```
import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class BadCodeJava {

    // --- Security Issues ---

    // Security Issue 1: Command injection vulnerability
    public static void insecureMethod() throws IOException {
        String userInput = "some_user_input";
        String cmd = "ls -l " + userInput;
        System.out.println("Executing: " + cmd);
        // This is a classic example of an insecure shell command execution.
        Process process = Runtime.getRuntime().exec(cmd);
        try {
            process.waitFor();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    // Security Issue 2: SQL injection vulnerability
    public static void vulnerableSql() {
        int id = 123;
        String name = "some_name";
        // The SQL query is constructed by concatenating strings directly,
        // making it vulnerable to injection attacks.
        String sql = "SELECT * FROM users WHERE id = " + id + " AND name = '" +
name + "'";
        System.out.println("Executing SQL: " + sql);

        try (Connection conn =
DriverManager.getConnection("jdbc:h2:mem:testdb");
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Security Issue 3: Directory traversal vulnerability
    public static void insecureFileAccess() throws IOException {
        String fileName = "../../../../etc/passwd";
        // The application accepts a filename from user input without
```

```
sanitization,
        // allowing access to sensitive files.
        File file = new File(fileName);
        if (file.exists()) {
            System.out.println("File found: " + file.getCanonicalPath());
            // In a real scenario, this would read the file's contents.
        }
    }

    // --- Syntax and Style Issues ---

    // A method with multiple syntax and style errors
    public static int syntaxErrorMethod(int a, int b) {
        int result = a + b; // Syntax Issue 1: Missing semicolon is a Java
compiler error.

        reutrn result; // Syntax Issue 2: Incorrect keyword usage ('return' is
misspelled).

        System.out.println("This is a syntax error"; // Syntax Issue 3:
Unmatched parenthesis.
    }

    // Method with logical and style issues
    public static void confusingCode(boolean flag) {
        if (flag == true) { // Style Issue 1: Redundant comparison.
            System.out.println("Flag is true");
        } else {
            System.out.println("Flag is false");
        }

        int unusedVariable = 100; // Style Issue 2: Unused variable.
    }

    public static void main(String[] args) throws IOException {
        insecureMethod();
        vulnerableSql();
        insecureFileAccess();
        // The following calls will not compile due to syntax errors,
        // but are included to demonstrate the issues.
        // syntaxErrorMethod(1, 2);
        confusingCode(true);
    }
}
```