University of Antwerp
Faculty of Applied Engineering

University of Antwerp
Faculty of Applied Engineering

5-Software Design

Lab Session 3

06/11/2025

Jens Duym

Course Outline

Outline labs

- **Part A: UML diagrams**
  Sessions 1 – 2
- **Part B: Design Patterns**
  Session 3 – 5
- **Part C: Projects in groups of 2**
  Session 6 – 9

- **Evaluation:**
  - Entire portfolio: zip containing code, UML diagrams, AI usage
    - Submit before 7th lab at defined date
    - Oral defence
  - Defence of projects

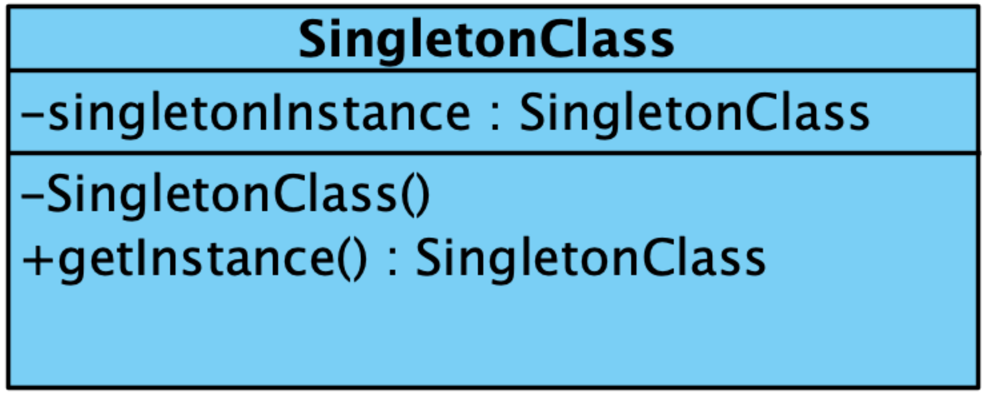
Part B
Design Patterns

Singleton & Observer Pattern

University of Antwerp
Faculty of Applied Engineering

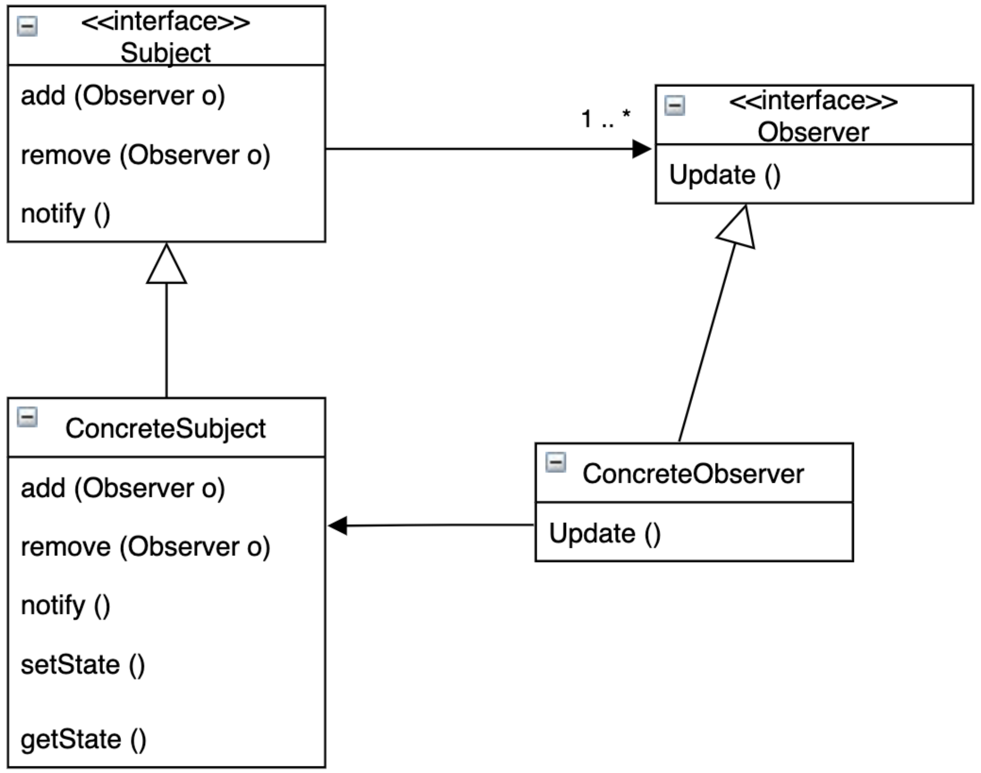
Singleton

- The **singleton pattern** ensured a class has **only one instance**; and provides a **global point of access** to it.


SingletonClass
–singletonInstance : SingletonClass
–SingletonClass()
+getInstance() : SingletonClass

Observer Pattern

- The **observer** pattern defines the **one-to-many-dependency** between objects, so that when **one object changes state**, **all of the dependencies** are notified and **updated automatically**.



Introduction to Git

Introduction to Git

- **About Git**
  - Created by Linus Torvalds, creator of Linux
  - Goals:
    - Making development of Linux more structured with multiple people
    - Working in structured versions for Linux kernel -> Version Control System (VCS)
    - Fully distributed, able to cope with large projects
- **What is Git?**
  - A system that keeps records of your changes
  - Allows collaborative development
  - Allows you to know who made what changes and when
  - Allows you to revert any changes and go back to any previous state

Introduction to Git

- **How does it work?**
  - Can be complicated at first, but few key concepts to understand
  - Git works in **snapshots** of your code
    - You decide when to take a snapshot, and of what files
    - Keeps track of all your snapshots over time
    - Reverting to earlier snapshots
  - Committing
    - The act of creating a snapshot
    - Link a small explanation to a commit, to tell what this snapshot is about
    - Projects are made from a bunch of commits
  - Repository
    - Could see this as your project
    - Collection of all your files, versions and commits
    - Lives both on your computer and in remote location (cloud) -> pushing to and pulling from remote
      - Eg. Github, Bitbucket, Gitlab, self-hosted, …

Introduction to Git

- **How does it work?**
  - Branching
    - To switch development between different features in one project
    - Very handy when working with multiple people
    - Very handy when working in different development stages (bugfixing, testing, integrating, production, …)
- **Basic commands**

`git clone` ***url***	Copy a Git repository so you can work locally
`git add` ***files***	Adds file contents to be snapshotted
`git commit –m “Explanation”`	Makes snapshot of added files and add an **explanation**
`git diff`	Shows diff of what will be added to snapshot and what not
`git pull`	Fetch from remote repository
`git push`	Push your local snapshots to the remote repository

Introduction to Git

- **Try it yourself**
  - Create repository on Github
    - New repositories are usually **public**
    - Private repositories can be made with a free student subscription
  - Download a Git client on your computer
    - Via Command Line (basic commands on previous slide)
    - Graphical user interfaces
      - Gitkraken (paid version for private repositories, but very good) https://www.gitkraken.com
      - Sourcetree (free and very good) https://www.sourcetreeapp.com
      - Github Desktop (rather unclear GUI) https://desktop.github.com
  - Useful links
    https://youtu.be/2sjqTHE0zok?t=48 (course from MIT, highly recommended)
    https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf
    https://docs.google.com/presentation/d/1P3SzBeCLlei-xxNYuMzEZMUM8SFrgMdFwHpPi7OJhBA/htmlpresent

Assignments

Singleton & Observer Pattern

Assignment

- **Three Sub assignments**
  - A Ticketing Service
    - Singleton IdGenerator ensures monotonically increasing ID generation
  - Auction Platform
    - New bids can be placed, two observers react differently
  - Inventory system with SKU’s (Stock Keeping Units)
    - Stock can be changed
    - Two observers keep watch of changes and respond differently
    - Singleton Database contains stock
    - Thread safe
    - Use PropertyChangeSupport and PropertyChangeListener

Assignment

- **At least one integration test** (free to choose)
- **At least one complete Unit test**
- **Bonus: Diagrams, not mandatory, but feel free**