Exercise 1

Group members: Eyad Hamed, Daniel Langbein, Artem Semenovykh, Sam Tadjiky

1 GQM quality model

Product:

Image database software

Goal:

Image database that works correctly and efficiently.

Question:

Can 10,000 images (average size: 10 MB) be added, deleted, updated and searched for correctly and efficiently?

Measures:

1. Number of correct operations on the database
2. Duration of operations on the database
3. Number of corrupted images

Measurement methods:

1. Count the number of failed operations
2. Measure the average response time (ms) for each operation
3. Check if any retrieved images are corrupted

Interpretation:

1. A correct database system would have 0 failed operations.
2. Add and update operations ~500ms; Delete operations ~10ms; Search operations ~15ms; etc
3. A correct database system would have 0 corrupted images.

2 Checkstyle in boulder dash

Errors:

[Declaration Order]: Reorder declarations: static above non-static, highest visibility on top.
[Visibility Modifier]: Use more restrictive visibility modifiers. If needed, create getters and setters for private variables.
[AbbreviationAsWordInName]: If it is a constant, make it static. In general this refers to having too many upper case letters in a variable name.
[HiddenField]: Rename local variables so that they don't hide fields with the same name.
[SimplifyBooleanExpression]: The result of an expression should not be compared to `false` or `true`. Instead, use `!` to negotiate an expression or remove `== true`.
[ReturnCount]: Rewrite the function to use at most one `return` statement. E.g. merge multiple if-conditions together or use a helper method for that.
[NeedBraces]: Add braces to if and else clauses.

Warnings:

[StringLiteralEquality]: The warning is displayed because the code is comparing Strings using "==" instead of using the equals method of the String class. The "==" operator only compares object references and this could cause unexpected behavior.

[EqualsAvoidNull]: The warning is displayed because you should put String literals on the left side of the .equals method, so that the equals function is called on an object which is guaranteed not to be null.

3 Integrated quality assurance in the agile process

How can quality assurance measures be seamlessly integrated into the agile development process?

Quality Assurance (QA) can be integrated into the Agile development process:

1. **Involve QA from the start:** QA should be involved from the beginning of the project. Incorporate of QA in early stages ensures that the QA team understands the product and can plan their testing strategies effectively.
2. **Cross-functional Teams:** Encourage cross-functional teams where developers and QA work closely together. This promotes better communication, faster feedback, and quicker resolution of issues.
3. **Test Early and Often:** In Agile, testing is not a phase that comes after development, but it's a continuous activity. Implement a Test-Driven Development approach where tests are written before the code. This helps in identifying issues early and makes it easier to fix them.
4. **Regular Reviews and Retrospectives**: Conduct regular code reviews to maintain code quality. Also, have sprint retrospectives to reflect on what went well and what can be improved in terms of QA.

Which specific practices and tools are useful in each phase of the agile development cycle?

Backlog creation

A good practice is to use a Customer Journey Map (CJM) - it refers to the path that a customer takes before making a “purchase” decision. Visualizing this path can help the team understand what features need to be in the project. Having QA at this stage allows him to draw up his work plan in advance. CJMs also helps to make sure that the requirements on which it is based are correct. (Principle of early fault detection and correction)

Tools: JIRA / Trello / etc for task management + Miro for CJM visualization

Planning

Planning Poker is a method that helps a team to exchange ideas about the scope or time required for the implementation of user stories in a sprint and thus to come to better estimates. Also, it is good to have Story Points to capture the required resources in a single paradigm. Furthermore for every story there are criteria for its completion. (Principle of product and process-dependent quality assurance)

Tools: Task Managers + Planning Poker + communication tools

Development

Pair Programming is a practice, when two developers sit at a work computer and work together on a task. One developer writes the code, while the other questions the correctness of the code and the solution. The second person can also be a QA, as they often have sufficient knowledge to review the code. (Principle of integrated quality assurance during development)

Tools: IDE + Git for version control

Testing

At this stage, QA is most active, checking the results of the team's work. (Principle of independent quality assurance). It is also good practice to automate testing to reduce the time it takes to check minor changes, as well as to use various testing techniques to ensure comprehensive coverage.

Tools: Unit testing tools (JUnit, Mocha) + Integration testing tools (Postman) + Functional testing (Selenium, Appium) + etc

Review and Retrospective

Discussing what went well, what didn't, and how to improve. This refers to the actual progress, as well as the process. (Principle of product and process dependent quality assurance)

Tools: The actual software + FunRetro / Retrium

4 Guidelines in OOP

1. **HiddenField:** This check ensures that class fields are not hidden by local variables or parameters with the same name. It helps maintain a clear distinction between the class's state and the local scope.
2. **VisibilityModifier:** This check ensures that the visibility of class members (fields, methods) is explicitly defined. Although having different scopes is not limited to OOP, it is very important to for instance hide the object complexity.
3. **MissingOverride:** This refers to a missing `@Override` statement on top of an inherited method. Inheritance is one of the core concepts of OOP. Using that annotation makes the code more readable as it gets clearer where the method of the inherited class is used and where it is modified.