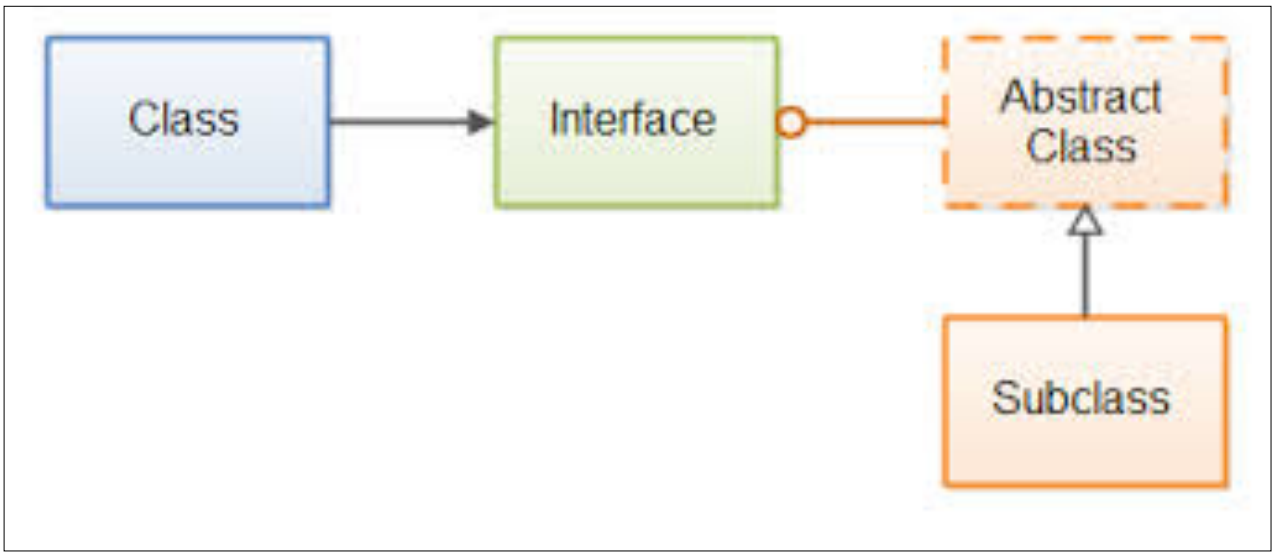

Assignment 01

26 October 2020
Assignment: 30 marks
Type: Report

OVERVIEW

Classes and interfaces are Java's units of abstraction. To create stable, robust and flexible APIs these units of abstraction must be used in a careful way. In this assignment you will research the principles that capture the best practice of using classes and interfaces in Java and write a report describing your analysis.

Deliverable

A report approximately 3000 words to be submitted via Canvas.

Due Date

Saturday 21st November 2020.

GUIDELINES FOR CLASSES AND INTERFACES

Abstraction is a fundamental principle of software design. Abstraction is expressed using the information hiding or encapsulation techniques available in a programming language. A component (any reusable software element, from an individual method, to a class, to a multi-package framework) possessing good abstraction separates its API from all aspects of its implementation. The components of a system communicate only through their APIs and are not dependent on specific implementation details.

The following guidelines support the principle of abstraction in the context of classes and interfaces (Bloch 2018)

- Minimise the accessibility of classes and members
- In public classes, use accessor methods, not public fields
- Minimize mutability
- Favour composition over inheritance
- Design and document for inheritance or else prohibit it
- Prefer interfaces to abstract classes
- Design interfaces for posterity
- Use interfaces only to define types
- Prefer class hierarchies to tagged classes
- Favour static member classes over non-static
- Limit source files to a single top-level class

These principles capture practices considered to be beneficial by experienced developers. Clarity and simplicity are emphasised. In particular, a user of a component should never be surprised by its behaviour; code should be reused rather than copied; the dependencies between components are reduced as much as possible. These guidelines characterise best programming practices for the majority of cases. However, developing software requires experience and consists of learning the rules and then learning when to break them – the goal is always to write programs that are clear, correct, usable, robust, flexible, and maintainable.

OBJECTIVES OF THE ASSIGNMENT

Deciding when to abstract is a contentious topic in software engineering and one where experienced Java developers often disagree. By exploring the principles above you will gain experience of the nature of these disagreements and be better able to defend design choices you make.

OUTCOMES

Write a 3000 word report where you

- Rank the principles above in the order of importance that you consider to best capture good design practice.
- Take your top three principles and describe, in your own words, what aspect of good design the principle captures and how it contributes to creating a stable, flexible Java API.

WORKING TOGETHER

Your report must be your own work, but I strongly encourage you to work with others in preparing the report. Working together will make discovering resources more efficient and will help you see the development process from someone else's perspective, which will result in a more rounded analysis. If you choose to work with others be sure to mention them in the acknowledgements section.

REFERENCES

Bloch 2018, Effective Java, Third Edition, Joshua Bloch Addison-Wesley, 2018