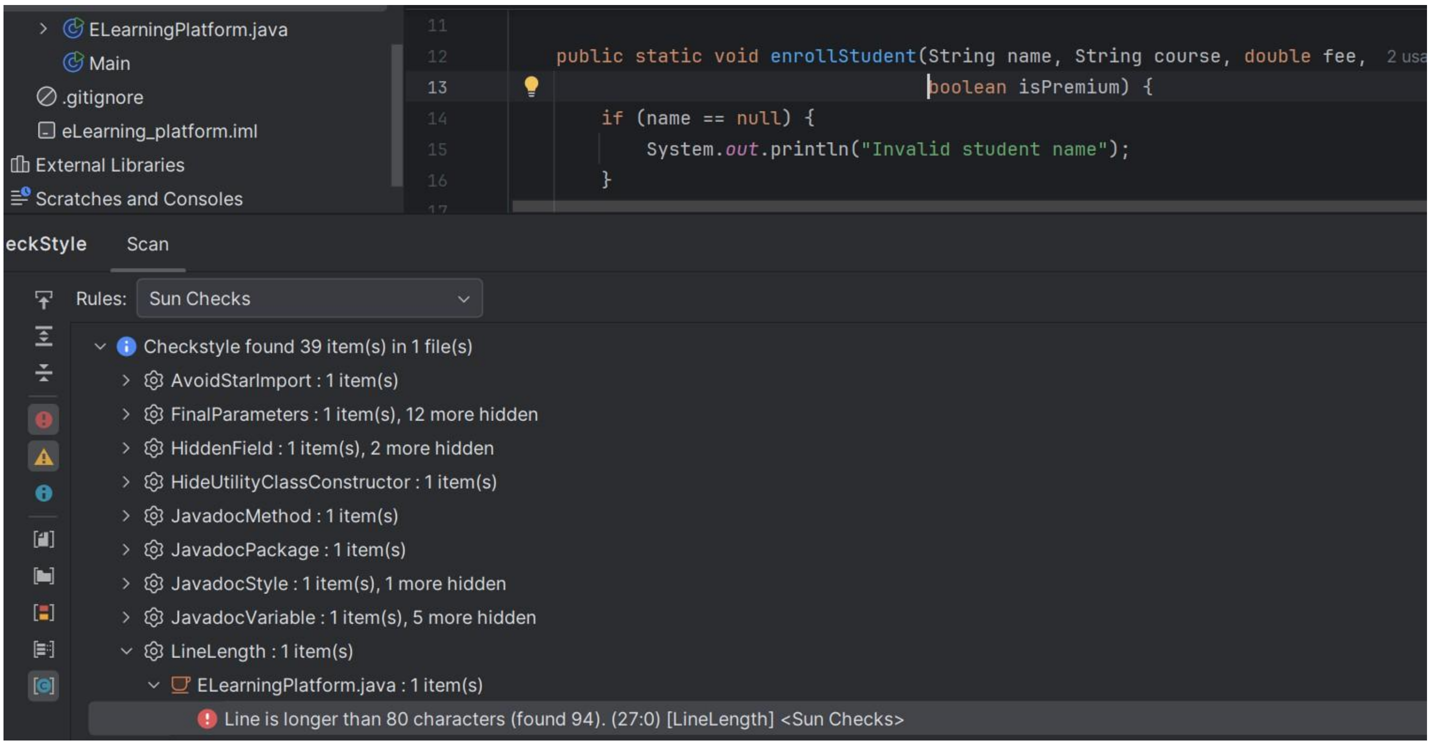
Before: Detected style problem using CheckStyle ,lines exceed the maximum allowed length



```
private static int totalStudents = 0;  2 usages

public static void enrollStudent(String name, String course, double fee, boolean isPremium) {
    if (name == null) {
        System.out.println("Invalid student name");
        return;
```

```
Rules: Sun Checks
Checkstyle found 40 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  FinalParameters : 1 item(s), 12 more hidden
  HiddenField : 1 item(s), 2 more hidden
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  LineLength : 1 item(s), 1 more hidden
    ELearningPlatform.java : 2 item(s)
      Line is longer than 80 characters (found 97). (12:0) [LineLength] <Sun Checks>
      Line is longer than 80 characters (found 94). (27:0) [LineLength] <Sun Checks>
```

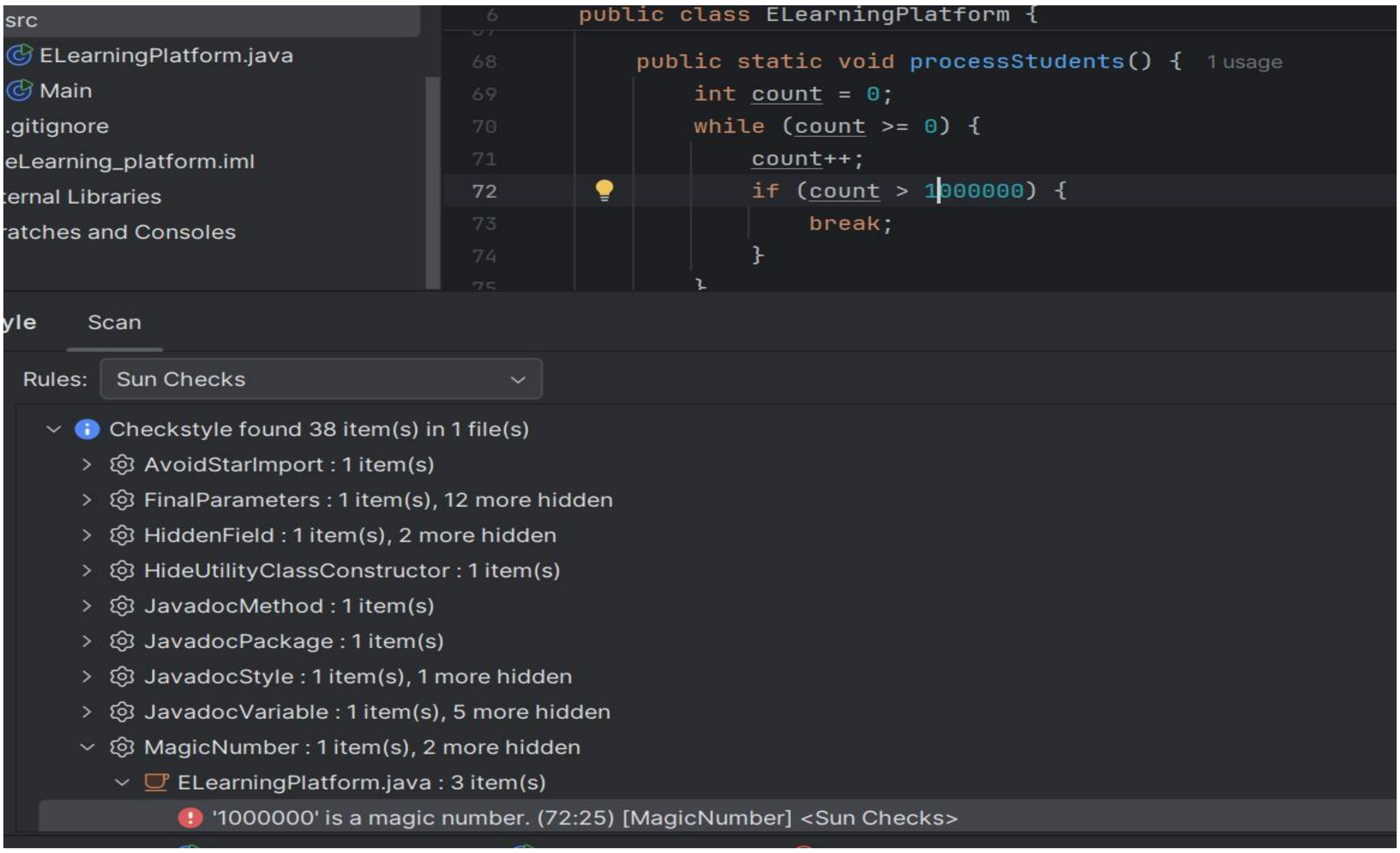
After Correct: Broke the long line into 2 lines to improve readability and comply with style rules.



```
public static void enrollStudent(String name, String course, double fee,
                                 boolean isPremium) {
    if (name == null) {
        System.out.println("Invalid student name");
    }
```

```
Rules: Sun Checks
Checkstyle found 39 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  FinalParameters : 1 item(s), 12 more hidden
  HiddenField : 1 item(s), 2 more hidden
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  LineLength : 1 item(s)
    ELearningPlatform.java : 1 item(s)
      Line is longer than 80 characters (found 94). (27:0) [LineLength] <Sun Checks>
```

Problem3 :magic number
Before : numeric 1000000 are used directly in the code, making the code less readable and harder to maintain



src
ELearningPlatform.java
Main
.gitignore
eLearning_platform.iml
ernal Libraries
atches and Consoles

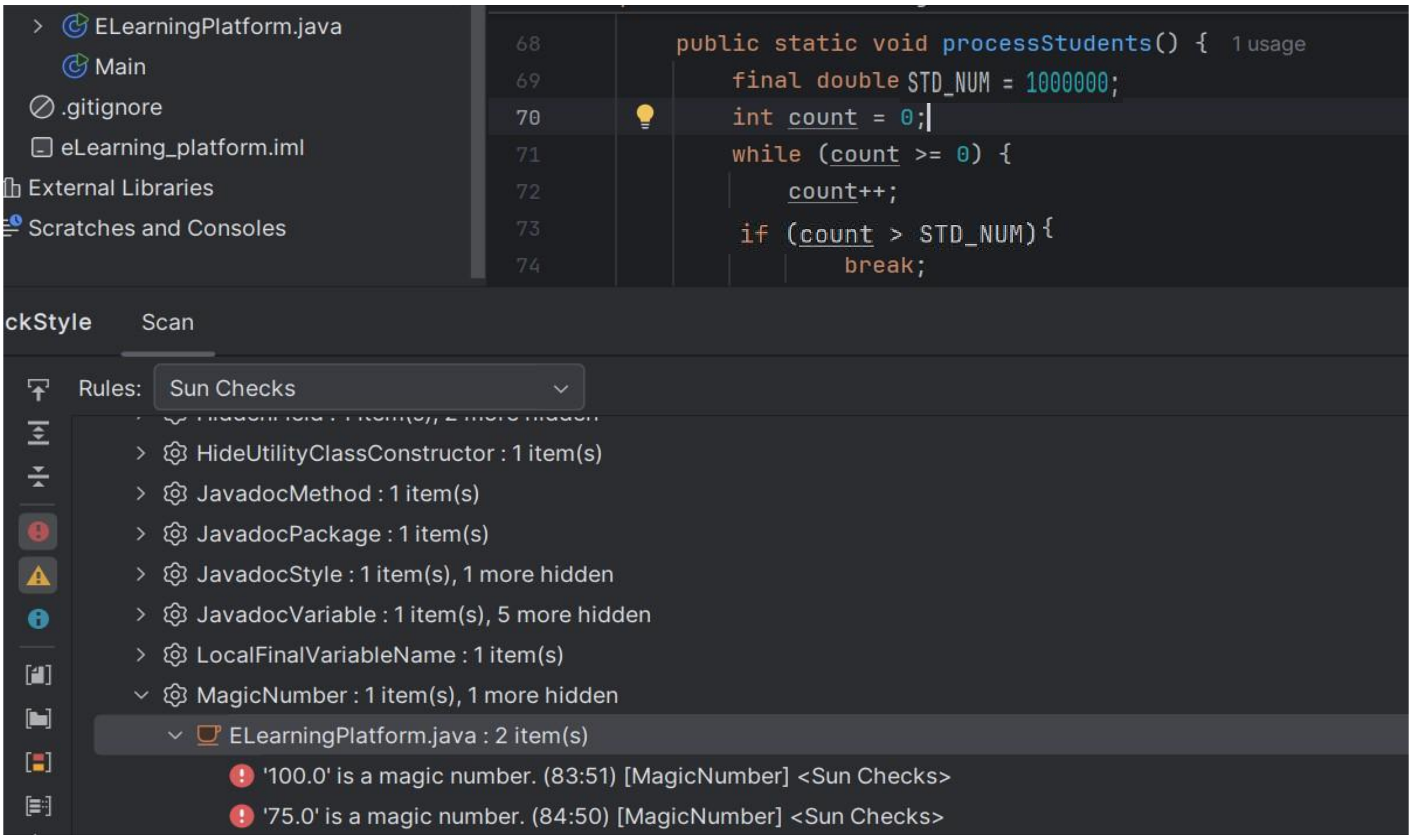
```
public class ELearningPlatform {

    public static void processStudents() {  1 usage
        int count = 0;
        while (count >= 0) {
            count++;
            if (count > 1000000) {
                break;
            }
```

yle   Scan

Rules: Sun Checks

```
Checkstyle found 38 item(s) in 1 file(s)
  AvoidStarImport : 1 item(s)
  FinalParameters : 1 item(s), 12 more hidden
  HiddenField : 1 item(s), 2 more hidden
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  MagicNumber : 1 item(s), 2 more hidden
    ELearningPlatform.java : 3 item(s)
      '1000000' is a magic number. (72:25) [MagicNumber] <Sun Checks>
```

After (Correct): Replaced the magic number with a named constant STD_NUM and reuse constant



ELearningPlatform.java
Main
.gitignore
eLearning_platform.iml
External Libraries
Scratches and Consoles

```
    public static void processStudents() {  1 usage
        final double STD_NUM = 1000000;
        int count = 0;
        while (count >= 0) {
            count++;
         if (count > STD_NUM){
                break;
```

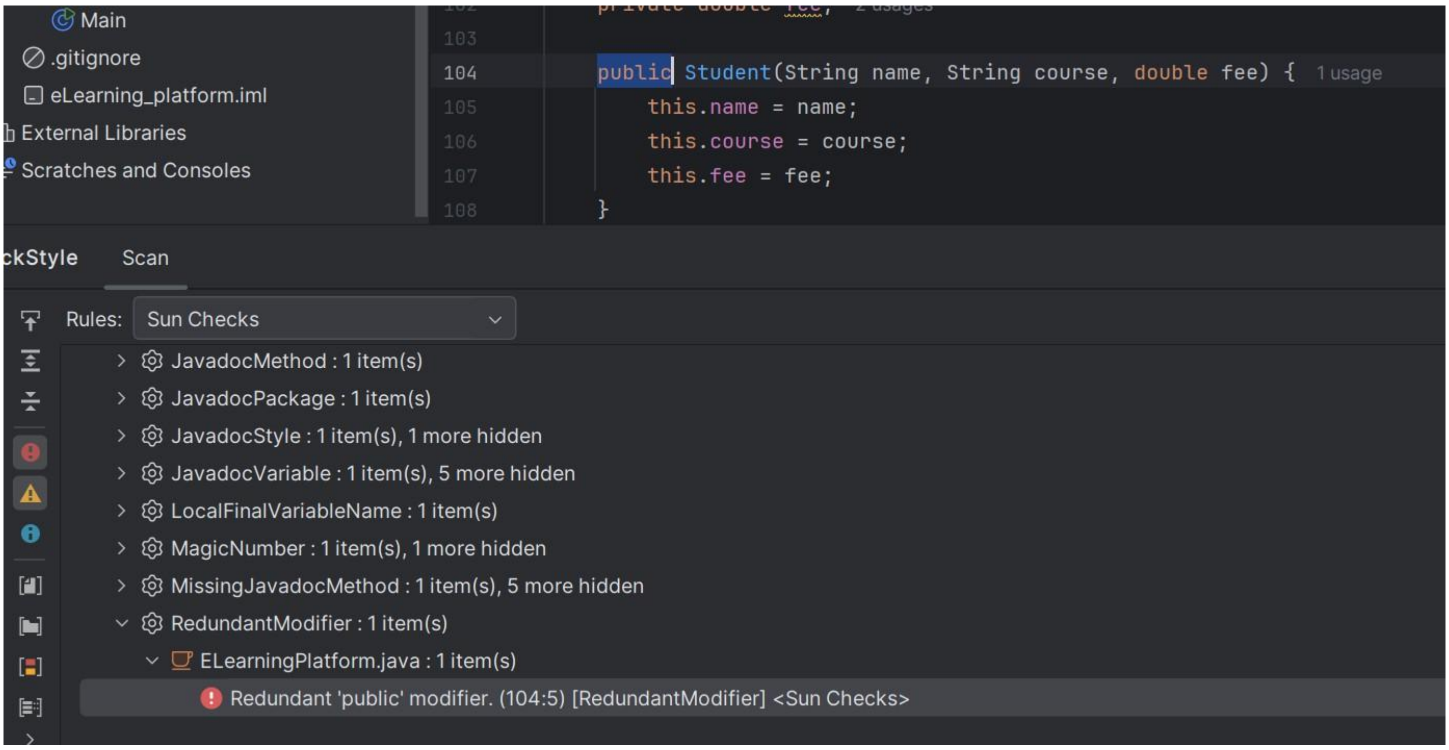
ckStyle   Scan

Rules: Sun Checks

```
  HideUtilityClassConstructor : 1 item(s)
  JavadocMethod : 1 item(s)
  JavadocPackage : 1 item(s)
  JavadocStyle : 1 item(s), 1 more hidden
  JavadocVariable : 1 item(s), 5 more hidden
  LocalFinalVariableName : 1 item(s)
  MagicNumber : 1 item(s), 1 more hidden
    ELearningPlatform.java : 2 item(s)
      '100.0' is a magic number. (83:51) [MagicNumber] <Sun Checks>
      '75.0' is a magic number. (84:50) [MagicNumber] <Sun Checks>
```

Problem 4: Redundant Modifier

Before: Constructor declared as `public`in the default package and constructors are accessible without `public`

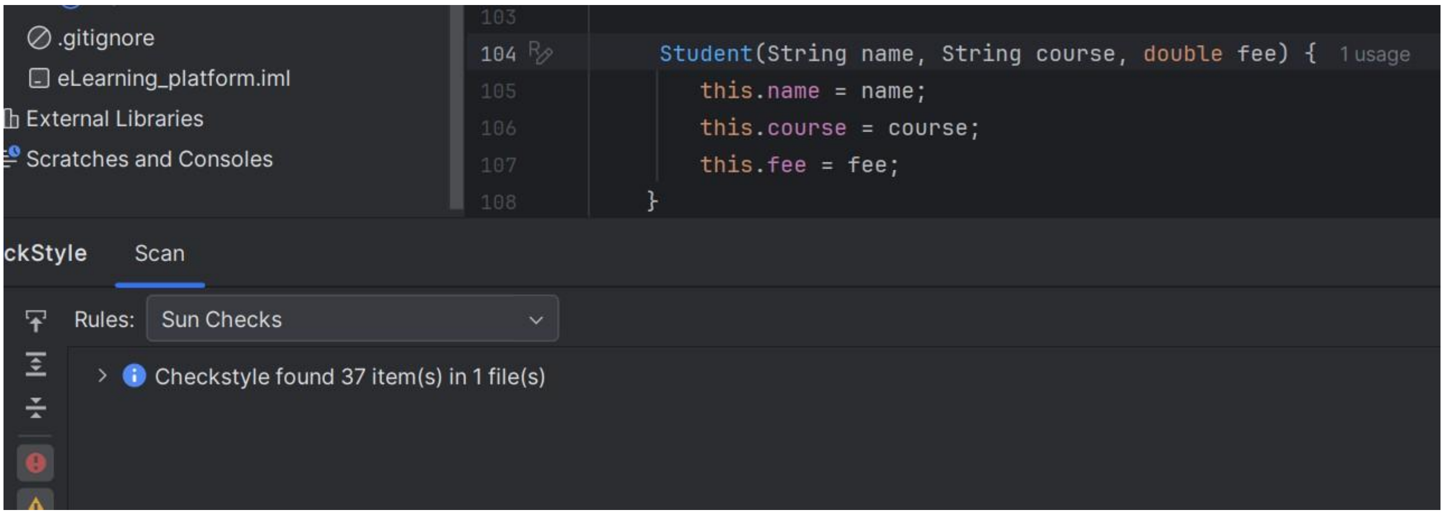


```
    public Student(String name, String course, double fee) {  1 usage
        this.name = name;
        this.course = course;
        this.fee = fee;
    }
```

Rules: Sun Checks

```
> JavadocMethod : 1 item(s)
> JavadocPackage : 1 item(s)
> JavadocStyle : 1 item(s), 1 more hidden
> JavadocVariable : 1 item(s), 5 more hidden
> LocalFinalVariableName : 1 item(s)
> MagicNumber : 1 item(s), 1 more hidden
> MissingJavadocMethod : 1 item(s), 5 more hidden
v RedundantModifier : 1 item(s)
  v ELearningPlatform.java : 1 item(s)
      Redundant 'public' modifier. (104:5) [RedundantModifier] <Sun Checks>
```

After (Correct): Removed the `public` modifier:



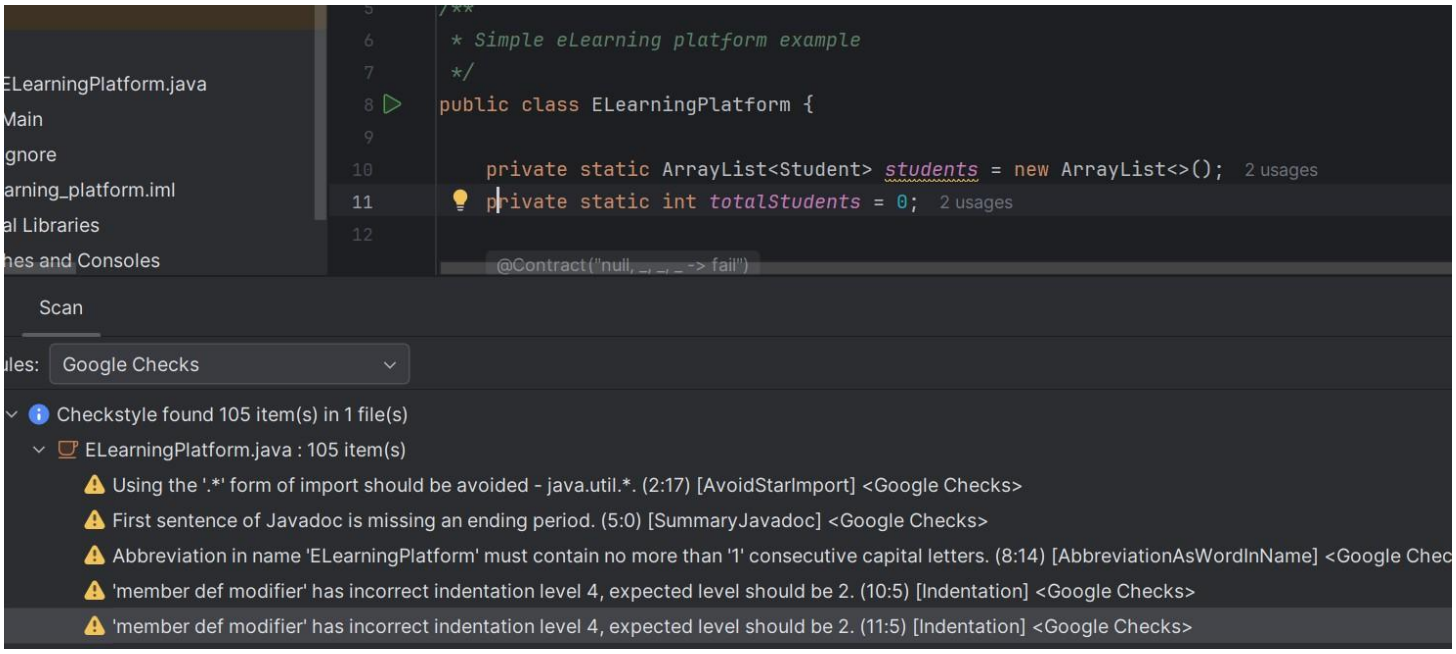
```
    Student(String name, String course, double fee) {  1 usage
        this.name = name;
        this.course = course;
        this.fee = fee;
    }
```

Rules: Sun Checks

```
> Checkstyle found 37 item(s) in 1 file(s)
```

Problem 4: Identation

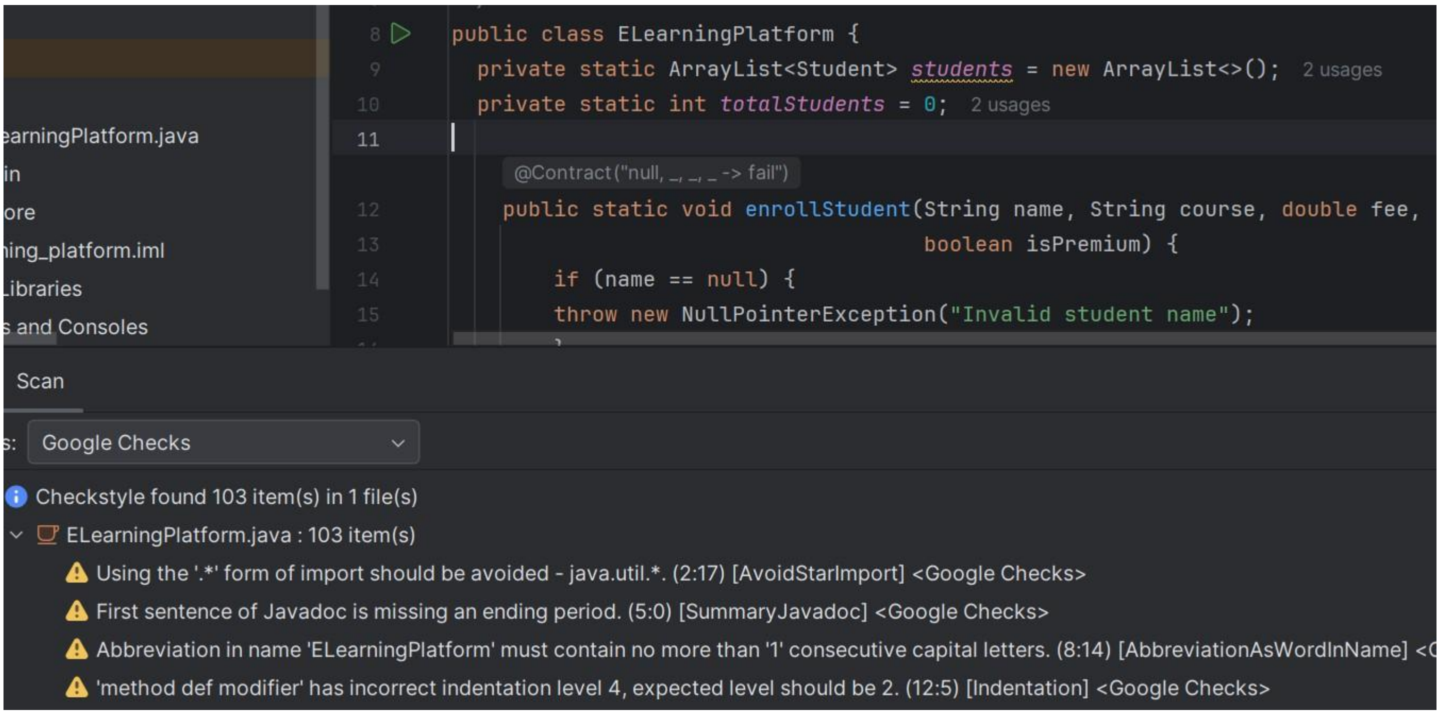
Before: Lines had 4 space indentation which violated Google Checkstyle rules (expected 2 spaces).



```
/**
 * Simple eLearning platform example
 */
public class ELearningPlatform {

    private static ArrayList<Student> students = new ArrayList<>();  2 usages
    private static int totalStudents = 0;  2 usages

  @Contract("null, _, _, _ -> fail")
```

```
Checkstyle found 105 item(s) in 1 file(s)
ELearningPlatform.java : 105 item(s)
Using the '.*' form of import should be avoided - java.util.*. (2:17) [AvoidStarImport] <Google Checks>
First sentence of Javadoc is missing an ending period. (5:0) [SummaryJavadoc] <Google Checks>
Abbreviation in name 'ELearningPlatform' must contain no more than '1' consecutive capital letters. (8:14) [AbbreviationAsWordInName] <Google Chec
'member def modifier' has incorrect indentation level 4, expected level should be 2. (10:5) [Indentation] <Google Checks>
'member def modifier' has incorrect indentation level 4, expected level should be 2. (11:5) [Indentation] <Google Checks>
```

After: Fixed indentation to 2 spaces follows proper Java formatting conventions.

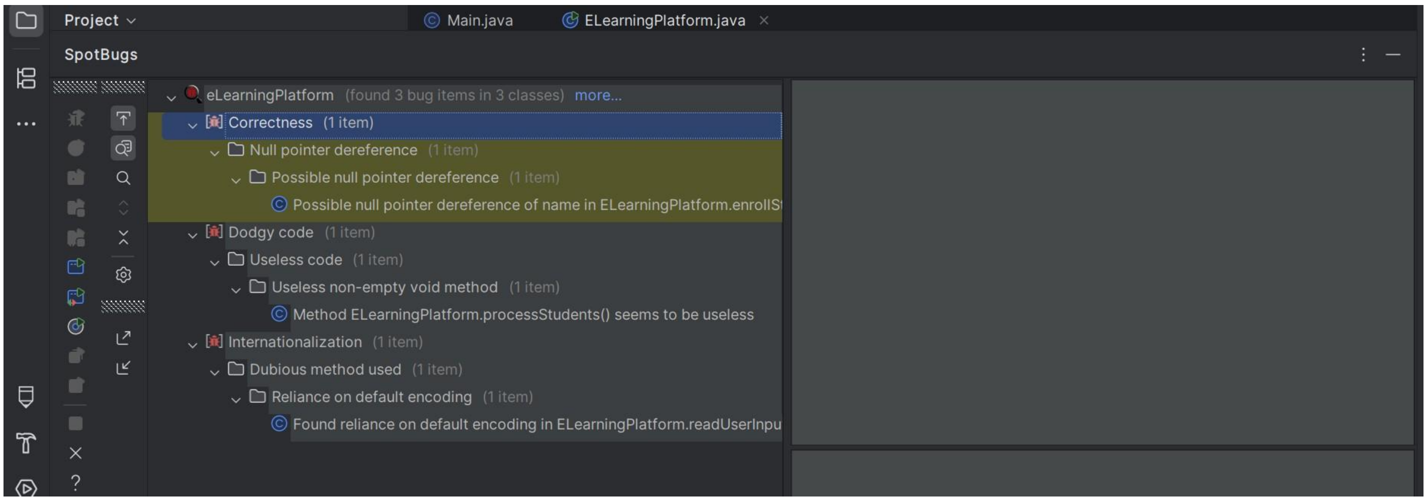


```
public class ELearningPlatform {
  private static ArrayList<Student> students = new ArrayList<>();  2 usages
  private static int totalStudents = 0;  2 usages

    @Contract("null, _, _, _ -> fail")
    public static void enrollStudent(String name, String course, double fee,
                                     boolean isPremium) {
        if (name == null) {
        throw new NullPointerException("Invalid student name");
```

```
Checkstyle found 103 item(s) in 1 file(s)
ELearningPlatform.java : 103 item(s)
Using the '.*' form of import should be avoided - java.util.*. (2:17) [AvoidStarImport] <Google Checks>
First sentence of Javadoc is missing an ending period. (5:0) [SummaryJavadoc] <Google Checks>
Abbreviation in name 'ELearningPlatform' must contain no more than '1' consecutive capital letters. (8:14) [AbbreviationAsWordInName] <(
'method def modifier' has incorrect indentation level 4, expected level should be 2. (12:5) [Indentation] <Google Checks>
```

Apply SpotBugs to detect at least 3 faults and correct them.

Before :applying spotBugs analysis and detect bugs



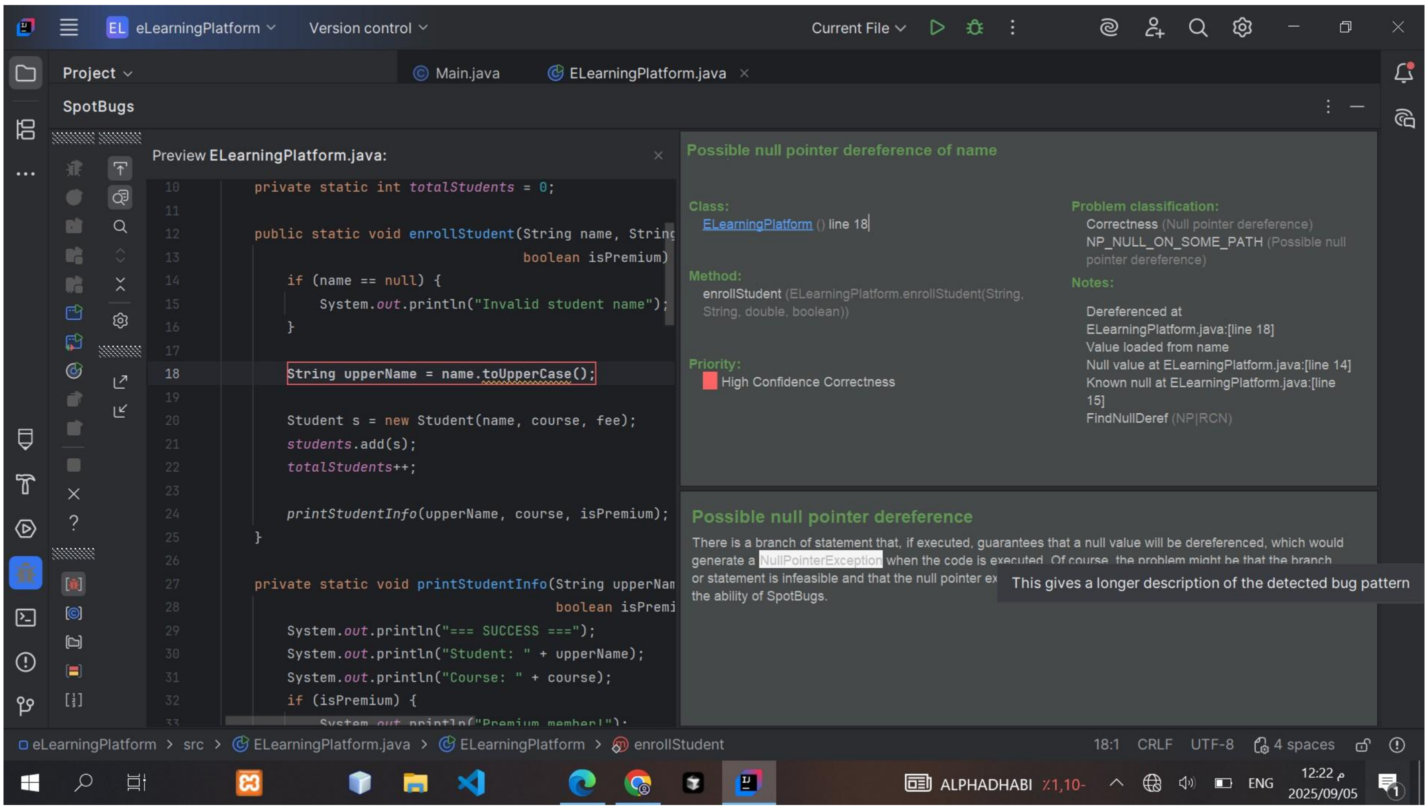
Project | Main.java | ELearningPlatform.java

SpotBugs

```
eLearningPlatform (found 3 bug items in 3 classes) more...
  Correctness (1 item)
    Null pointer dereference (1 item)
      Possible null pointer dereference (1 item)
        Possible null pointer dereference of name in ELearningPlatform.enrollSt
  Dodgy code (1 item)
    Useless code (1 item)
      Useless non-empty void method (1 item)
        Method ELearningPlatform.processStudents() seems to be useless
  Internationalization (1 item)
    Dubious method used (1 item)
      Reliance on default encoding (1 item)
        Found reliance on default encoding in ELearningPlatform.readUserInpu
```

Bug 1:Ineffective Exception Handling (Dead Code)

Before: In method `enrollStudent`, when `name` was `null`, the code only printed an error message ,which could lead to unexpected behavior later in the program.



Preview ELearningPlatform.java:

```
10      private static int totalStudents = 0;
11
12      public static void enrollStudent(String name, Strinc
13                                       boolean isPremium)
14          if (name == null) {
15              System.out.println("Invalid student name");
16          }
17
18          String upperName = name.toUpperCase();
19
20          Student s = new Student(name, course, fee);
21          students.add(s);
22          totalStudents++;
23
24          printStudentInfo(upperName, course, isPremium);
25      }
26
27      private static void printStudentInfo(String upperNam
28                                           boolean isPremi
29          System.out.println("=== SUCCESS ===");
30          System.out.println("Student: " + upperName);
31          System.out.println("Course: " + course);
32          if (isPremium) {
33              System.out.println("Premium member!");
```

Possible null pointer dereference of name

Class:
ELearningPlatform () line 18

Method:
enrollStudent (ELearningPlatform.enrollStudent(String, String, double, boolean))

Priority:
🟥 High Confidence Correctness

Problem classification:
Correctness (Null pointer dereference)
NP_NULL_ON_SOME_PATH (Possible null pointer dereference)

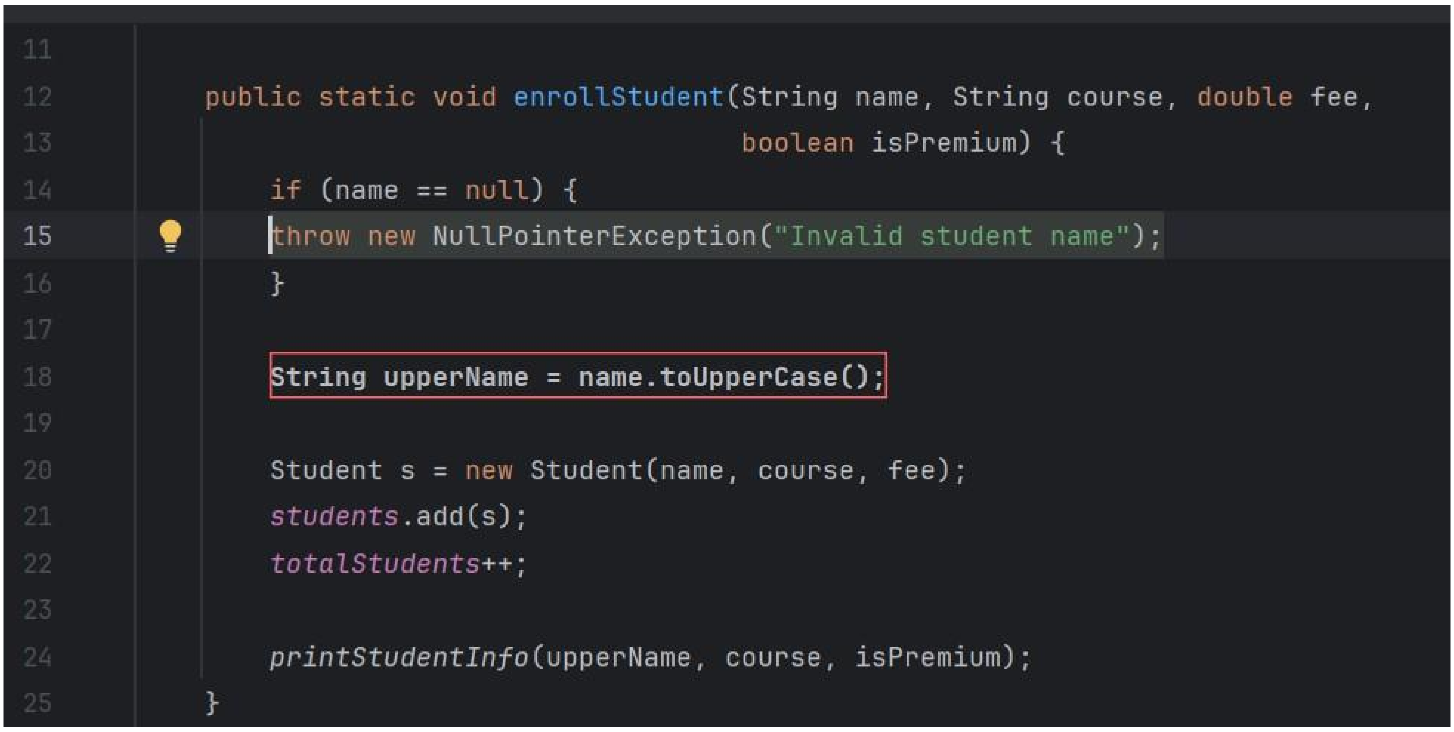
Notes:
Dereferenced at ELearningPlatform.java:[line 18]
Value loaded from name
Null value at ELearningPlatform.java:[line 14]
Known null at ELearningPlatform.java:[line 15]
FindNullDeref (NP|RCN)

Possible null pointer dereference

There is a branch of statement that, if executed, guarantees that a null value will be dereferenced, which would generate a NullPointerException when the code is executed. Of course, the problem might be that the branch or statement is infeasible and that the null pointer ex the ability of SpotBugs.

This gives a longer description of the detected bug pattern

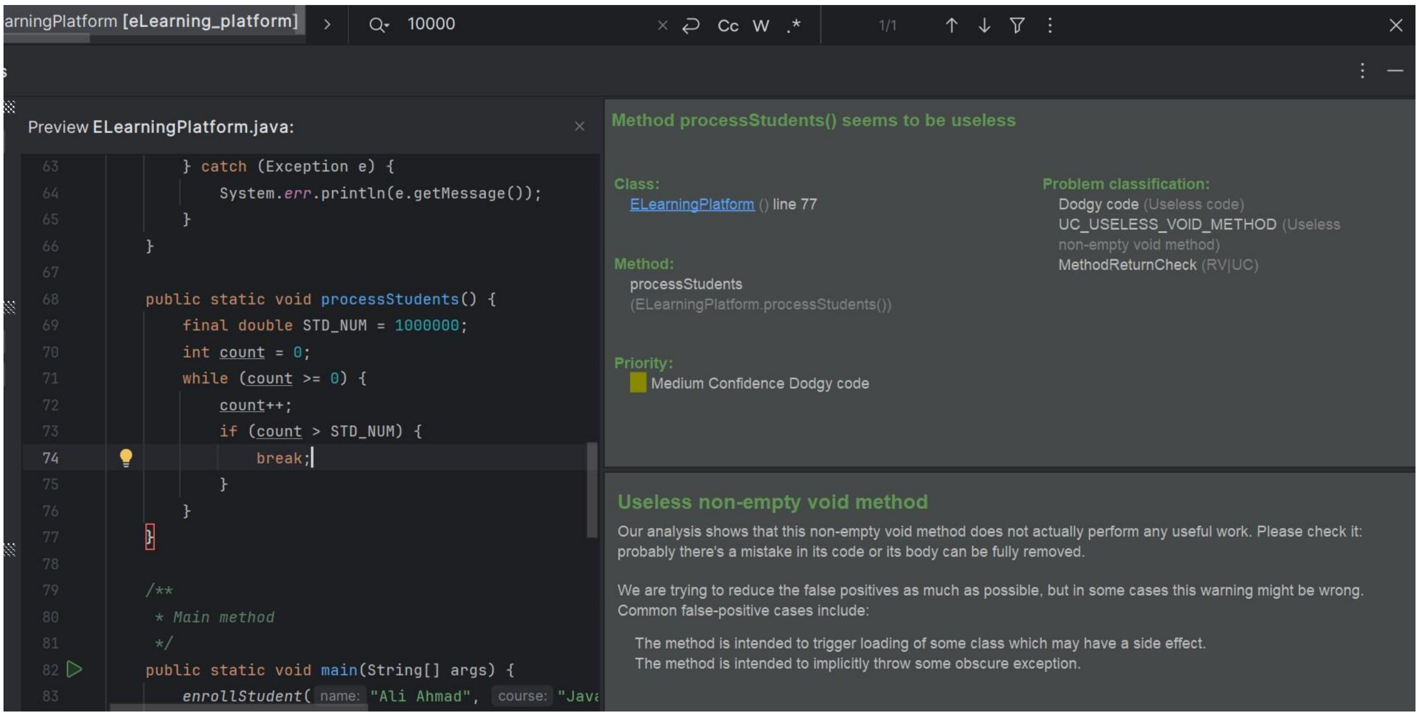
After (Correct): Replaced with throwing an exception to properly stop execution and signal an error.



```java

    public static void enrollStudent(String name, String course, double fee,
                                     boolean isPremium) {
        if (name == null) {
        throw new NullPointerException("Invalid student name");
        }

        String upperName = name.toUpperCase();

        Student s = new Student(name, course, fee);
        students.add(s);
        totalStudents++;

        printStudentInfo(upperName, course, isPremium);
    }
```

Bug 2: Useless Method

Before: Method had useless infinite loop that did nothing ,`while (count >= 0)` count keeps increasing forever, never stops



arningPlatform [eLearning_platform]   10000   1/1

Preview ELearningPlatform.java:

```java
63            } catch (Exception e) {
64                System.err.println(e.getMessage());
65            }
66        }
67
68        public static void processStudents() {
69            final double STD_NUM = 1000000;
70            int count = 0;
71            while (count >= 0) {
72                count++;
73                if (count > STD_NUM) {
74                    break;
75                }
76            }
77        }
78
79        /**
80         * Main method
81         */
82        public static void main(String[] args) {
83            enrollStudent( name: "Ali Ahmad",  course: "Jav
```

Method processStudents() seems to be useless

Class:
ELearningPlatform () line 77

Method:
processStudents
(ELearningPlatform.processStudents())

Priority:
Medium Confidence Dodgy code

Problem classification:
Dodgy code (Useless code)
UC_USELESS_VOID_METHOD (Useless non-empty void method)
MethodReturnCheck (RV|UC)

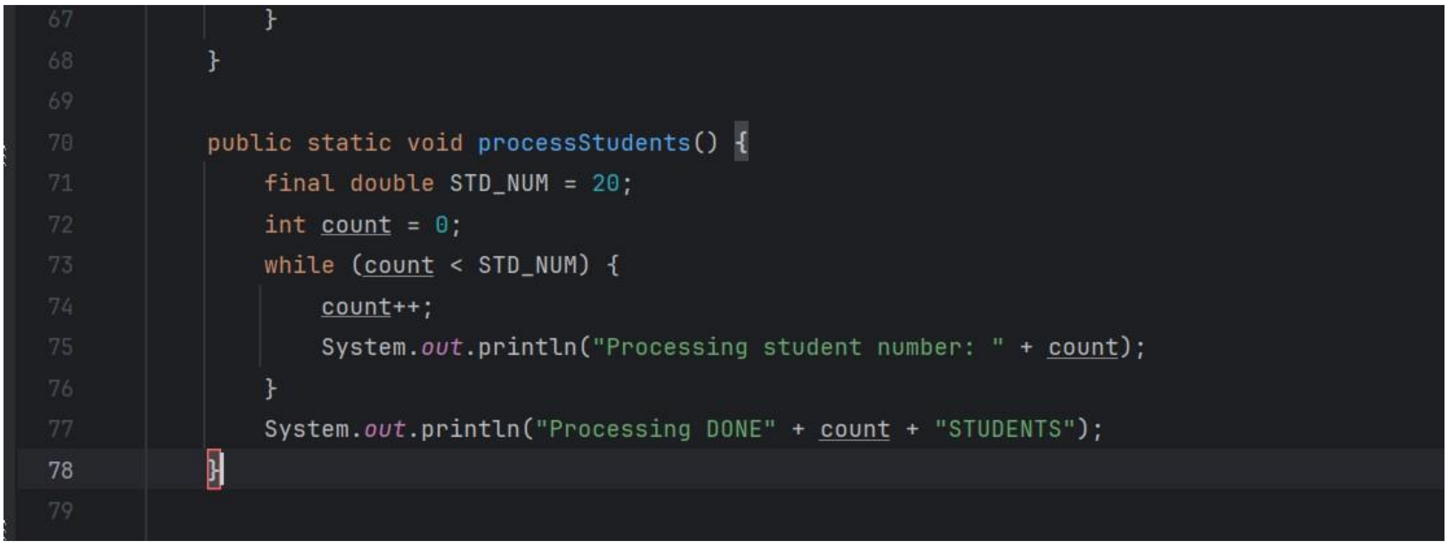
Useless non-empty void method

Our analysis shows that this non-empty void method does not actually perform any useful work. Please check it: probably there's a mistake in its code or its body can be fully removed.

We are trying to reduce the false positives as much as possible, but in some cases this warning might be wrong. Common false-positive cases include:

The method is intended to trigger loading of some class which may have a side effect.
The method is intended to implicitly throw some obscure exception.

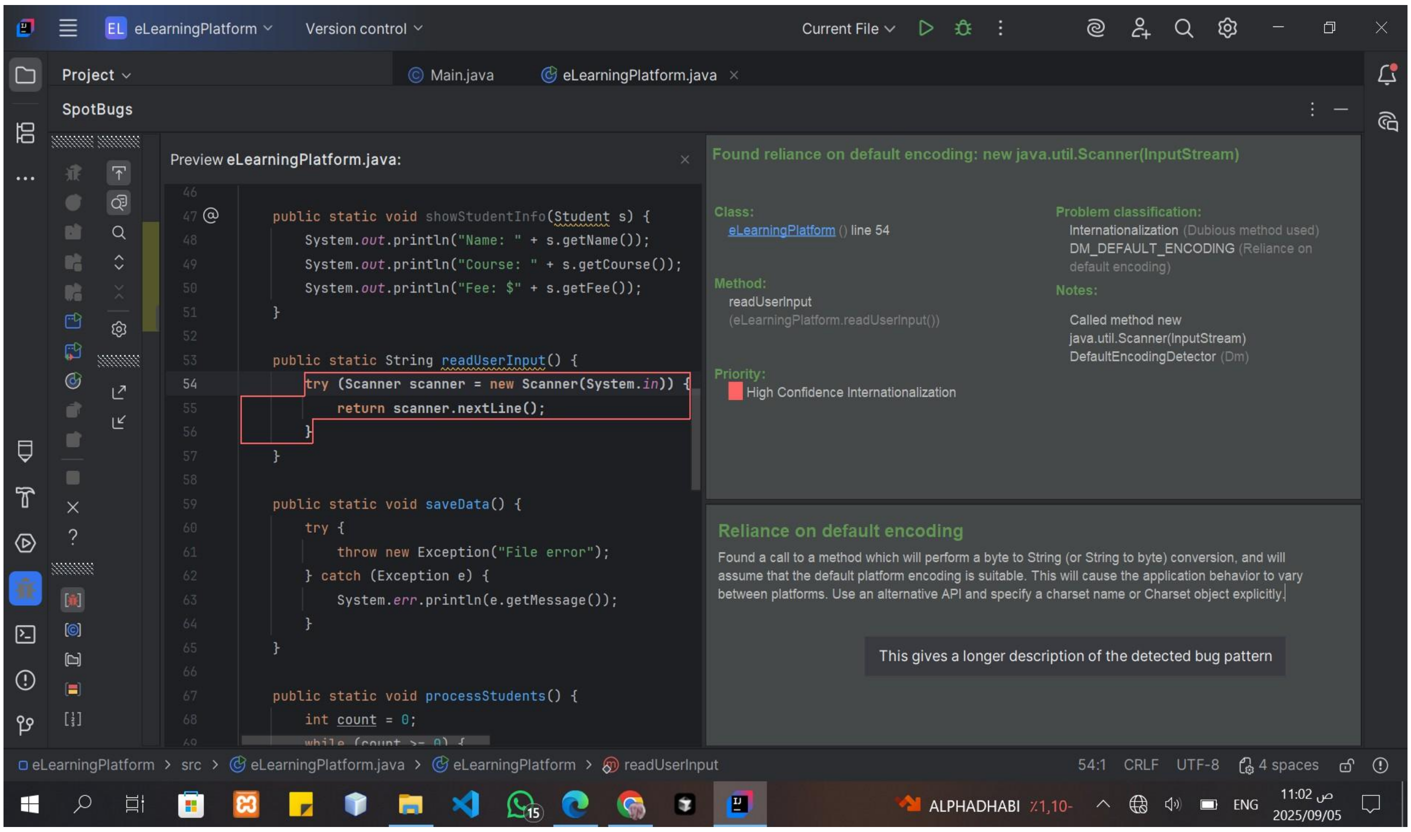
After: Fixed the loop condition and added meaningful output - now it actually processes and prints student information, making it useful.



```java
        }
    }

    public static void processStudents() {
        final double STD_NUM = 20;
        int count = 0;
        while (count < STD_NUM) {
            count++;
            System.out.println("Processing student number: " + count);
        }
        System.out.println("Processing DONE" + count + "STUDENTS");
    }
```

Bug 3: reliance on default encoding

Before: Using `new Scanner(System.in)` with default encoding , SpotBugs warned about "reliance on default encoding" which can cause issues across different platforms.



```java
    public static void showStudentInfo(Student s) {
        System.out.println("Name: " + s.getName());
        System.out.println("Course: " + s.getCourse());
        System.out.println("Fee: $" + s.getFee());
    }

    public static String readUserInput() {
        try (Scanner scanner = new Scanner(System.in)) {
            return scanner.nextLine();
        }
    }

    public static void saveData() {
        try {
            throw new Exception("File error");
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }

    public static void processStudents() {
        int count = 0;
        while (count >= 0) {
```

Found reliance on default encoding: new java.util.Scanner(InputStream)

Class:
eLearningPlatform () line 54

Method:
readUserInput
(eLearningPlatform.readUserInput())

Priority:
🟥 High Confidence Internationalization

Problem classification:
Internationalization (Dubious method used)
DM_DEFAULT_ENCODING (Reliance on default encoding)

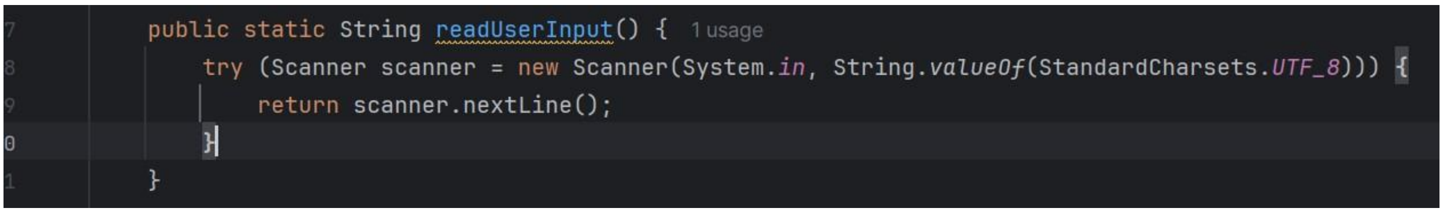
Notes:
Called method new java.util.Scanner(InputStream)
DefaultEncodingDetector (Dm)

Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behavior to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

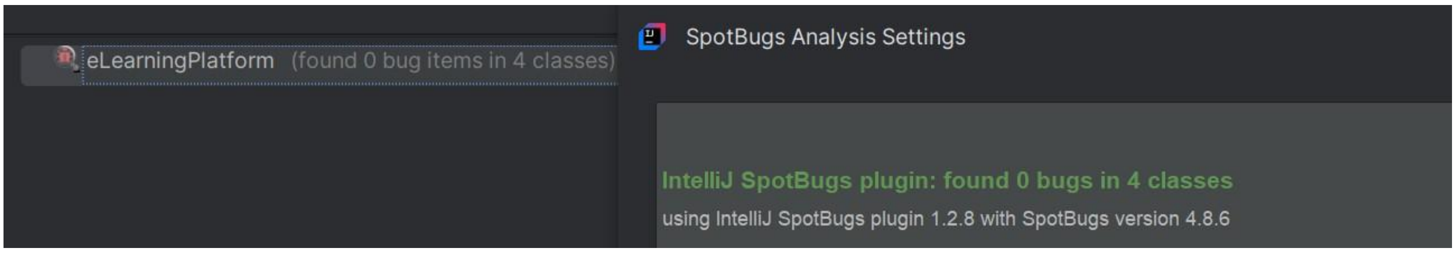
This gives a longer description of the detected bug pattern

After: Fixed by explicitly specifying UTF-8 charset - `Scanner(System.in,String.valueOf(StandardCharsets.UTF_8))`



```
    public static String readUserInput() {  1 usage
        try (Scanner scanner = new Scanner(System.in, String.valueOf(StandardCharsets.UTF_8))) {
            return scanner.nextLine();
        }
    }
```

After: applying spotBugs analysis and detect bugs



eLearningPlatform (found 0 bug items in 4 classes)

SpotBugs Analysis Settings

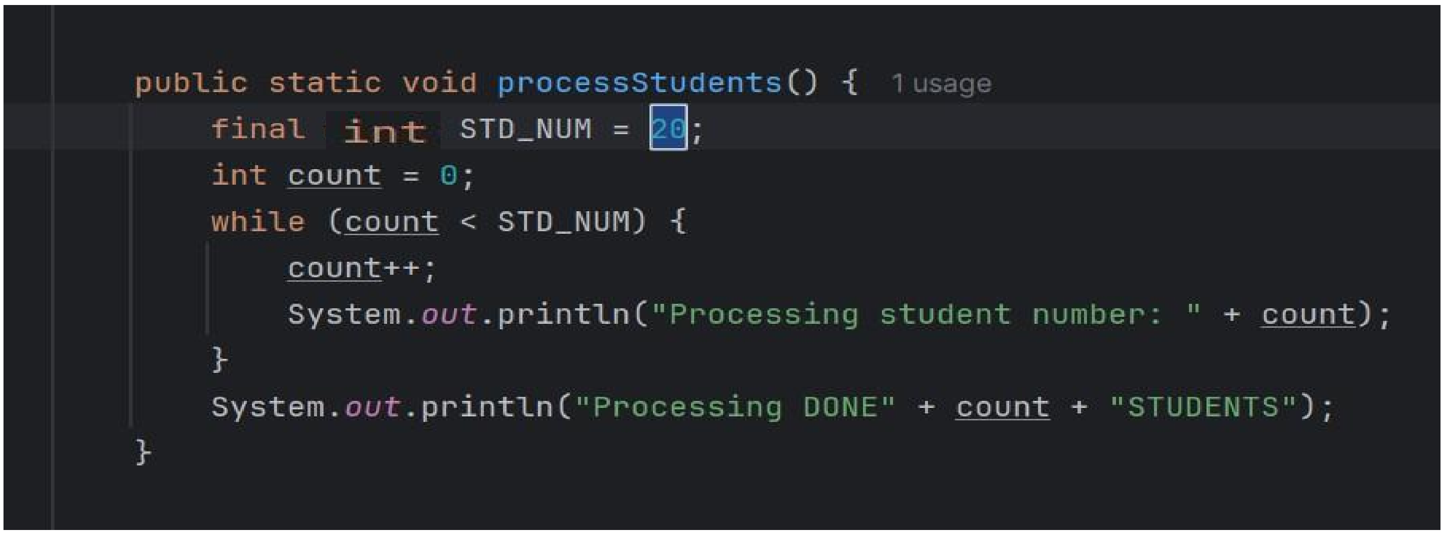
IntelliJ SpotBugs plugin: found 0 bugs in 4 classes

using IntelliJ SpotBugs plugin 1.2.8 with SpotBugs version 4.8.6

Detect different types of code smells and use Refectory to refactor them.

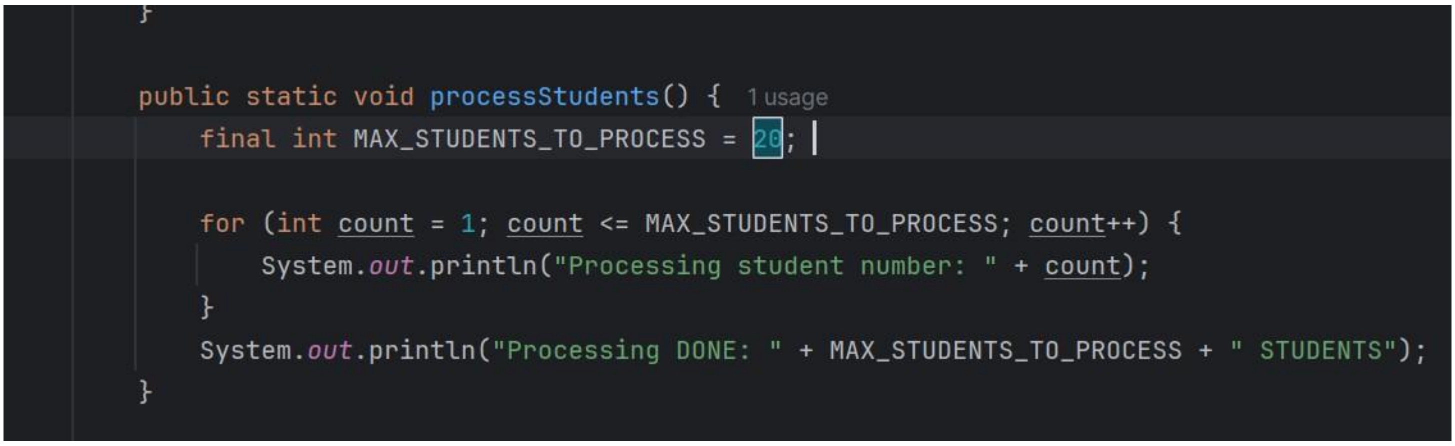
Code smell 1:Magic Number

Before :Using raw numbers directly in the code makes it unclear what they represent



```
public static void processStudents() {  1 usage
    final  int  STD_NUM = 20;
    int count = 0;
    while (count < STD_NUM) {
        count++;
        System.out.println("Processing student number: " + count);
    }
    System.out.println("Processing DONE" + count + "STUDENTS");
}
```

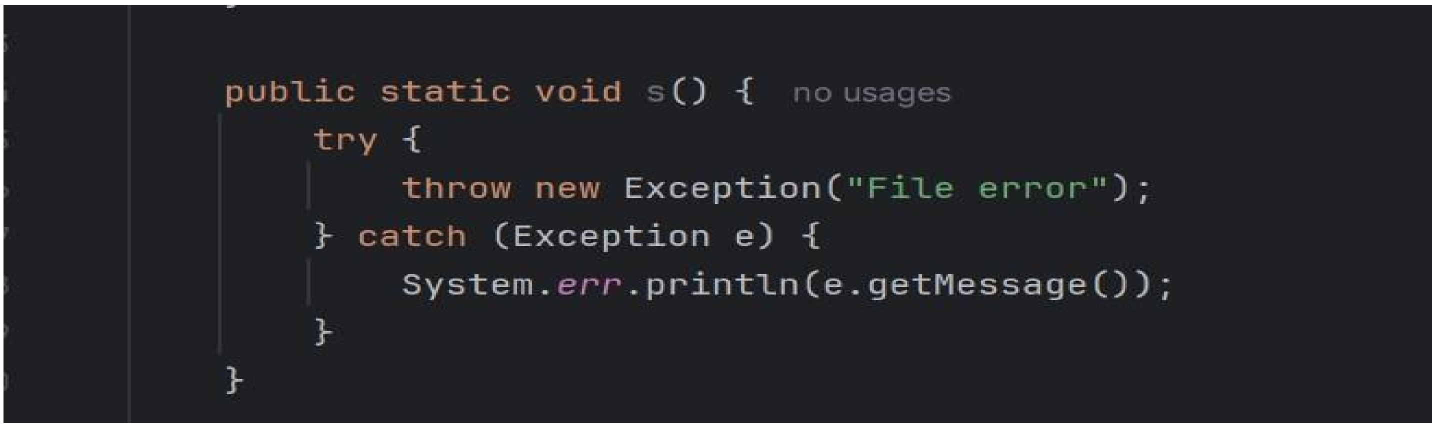
After: Replacing them with named constants improves readability and maintainability



```
}

public static void processStudents() {  1 usage
    final int MAX_STUDENTS_TO_PROCESS = 20;

    for (int count = 1; count <= MAX_STUDENTS_TO_PROCESS; count++) {
        System.out.println("Processing student number: " + count);
    }
    System.out.println("Processing DONE: " + MAX_STUDENTS_TO_PROCESS + " STUDENTS");
}
```

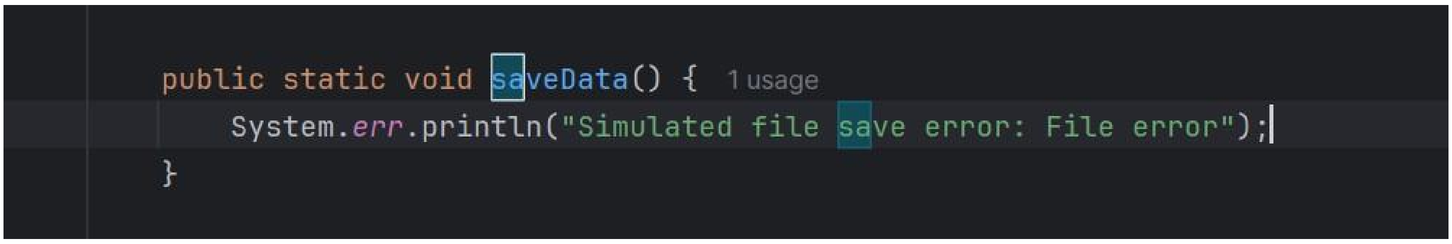
Code smell 2: Unclear method name & Exception Handling Smell

Before : the `s()` method throws an exception and immediately catches it inside the same method and has unclear name



```
public static void s() {  no usages
    try {
        throw new Exception("File error");
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
```

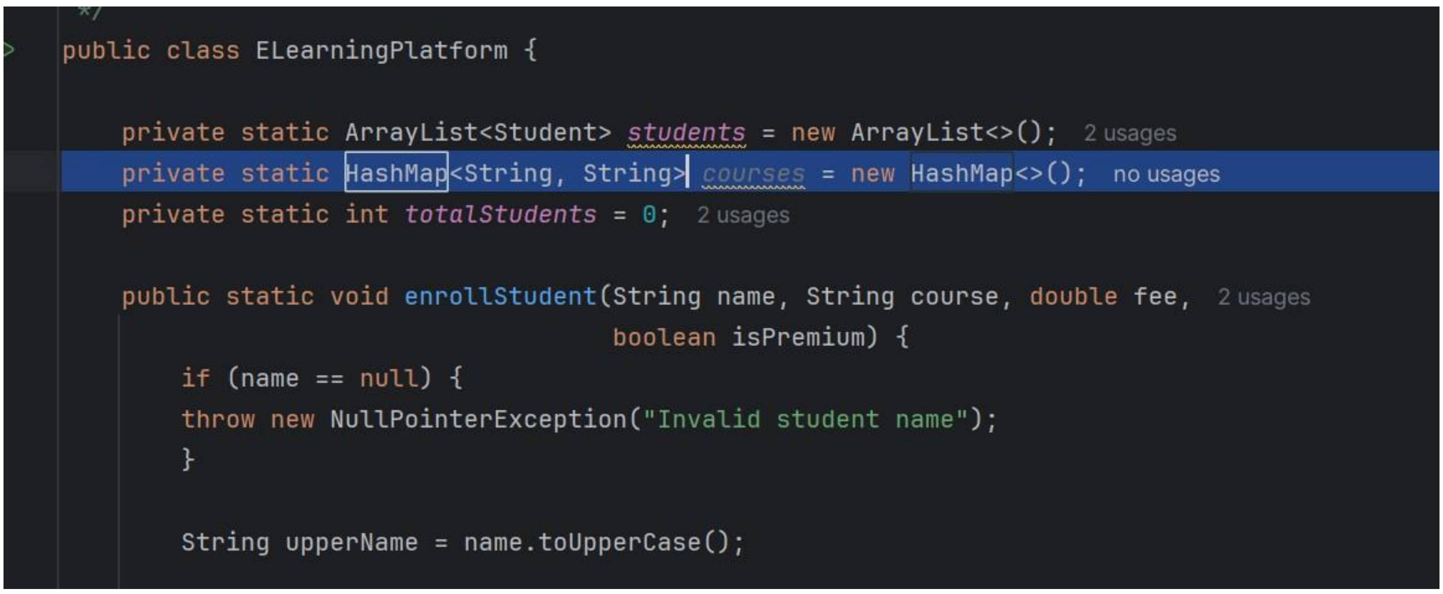
After Refactoring : easier to understand, and avoids creating meaningless exceptions



```
    public static void saveData() {  1 usage
        System.err.println("Simulated file save error: File error");
    }
```

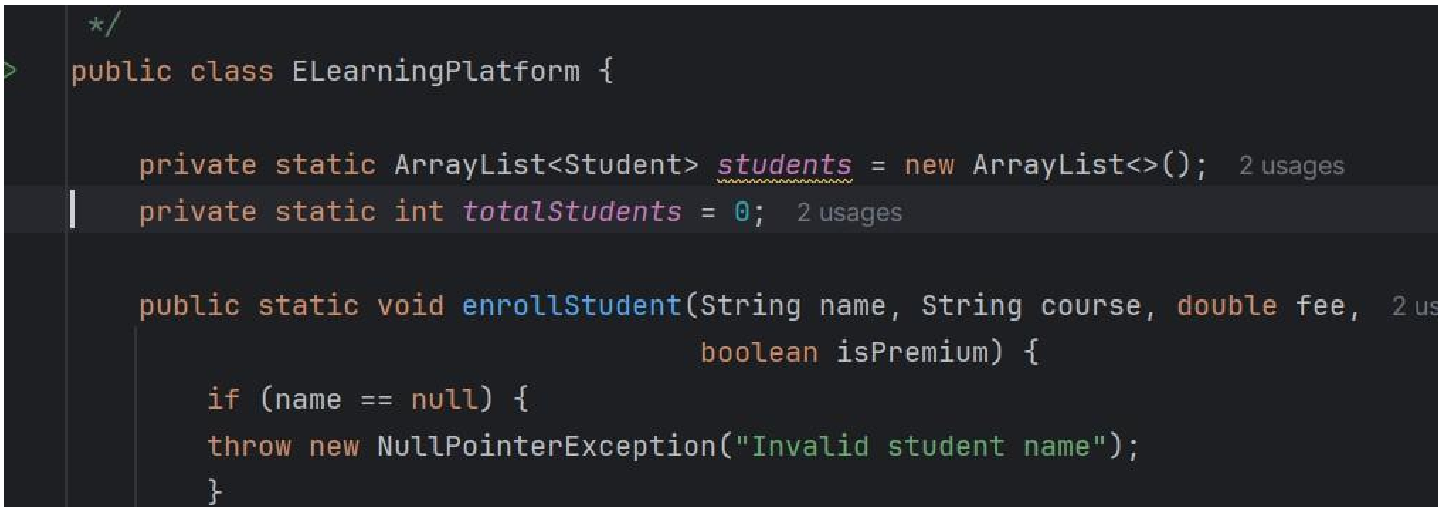
Code smell 3: Inline Method (Unused Code)

Before: A `HashMap<String, String> courses` was declared inside the class but never used.



```
public class ELearningPlatform {

    private static ArrayList<Student> students = new ArrayList<>();  2 usages
    private static HashMap<String, String> courses = new HashMap<>();  no usages
    private static int totalStudents = 0;  2 usages

    public static void enrollStudent(String name, String course, double fee,  2 usages
                                     boolean isPremium) {
        if (name == null) {
        throw new NullPointerException("Invalid student name");
        }

        String upperName = name.toUpperCase();
```

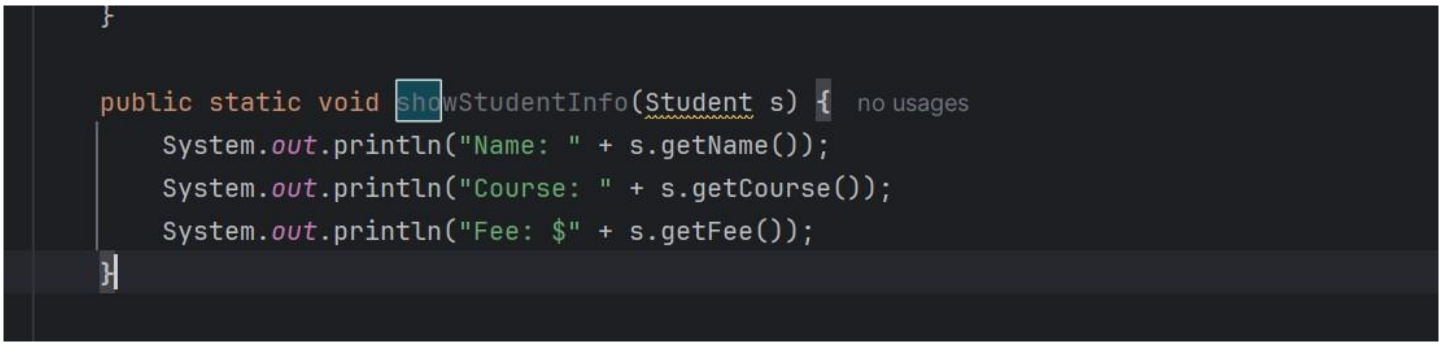
After refactoring : removed unused code (hashMap) to make the class cleaner and more focused



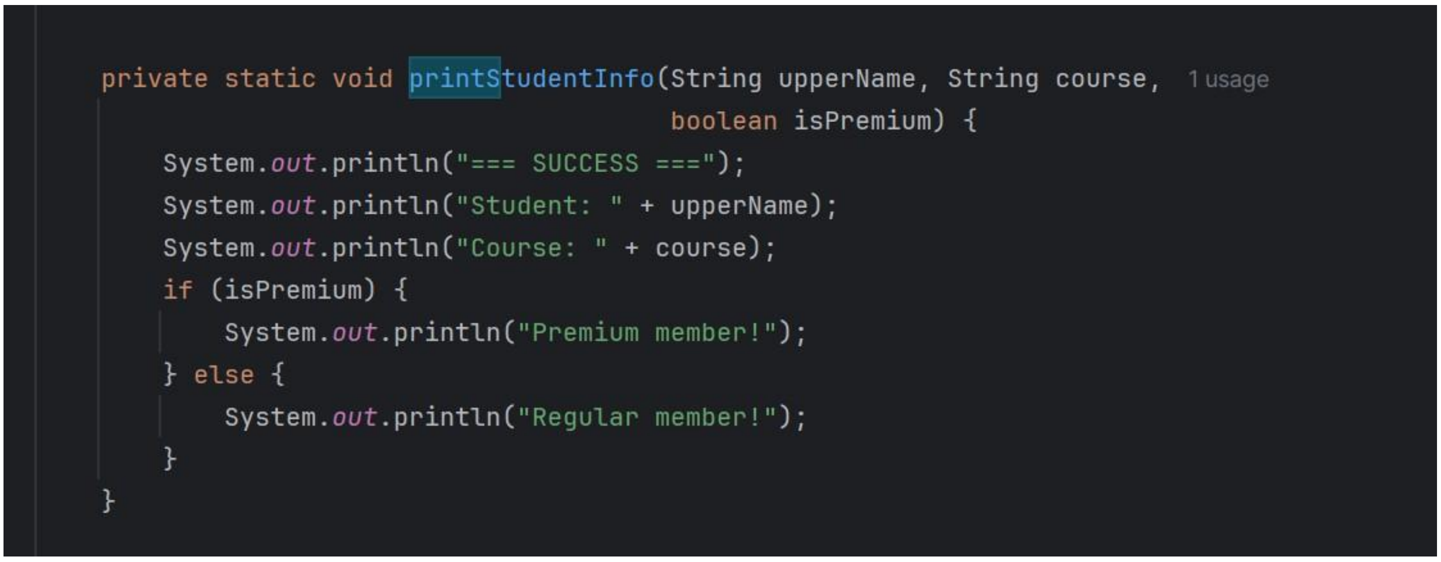
```
 */
public class ELearningPlatform {

    private static ArrayList<Student> students = new ArrayList<>();  2 usages
    private static int totalStudents = 0;  2 usages

    public static void enrollStudent(String name, String course, double fee,  2 us
                                     boolean isPremium) {
        if (name == null) {
        throw new NullPointerException("Invalid student name");
        }
```

Code Smell 4: Printing Logic Mixed with Business Logic

Before: Printing logic was directly implemented inside `ELearningPlatform` (methods `printStudentInfo` and `showStudentInfo`), mixing presentation with student management.

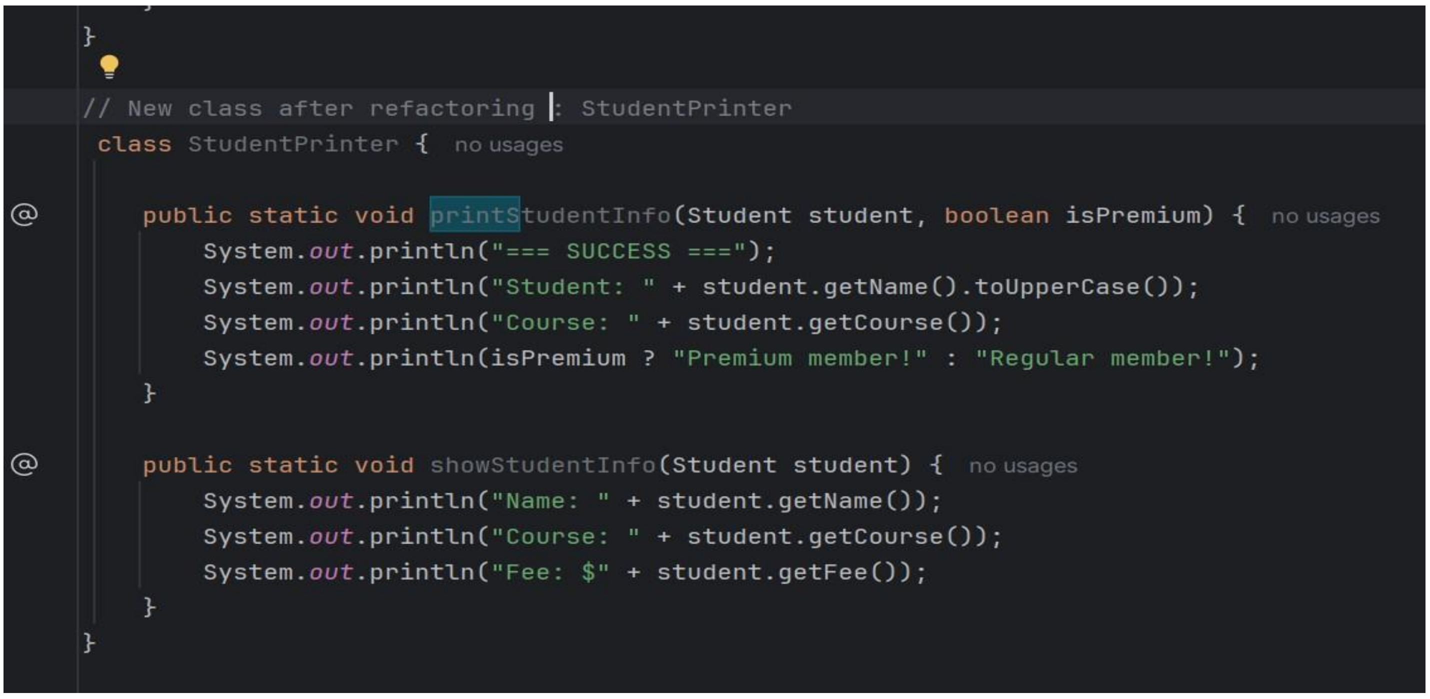


```java
    }

    public static void showStudentInfo(Student s) {  no usages
        System.out.println("Name: " + s.getName());
        System.out.println("Course: " + s.getCourse());
        System.out.println("Fee: $" + s.getFee());
    }
```



```java
    private static void printStudentInfo(String upperName, String course,  1 usage
                                         boolean isPremium) {
        System.out.println("=== SUCCESS ===");
        System.out.println("Student: " + upperName);
        System.out.println("Course: " + course);
        if (isPremium) {
            System.out.println("Premium member!");
        } else {
            System.out.println("Regular member!");
        }
    }
```

After (Refactoring)**:** Moved printing logic to a new `StudentPrinter` class, separating **presentation** from **business logic** and improving maintainability.



```java
}

// New class after refactoring : StudentPrinter
class StudentPrinter {  no usages

    public static void printStudentInfo(Student student, boolean isPremium) {  no usages
        System.out.println("=== SUCCESS ===");
        System.out.println("Student: " + student.getName().toUpperCase());
        System.out.println("Course: " + student.getCourse());
        System.out.println(isPremium ? "Premium member!" : "Regular member!");
    }

    public static void showStudentInfo(Student student) {  no usages
        System.out.println("Name: " + student.getName());
        System.out.println("Course: " + student.getCourse());
        System.out.println("Fee: $" + student.getFee());
    }
}
```