Assignment:

Research and Application of Source Code Analysis Tools (Snyk and Codacy) in a Java OOP Project

Task Description

Students will develop a simple student information management application using Java with Object-Oriented Programming (OOP) principles. They will then use source code analysis tools (Snyk and Codacy) to evaluate and improve their code's risk profile, performance, and adherence to coding standards. The assignment aims to teach students how to write robust Java code, apply OOP concepts, use industry-standard analysis tools, and document their findings in a professional report.

Specific Requirements

1. Develop the Java Application

Create a Java application to manage student information using a Student class with the following attributes:

- **Student ID** (integer, unique)
- **Full Name** (String, max 50 characters)
- **GPA** (double, range 0.0–4.0)

Implement the following functionality using a collection (e.g., ArrayList<Student>) to store up to 100 students:

- **Add a student**: Add a new student to the collection with input validation (e.g., unique ID, valid GPA).
- **Delete a student**: Remove a student by ID.
- **Search for a student**: Find and display a student by partial or full name (case-insensitive).
- **Display all students**: Show all students in a formatted output (e.g., table-like structure).

Requirements:

- Apply OOP principles: encapsulation (private fields, public getters/setters), abstraction, and proper class design.

- Include error handling for invalid inputs (e.g., duplicate IDs, invalid GPA) using exceptions.
- Organize the code into appropriate classes and packages (e.g., model, service, util).
- Use Java collections (e.g., ArrayList) for storing students.
- Optionally, implement a simple command-line interface for user interaction.

2. Source Code Analysis Using Snyk

Use Snyk (free tier or provided access) to analyze the Java code for risks, such as insecure libraries, null pointer dereferences, or improper exception handling.

Tasks:

- Run Snyk on your source code. If Snyk requires a repository, upload your code to a private GitHub repository.
- List at least three risks identified (or note if none were found).
- Explain each risk's cause, potential impact, and proposed fix.
- Apply fixes to the code and re-run Snyk to verify improvements.

Deliverables:

- Screenshots of Snyk analysis results (before and after fixes).
- Screenshots must include the taskbar with the current date/time and a unique identifier (e.g., student ID or username) visible in the tool's interface.
- Obscure any personal information not required for the assignment.

3. Source Code Analysis Using Codacy

Use Codacy (free tier or provided access) to analyze code quality, focusing on:

- **Coding standards** (e.g., Java naming conventions, formatting, adherence to OOP principles).
- **Code complexity** (e.g., cyclomatic complexity, method length).
- **Potential bugs** (e.g., null pointer issues, unhandled exceptions).

Tasks:

- Run Codacy on your source code (via repository or local analysis if supported).
- List at least three issues identified, explaining their cause, significance, and proposed fix.
- Apply fixes to the code and re-run Codacy to verify improvements.

Deliverables:

- Screenshots of Codacy analysis results (before and after fixes).
- Screenshots must include the taskbar with the current date/time and a unique identifier (e.g., student ID or username).
- Obscure any personal information not required.

4. Reporting Results

Write a report (500–1000 words) and submit it in both DOCX and PDF formats. The DOCX file can be created using a word processor (e.g., Microsoft Word, LibreOffice). The PDF can be generated from the DOCX or another tool of your choice. The report should include:

Introduction:

- Briefly describe Snyk and Codacy (purpose, features, and relevance to Java programming).

Analysis Results:

- Summarize issues found by Snyk and Codacy.
- Explain how each issue was resolved, including code snippets showing changes.
- Discuss improvements in code quality, risk profile, or performance.

Comparison:

- Compare code quality before and after applying fixes (e.g., number of issues, complexity metrics, or risks).
- Include a table or visual comparison if applicable.

Reflection:

- Discuss what you learned about risk-aware coding, OOP principles, and code quality in Java.
- Comment on the strengths and limitations of Snyk and Codacy for Java projects.

Screenshots:

- Embed screenshots from Snyk and Codacy analyses in both DOCX and PDF reports.

Submission Guidelines

Submit the following:

- Source code (.java files, organized in a project structure with packages).

- Report in both .docx and .pdf formats.
- Screenshots (embedded in the reports or as separate image files).

Ensure all screenshots are clear and meet the specified requirements. Submit via the designated platform (e.g., GitHub Classroom, LMS) by the deadline.

Notes

- Use the free tiers of Snyk and Codacy unless otherwise specified. Contact the instructor for access issues.
- If Snyk or Codacy requires a repository, use a private GitHub repository.
- Ensure your code compiles and runs without errors before analysis.
- Focus on learning risk-aware coding practices, OOP design, and interpreting tool feedback.
- Ensure the DOCX and PDF reports contain identical content, formatted appropriately for each file type.