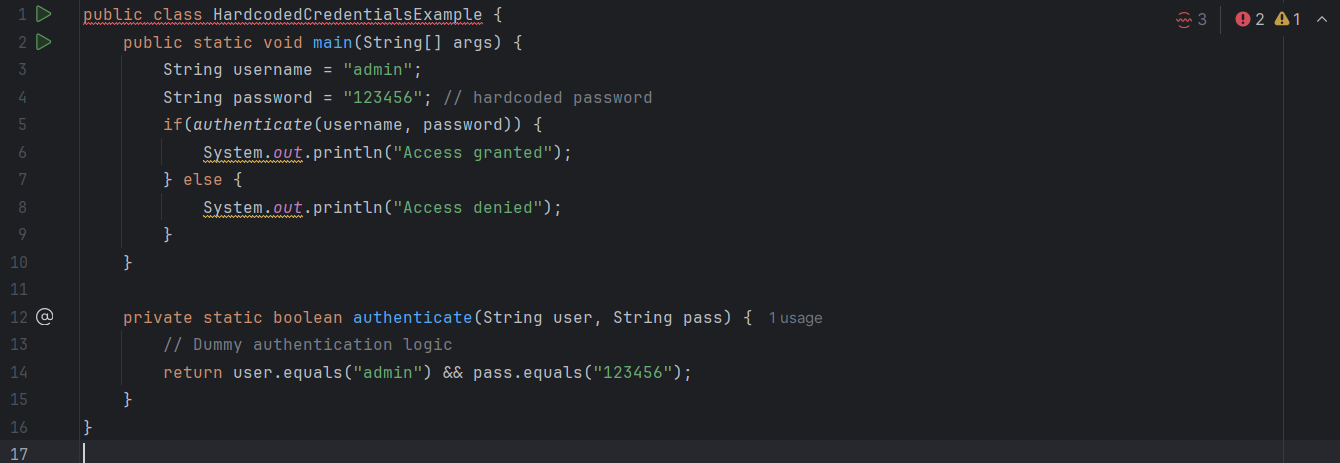
DE190916

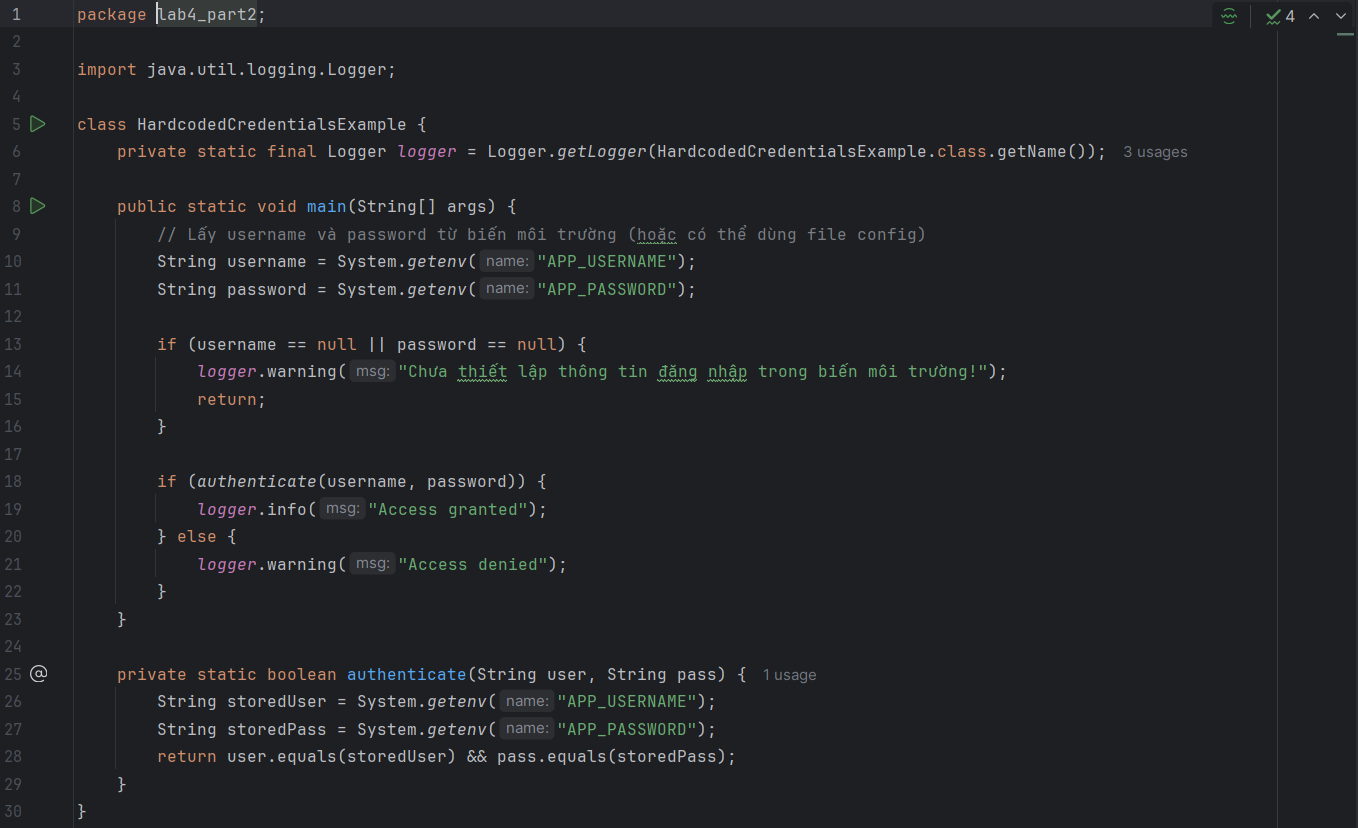
Câu 1:HardcodedCredentialsExample



Một số lỗi xảy ra:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| System.out.println(…)  (dòng 6,8) | Replace this use of “System.out” by a logger | Không nên dùng System.out trong ứng dụng thực, vì khó kiểm soát log, không lưu file được | Dùng logger như java.util.logging.Logger |
| password = "123456" (dòng 4) | Hardcoded password | Mật khẩu được ghi cứng trong code dễ bị lộ thông tin nhạy cảm | Đọc mật khẩu từ biến môi trường hoặc file cấu hình thay vì ghi trực tiếp |
| authenticate() (dòng 12-14) | Dummy authentication logic | Hàm xác thực chỉ so sánh chuỗi, không có cơ chế bảo mật thật sự | Thay bằng xác thực đúng chuẩn, ví dụ kiểm tra hash mật khẩu từ DB |
| username = "admin" (dòng 3) | Hardcoded username | Tài khoản admin viết cứng trong code gây rủi ro bảo mật | Đọc từ input hoặc config, không hardcode |
| HardcodedCredentialsExample class | Security issue | Class demo có thể được hiểu là ví dụ kém an toàn | Giữ cho mục đích học hoặc đổi tên, thêm comment “for demo only” |

Sửa code:



Câu 2:PathTraversalExample



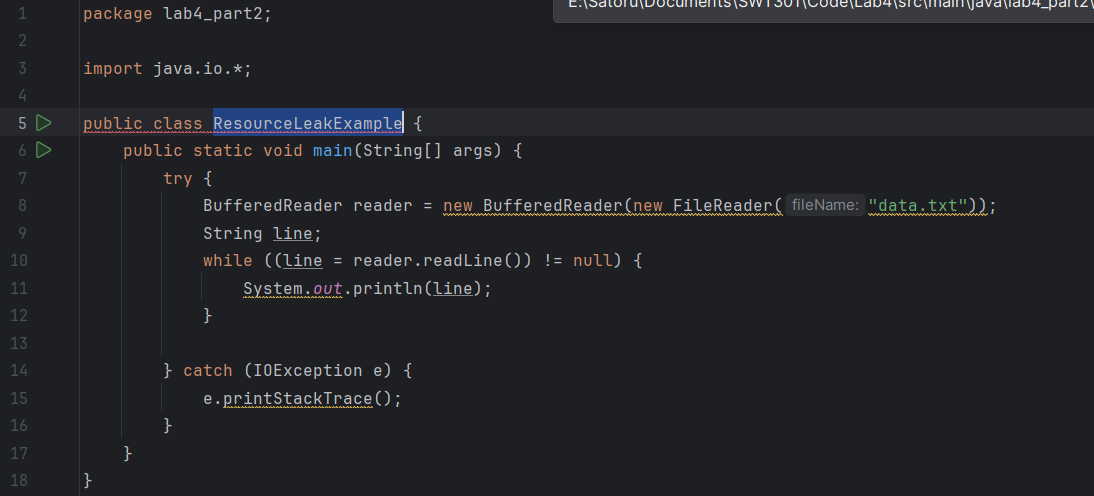
Một số lỗi xảy ra:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| String userInput = "../secret.txt"; (dòng 7) | Path Traversal | Cho phép truy cập file bằng đường dẫn tương đối, có thể vượt ra ngoài thư mục an toàn | Validate đường dẫn – chỉ cho phép đọc trong một thư mục cụ thể (vd: /data/uploads) |
| new File(userInput) (dòng 8) | Unvalidated file path | Không kiểm tra xem đường dẫn có trỏ đến vùng hợp lệ hay không | Dùng getCanonicalPath() để kiểm tra file thật sự nằm trong thư mục cho phép |
| System.out.println(...) (dòng 11) | Replace with logger | Không nên in log trực tiếp ra console trong ứng dụng thực tế | Dùng Logger như java.util.logging.Logger |
| reader.close(); (dòng 12) | Resource leak risk | Nếu có exception xảy ra trước khi close, reader sẽ không được đóng | Dùng try-with-resources để tự động đóng reader |

Sửa code:



Câu 3: ResourceLeakExample



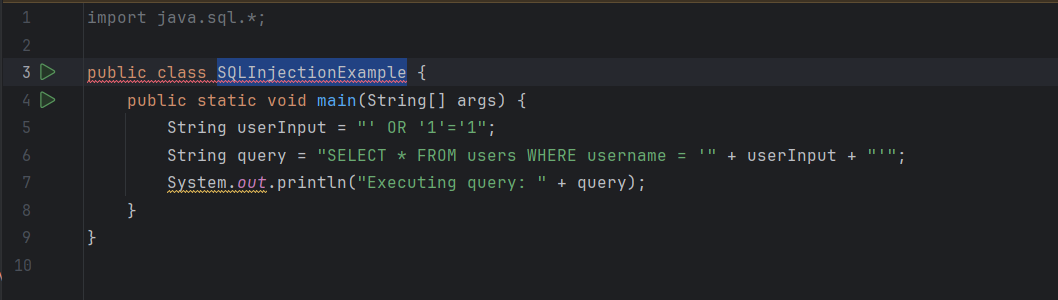
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| BufferedReader reader = new BufferedReader(new FileReader("data.txt")); (dòng 8) | Resource leak (Tài nguyên không được đóng) | reader mở file nhưng không được đóng nếu có exception xảy ra → gây rò rỉ file descriptor | **Dùng try-with-resources** để tự động đóng reader sau khi dùng |
| System.out.println(line); (dòng 11) | Replace System.out by logger | Không nên in log trực tiếp ra console, khó kiểm soát trong ứng dụng lớn | **Dùng Logger** (java.util.logging.Logger |
| e.printStackTrace(); (dòng 13) | In trực tiếp stack trace | Không an toàn vì có thể lộ thông tin nội bộ khi chạy thực tế | **Log lỗi qua logger hoặc hiển thị thông báo người dùng thân thiện** |

Sửa code:



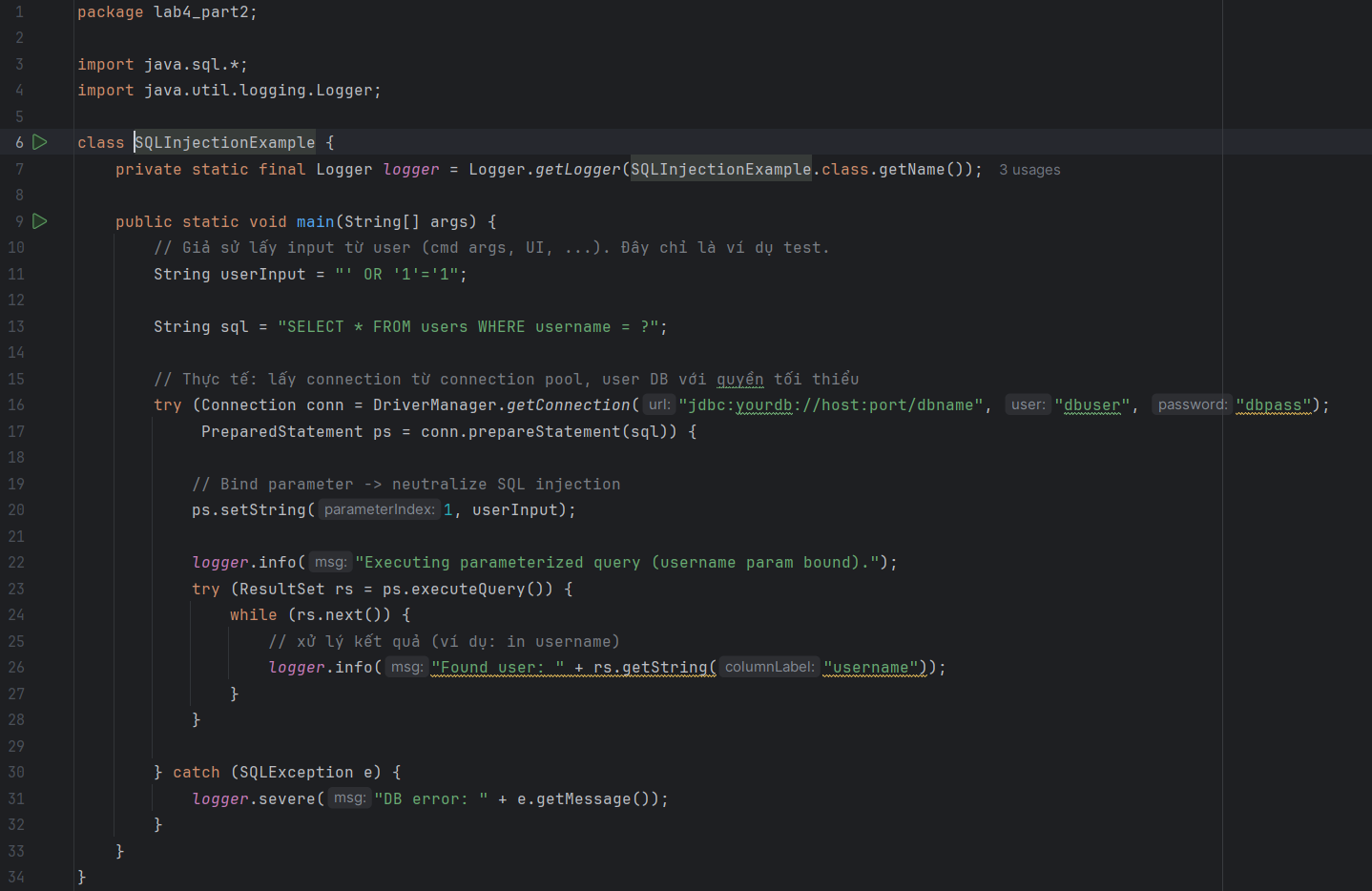
Câu 4:SQLInjectionExample



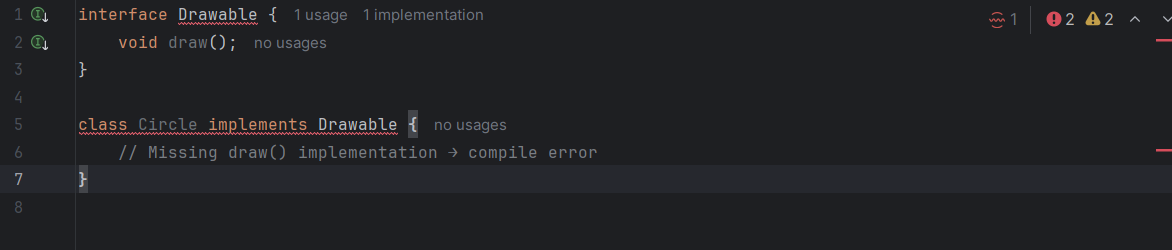
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| String query = "SELECT \* FROM users WHERE username = '" + userInput + "'"; | SQL Injection (concatenation) | Nối trực tiếp input người dùng vào câu SQL → kẻ xấu chèn payload như ' OR '1'='1 và bypass auth / rút dữ liệu | Dùng **PreparedStatement** (parameterized query) hoặc ORM; KHÔNG bao giờ dùng concat để tạo SQL với input người dùng |
| System.out.println("Executing query: " + query); | Replace System.out by logger / leak thông tin | In thẳng câu SQL ra console dễ lộ thông tin nội bộ / query structure | Dùng Logger (ghi log ở mức phù hợp — debug/info) hoặc không log được raw SQL với dữ liệu nhạy cảm |
| String userInput = "' OR '1'='1"; | Hardcoded malicious example in prod code | Nếu để kiểu hardcode test trong prod có thể gây nhầm lẫn — nhưng trong lab ok | Chỉ dùng test data trong test, hoặc đọc input từ nguồn an toàn (input, var env, mock) |
| (tổng thể) | Thiếu input validation & least privilege | Không validate input, không áp chế quyền DB | Áp dụng validation/whitelisting; DB user nên có quyền tối thiểu; sử dụng parameterized query + rate-limit + prepared statements |

Sửa code:



Câu 5:UnimplementedInterfaceExample



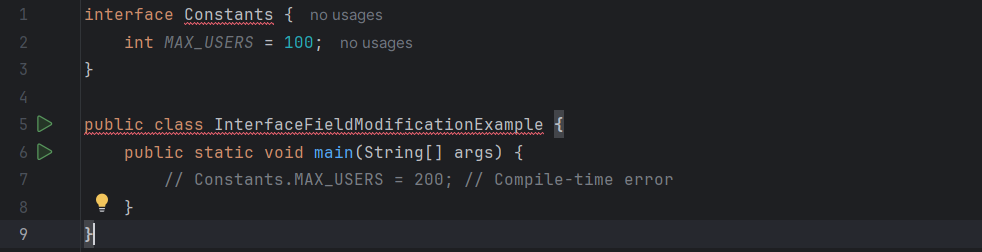
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| class Circle implements Drawable (dòng 5) | Missing method implementation | Khi một class implements interface mà không override tất cả method, Java báo lỗi biên dịch (Class 'Circle' must either be declared abstract or implement abstract method 'draw()' in 'Drawable') | **Thêm phương thức draw()** hoặc **khai báo class là abstract** nếu không muốn implement |

Sửa code:



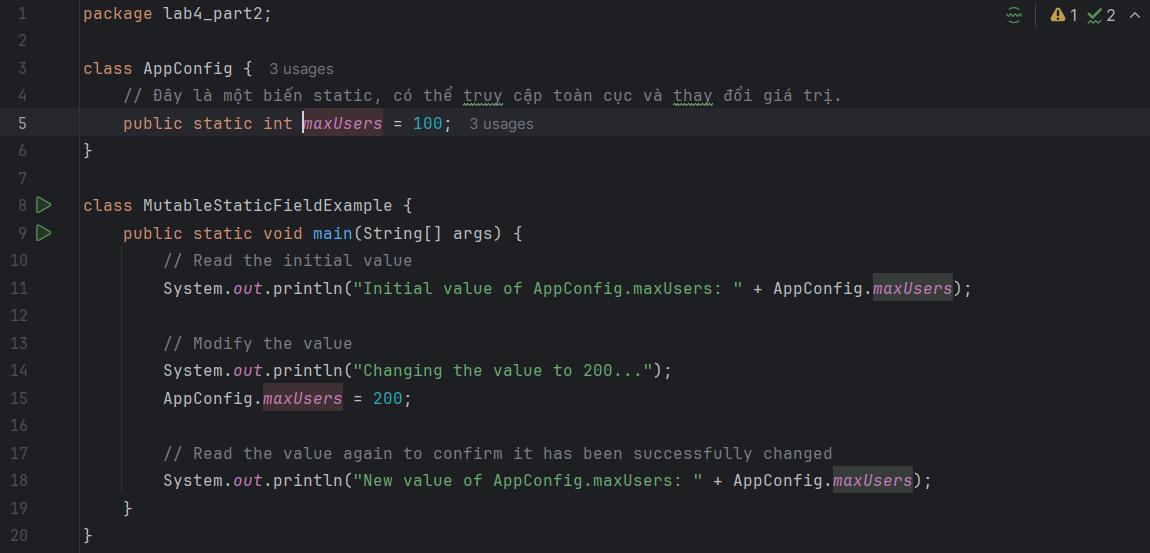
Câu 6:InterfaceFieldModificationExample



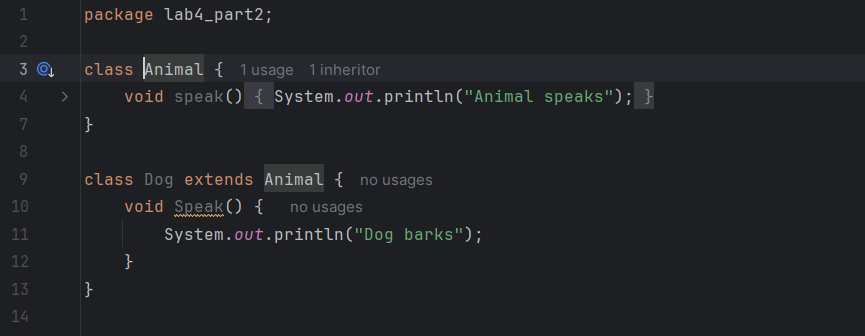
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| int MAX\_USERS = 100; trong interface Constants | Attempt to modify constant | Fields trong interface mặc định public static final → không thể gán lại. Nếu cố gán Constants.MAX\_USERS = 200; sẽ **compile error**. | Nếu cần value có thể thay đổi, **đừng dùng interface** cho biến mutable. Dùng class với field static (không final) hoặc cấu hình (file, DB, env var) |

Sửa code:



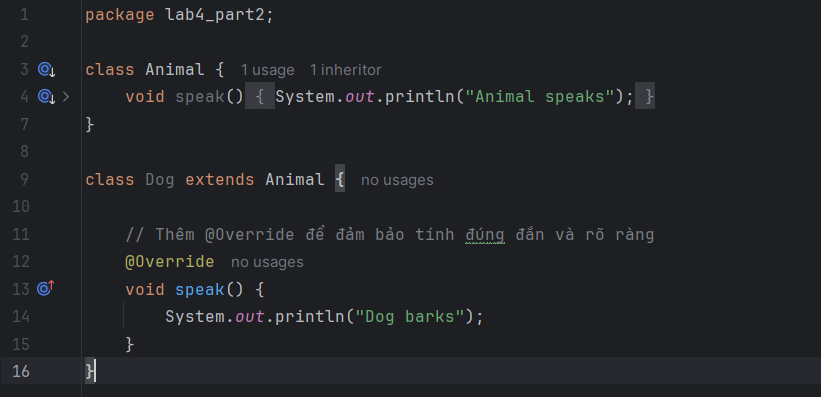
Câu 7:



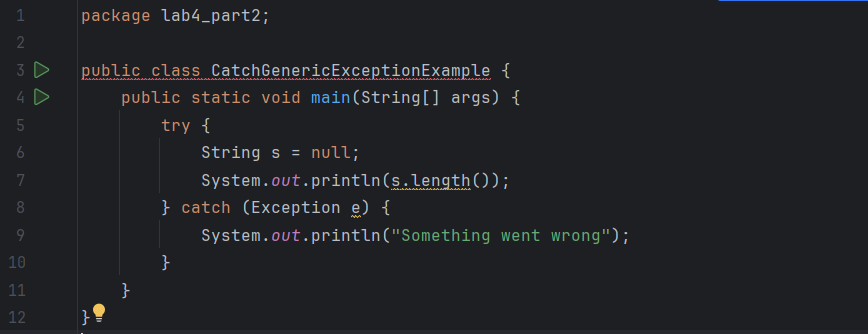
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| void speak() trong class Dog | Thiếu Annotation @Override | Mặc dù code vẫn chạy, đây là một lỗi **tiềm ẩn** và là thói quen code xấu. Việc không dùng @Override khiến trình biên dịch không thể kiểm tra giúp bạn xem có thực sự đang ghi đè phương thức của lớp cha hay không. Nếu bạn vô tình gõ sai tên (ví dụ: spek() thay vì speak()), chương trình sẽ không báo lỗi và sẽ gọi nhầm phương thức của lớp Animal, gây ra bug khó tìm. | Thêm annotation @Override ngay trên phương thức speak() của lớp Dog. |

Sửa code:



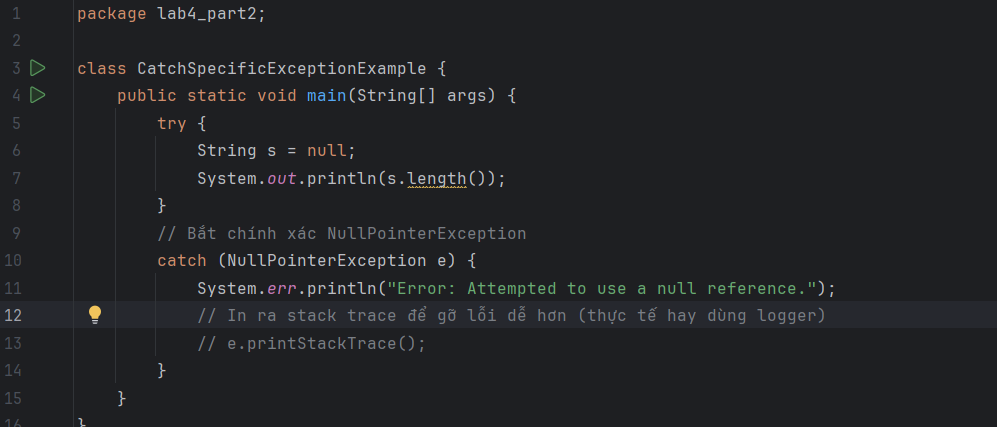
Câu 8:



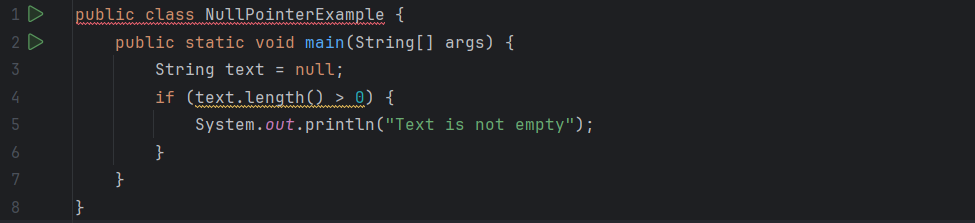
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| catch (Exception e) | Bắt (Catch) kiểu ngoại lệ quá chung chung | Việc bắt Exception sẽ che giấu đi lỗi cụ thể đang xảy ra (trong trường hợp này là NullPointerException). Nếu khối try có thể phát sinh nhiều loại lỗi khác nhau (ví dụ ArrayIndexOutOfBoundsException), tất cả sẽ bị xử lý theo cùng một cách, gây khó khăn cho việc gỡ lỗi và xử lý lỗi một cách chính xác. Đây là một **thói quen code xấu** (bad practice). | Bắt (catch) loại ngoại lệ cụ thể nhất có thể. Trong trường hợp này, hãy thay thế Exception bằng NullPointerException. |

Sửa code:



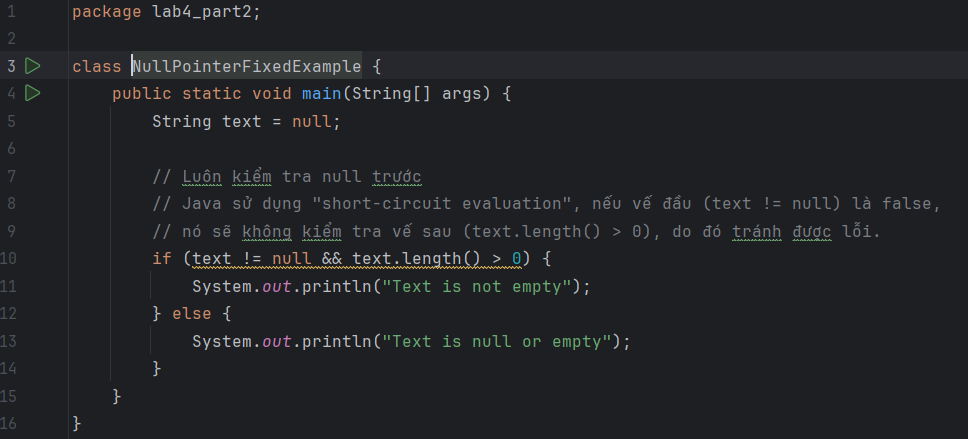
Câu 9:



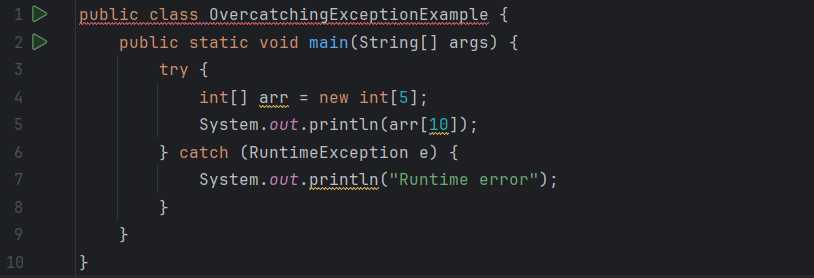
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| if (text.length() > 0) | Gây ra NullPointerException | Biến text đã được gán giá trị là null, có nghĩa là nó không trỏ đến bất kỳ đối tượng chuỗi nào. Việc gọi phương thức .length() trên một tham chiếu null sẽ ngay lập tức gây ra ngoại lệ NullPointerException tại thời điểm chạy (runtime exception), và chương trình sẽ dừng lại. | Luôn kiểm tra xem một đối tượng có null hay không **trước khi** gọi bất kỳ phương thức nào của nó. Điều kiện đúng phải là text != null và sau đó mới kiểm tra đến độ dài. |

Sửa code:



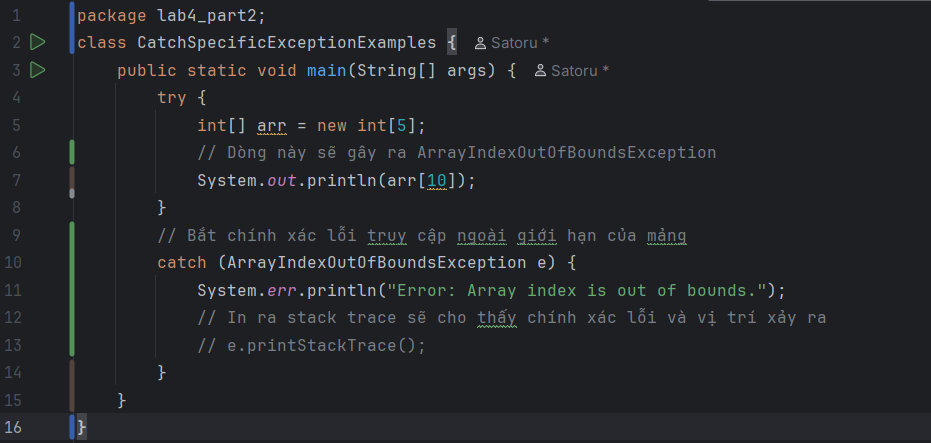
Câu 10:



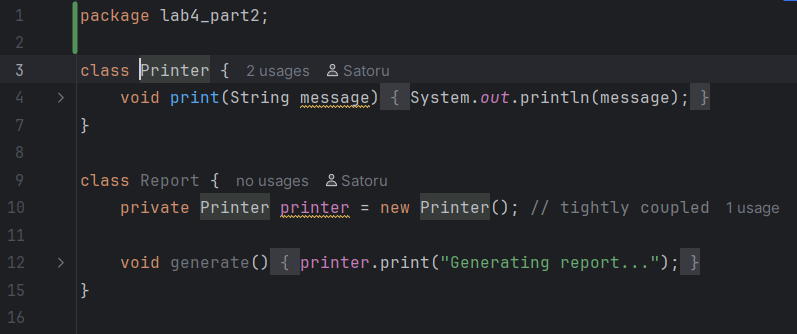
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| catch (RuntimeException e) | Bắt (Catch) ngoại lệ quá chung chung | Mặc dù ArrayIndexOutOfBoundsException (lỗi sẽ xảy ra ở dòng arr[10]) là một loại của RuntimeException, việc bắt một cách chung chung như vậy sẽ che giấu đi bản chất thật của lỗi. Nếu trong khối try có thể xảy ra nhiều loại RuntimeException khác nhau (ví dụ NullPointerException), tất cả chúng sẽ rơi vào cùng một khối catch, làm mất đi khả năng xử lý lỗi một cách tường minh và gây khó khăn cho việc gỡ lỗi. | Hãy bắt (catch) ngoại lệ cụ thể nhất có thể. Trong trường- hợp này, hãy thay RuntimeException bằng ArrayIndexOutOfBoundsException |

Sửa code:



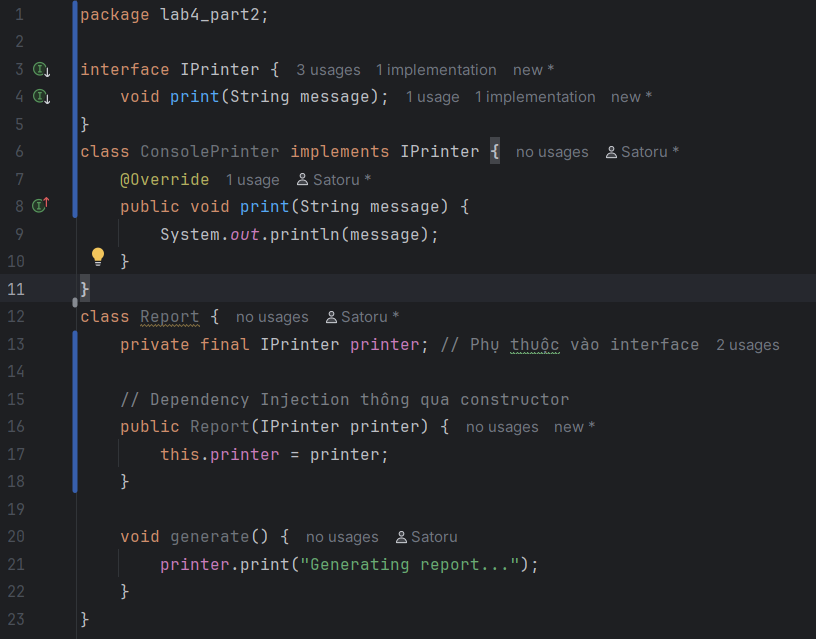
Câu 11:



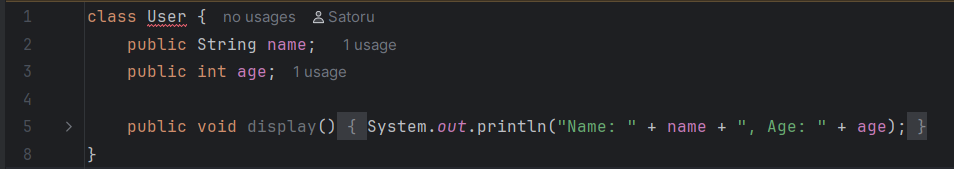
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| private Printer printer = new Printer(); | **Coupling Chặt** (Tight Coupling) | Lớp Report bị phụ thuộc cứng vào lớp cụ thể là Printer. Nó tự chịu trách nhiệm tạo ra đối tượng printer. Điều này làm cho code rất khó để kiểm thử (unit test) và khó mở rộng. Nếu sau này bạn muốn Report in ra file (FilePrinter) hoặc qua mạng (NetworkPrinter), bạn sẽ phải sửa đổi trực tiếp code của lớp Report. | Áp dụng nguyên tắc **Đảo ngược Phụ thuộc** (Dependency Inversion Principle) và kỹ thuật **Tiêm Phụ thuộc** (Dependency Injection). Report không nên tự tạo Printer, mà nên được cung cấp (tiêm) một đối tượng Printer từ bên ngoài, tốt nhất là thông qua một interface. |

Sửa code:

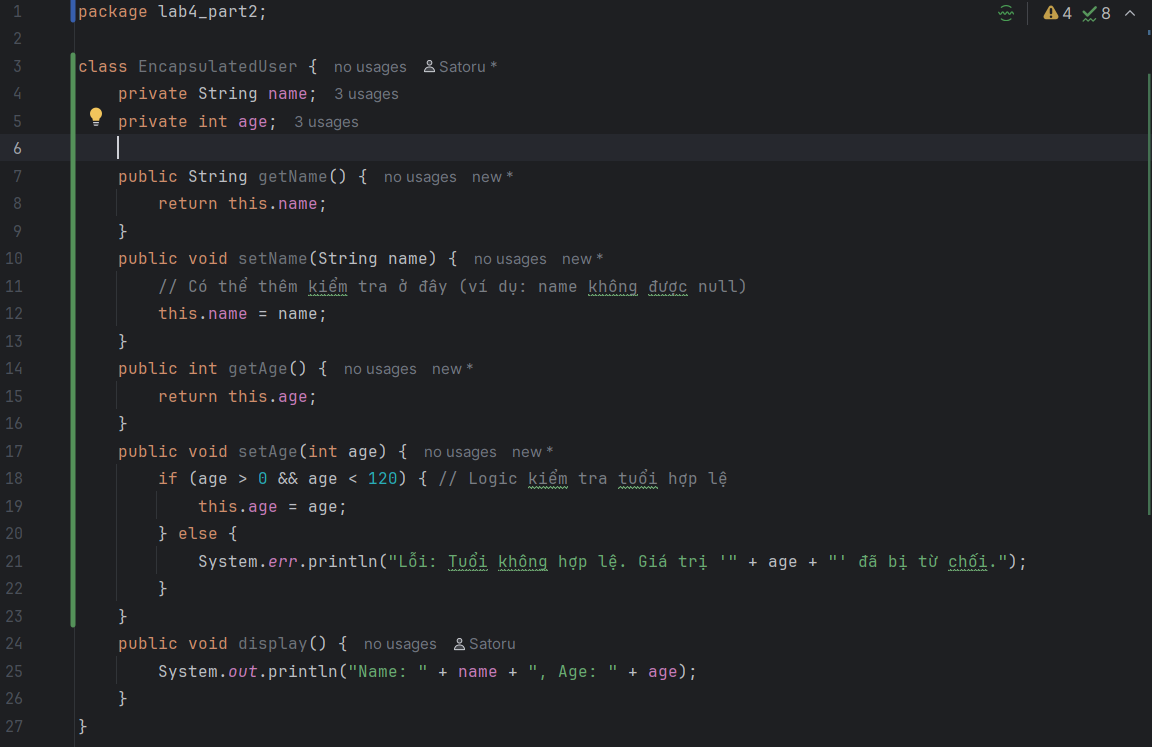


Câu 12:

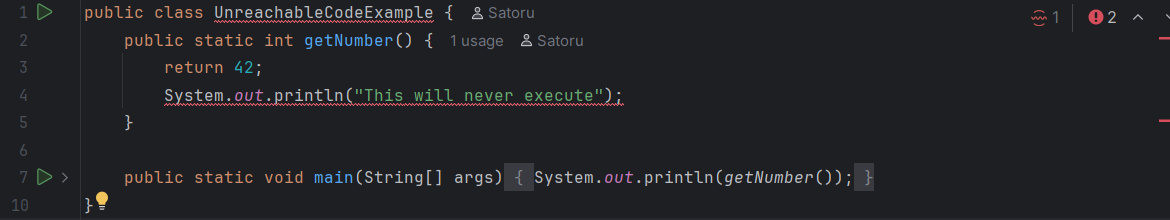


Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| public String name; public int age; | **Vi phạm Tính Đóng Gói** (Violation of Encapsulation) | Các trường (fields) được khai báo là public, cho phép bất kỳ đoạn code nào bên ngoài lớp có thể truy cập và sửa đổi chúng một cách trực tiếp. Điều này phá vỡ nguyên tắc che giấu thông tin. Lớp User hoàn toàn mất kiểm soát đối với trạng thái bên trong của nó. Ví dụ, một người dùng khác có thể dễ dàng gán một giá trị vô lý như user.age = -50 mà không có bất kỳ sự kiểm tra nào, dẫn đến dữ liệu không hợp lệ. | Chuyển phạm vi truy cập của các trường thành private. Sau đó, cung cấp các phương thức public (được gọi là **getters** và **setters**) để cho phép truy cập và sửa đổi các trường này một cách có kiểm soát. Bên trong các setter, bạn có thể thêm logic xác thực (validation) để đảm bảo tính toàn vẹn của dữ liệu. |

Sửa code:

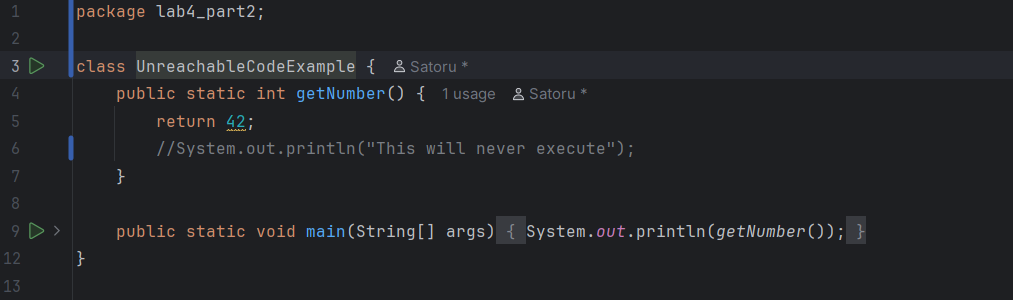
Câu 13:



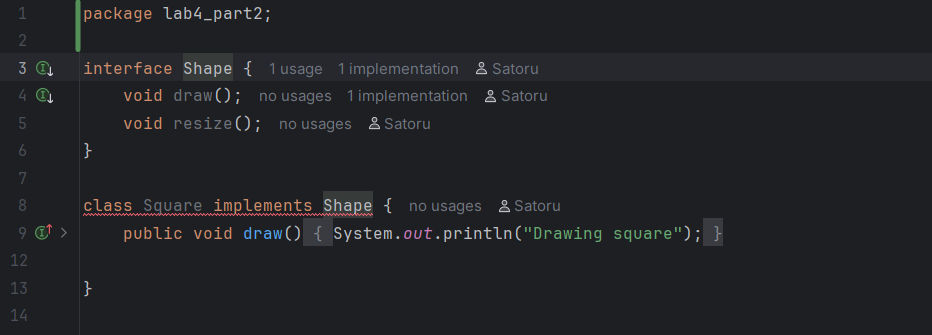
Một số lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| System.out.println("This will never execute"); | **Mã không thể truy cập** (Unreachable Code) | Lệnh return 42; ngay lập tức kết thúc phương thức getNumber() và trả về một giá trị. Bất kỳ đoạn mã nào trong cùng một khối lệnh mà nằm sau câu lệnh return đều sẽ không bao giờ được thực thi. Đây là một **lỗi biên dịch** (compile-time error), trình biên dịch Java sẽ phát hiện và báo lỗi, không cho phép chương trình được biên dịch. | Xóa bỏ dòng code không thể truy cập này. Nếu logic của bạn yêu cầu dòng code đó phải chạy, bạn cần đặt nó **trước** câu lệnh return. |

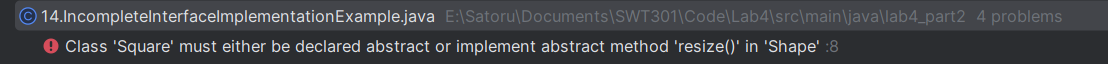
Sửa code:



Câu 14:



Một số lỗi:

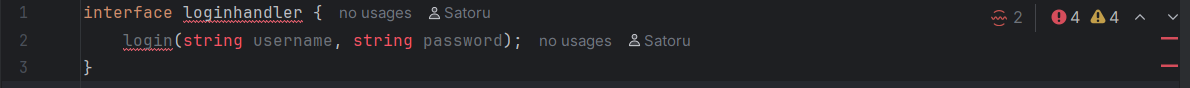


|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| class Square implements Shape | **Triển khai interface không đầy đủ** (Incomplete Interface Implementation) | Khi một lớp cụ thể (concrete class) implements một interface, nó bắt buộc phải cung cấp một triển khai (phần thân code) cho **tất cả** các phương thức được định nghĩa trong interface đó. Lớp Square đã triển khai phương thức draw() nhưng lại thiếu mất phương thức resize(). Điều này gây ra một **lỗi biên dịch** (compile-time error). | Có hai cách để sửa lỗi này: 1. **Cách phổ biến nhất:** Triển khai phương thức còn thiếu (resize()) trong lớp Square. 2. **Cách khác:** Nếu bạn chưa muốn triển khai tất cả các phương thức, bạn phải khai báo lớp Square là một lớp abstract. |

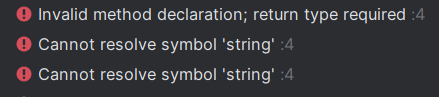
Sửa code:



Câu 15:



Một số lỗi:



|  |  |  |  |
| --- | --- | --- | --- |
| Vị trí | Lỗi | Giải thích | Cách sửa |
| interface loginhandler và login(string username, string password); | Sai quy tắc đặt tên và cú pháp không hợp lệ | 1. **Quy tắc đặt tên:** Tên interface (và class) trong Java nên theo quy tắc **PascalCase** (viết hoa chữ cái đầu mỗi từ), tức là LoginHandler. 2. **Sai kiểu dữ liệu:** Kiểu dữ liệu chuỗi trong Java là String (viết hoa chữ S), không phải string. 3. **Thiếu kiểu trả về:** Mọi phương thức trong Java (kể cả trong interface) đều phải có kiểu trả về (return type). Ví dụ: boolean để trả về trạng thái đăng nhập thành công/thất bại, hoặc void nếu không cần trả về gì. | Sửa tên interface thành LoginHandler, đổi string thành String, và thêm kiểu trả về cho phương thức login (ví dụ: boolean). |

Sửa code:

