First, a Quick Review!

UML: Class Diagrams

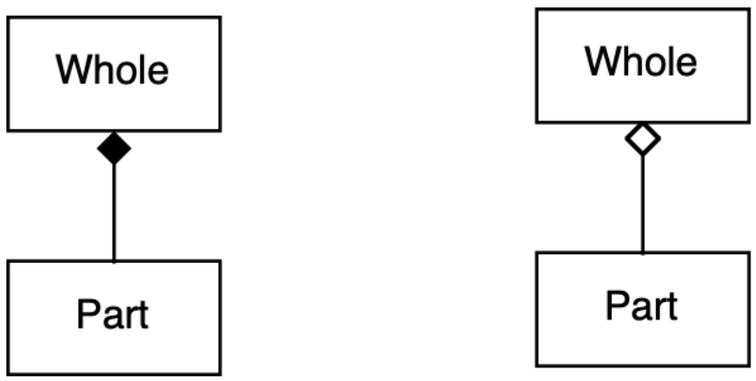
Each class is represented **by a box** divided in three sections:

Class name, Data members, Methods

Player
-name : String -manaPoints : int -healthPoints: int
+Player(name: String, manaPoints: int, healthPoints: int) +getName() :String +fight(other: Player) : void +toString() : String

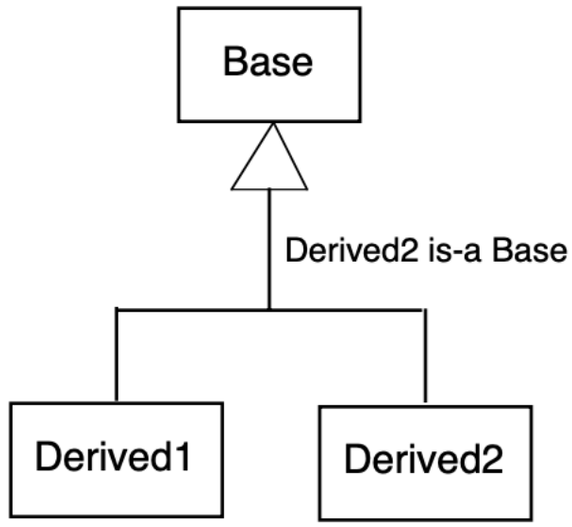
UML: Class Diagram Relationships

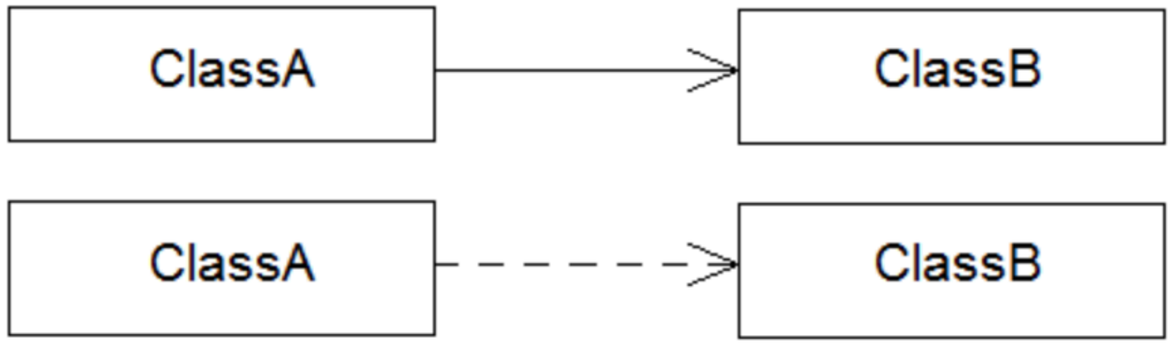
Composition and Aggregation are represented with diamonds.
A composition is a solid diamond, and an aggregation is not.



Inheritance, or **IS-A** relationships,
are represented with triangles

Relationships with i**nterfaces** look similar,
but lines between boxes are dashed.





Associations and **Dependencies** between classes can be represented with arrows.

UML for Design Patterns

A Design Pattern:

- Describes software solutions and component structures that are common
- Defines software classes, roles and relationships
- Does not depend on any particular programming language
- Must be applied, tailored to create particular solutions

The benefits are many:

- The patterns **work**
- They facilitate **documentation**
- They enhance **communication**
- They promote **re-use**
- They limit **errors**

Classified in four main groups:

Behavioural, Creational, Structural, Concurrency

UNIVERSITY OF TORONTO MISSISSAUGA

Patterns you have seen:

- Assignment 2 separates Views from an underlying Model of program state. This reflects a pattern we will discuss in more detail soon: Model-View-Controller.

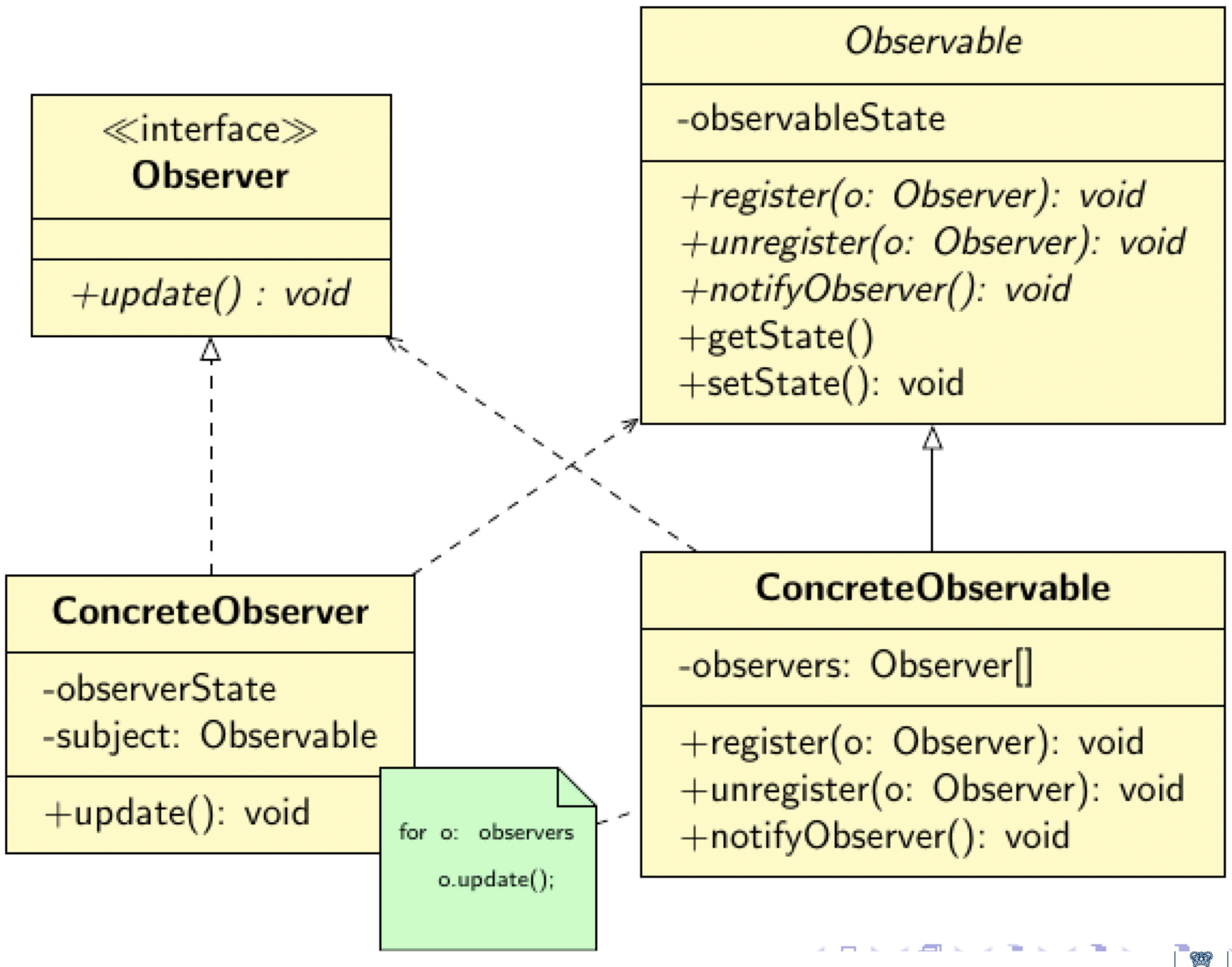
- You have seen Iterators and Iterables in past lectures. Iterator is a Design Pattern, too.

- Your lab this week relates to another popular Design Pattern called “Observer”.

- More design patterns have been posted to Quercus; you will want to pick a few that interest or intrigue you for your group project, and you will need to relate these to features your user stories.

Design Patterns: Observer Example

Observer Pattern UML



Accessibility

Web Content Accessibility Guidelines

- The international standard for creating accessible interfaces
- The most well-accepted, well-documented accessibility guidelines in the world
- WCAG issued by W3C's Web Accessibility Initiative
- WCAG 1.0 issued in 1999
- WCAG 2.0 issued in 2008/WCAG 2.1 in 2018
- WCAG2ICT—guidance for applying WCAG 2 to non-web information technologies (like our group projects.)

Web Content Accessibility Guidelines

Software should be:

- **Perceivable**: Information and user interface components must be presentable to users in ways they can perceive

- **Operable**: User interface components and navigation must be operable

- **Understandable:** Information and the operation of user interface must be understandable

- **Robust:** Content must be robust enough that it can be interpreted by by a wide variety of user agents, including assistive technologies.

Examples of Guidelines: Contrast

- **Colour Contrast**
  contrast is a measure of the difference in perceived "luminance" or brightness between two colours
  Ratio of text colour on background colour should be at minimum 4.5:1

- **Example of colour contrast that has low ratio**
  Pure red (#FF0000) on white background has a ratio of 4:1.
  I am red text

- Pure green (#00FF00) on white background has a very low ratio of 1.4:1.
  I am green text.

- **Example of colour contrast that has good ratio**

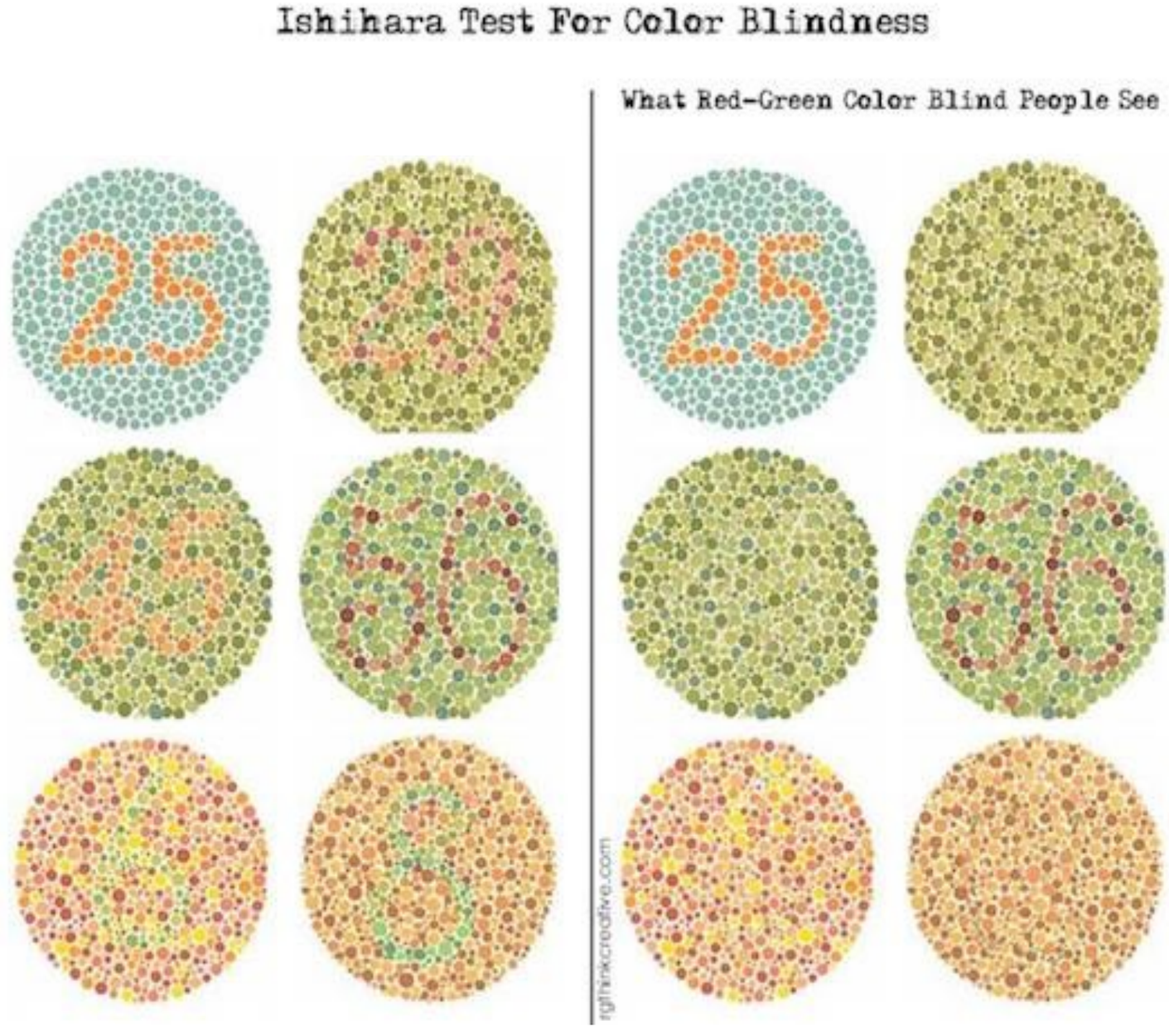
- Pure blue (#000FF) on white background has a contrast ratio of 8.6:1.
  I am blue text

- **Test all coloured elements for accessibility to the colourblind**

- **Allow colours to be changed through a preferences pane**

Examples of Guidelines: Contrast

- **Avoid these colour combos:**
  - Green & Red
  - Green & Brown
  - Blue & Purple
  - Green & Blue
  - Light Green & Yellow
  - Blue & Grey
  - Green & Grey
  - Green & Black



Examples of Guidelines: Contrast

Use patterns and textures to show contrast

Examples of Guidelines: Alt Text, Sounds

- **Provide descriptive text for all visual components**
  - E.g. ‘alt’ text in an image
  - Provide an option to hide the graphics entirely
  - Test with Microsoft Narrator, Apple VoiceOver, NVDA, JAWS (screen reading apps)

- **Allow users to configure the volume and frequencies of sounds**

Accessible User Stories and Acceptance Criteria

There are two methods for including accessibility in your user stories and acceptance criteria:

Add acceptance criteria that considers diverse user needs

Create user stories that focus on personae with complex communication needs.

Ideally, when you write user stories, many differing needs and abilities should be represented and considered.

From *Indiana University's University Information Technology Services*

Example: Accessible User Story

User Story: As a user who is receiving search results, I want to know how many results there are so that I can change my search terms if needed.

Acceptance Criteria:

- Search, filters, and results can be navigated using keyboard only.
- User is aware when results appear if using a screen reader.
- User is aware of the number of results if using a screen reader.
- User is aware when results appear if using screen magnification.
- User is aware of the number of results if using screen magnification.
- Text should have a contrast ratio of at least 4.5:1.

Estimating Effort

- Once you have user stories, you will need to estimate how much effort each one of them will take for your group.

- But estimating effort is not easy! And your estimates will depend on the people in your team.

- The effort your team requires to implement a feature may not be at all comparable to the effort other teams require. And that’s ok!! Your units must be calibrated to your context.

Project Brainstorming Sessions

- Brainstorm project ideas with a TA and an Accessibility Mentor from CNIB.

- Fifteen minute Zoom consults available on: **October 26, 27 and November 1**

- Sign up at this URL

- Present ideas for and/or drafts of user stories and acceptance criteria! Consider user stories that are accessible and engaging to users with low vision.

Prioritization Poker

Planning Poker

- Let’s play a game to help us assess the effort a user story might require.
- First, appoint a “scrum master” or sprint leader.
- **For each of the ATM user stories:**
    - Have the sprint leader read the story aloud.
    - Talk about the story as though you must implement it. How many of you will you need to work on this story? What skills will the work require? Do you have these skills already, or will you need budget time to learn?
    - Once you have discussed the story, let each group member privately select a number to reflect the units of “effort” you think the story will take. Select a number from among these options: **0, 1, 2, 3, 5, 8, 13.** Let each team member write an estimate on a piece of paper that is hidden to others.
    - Now, reveal your numbers! Do you all agree? If not, have the sprint leader ask members with the highest and lowest estimates to justify their choices. What is the source of the discrepancy? Why such a high number, or why such a low onw?
    - Repeat the process until your entire team agrees on a numeric estimate. The sprint leader should then record this estimate alongside the user story in the team’s product backlog.