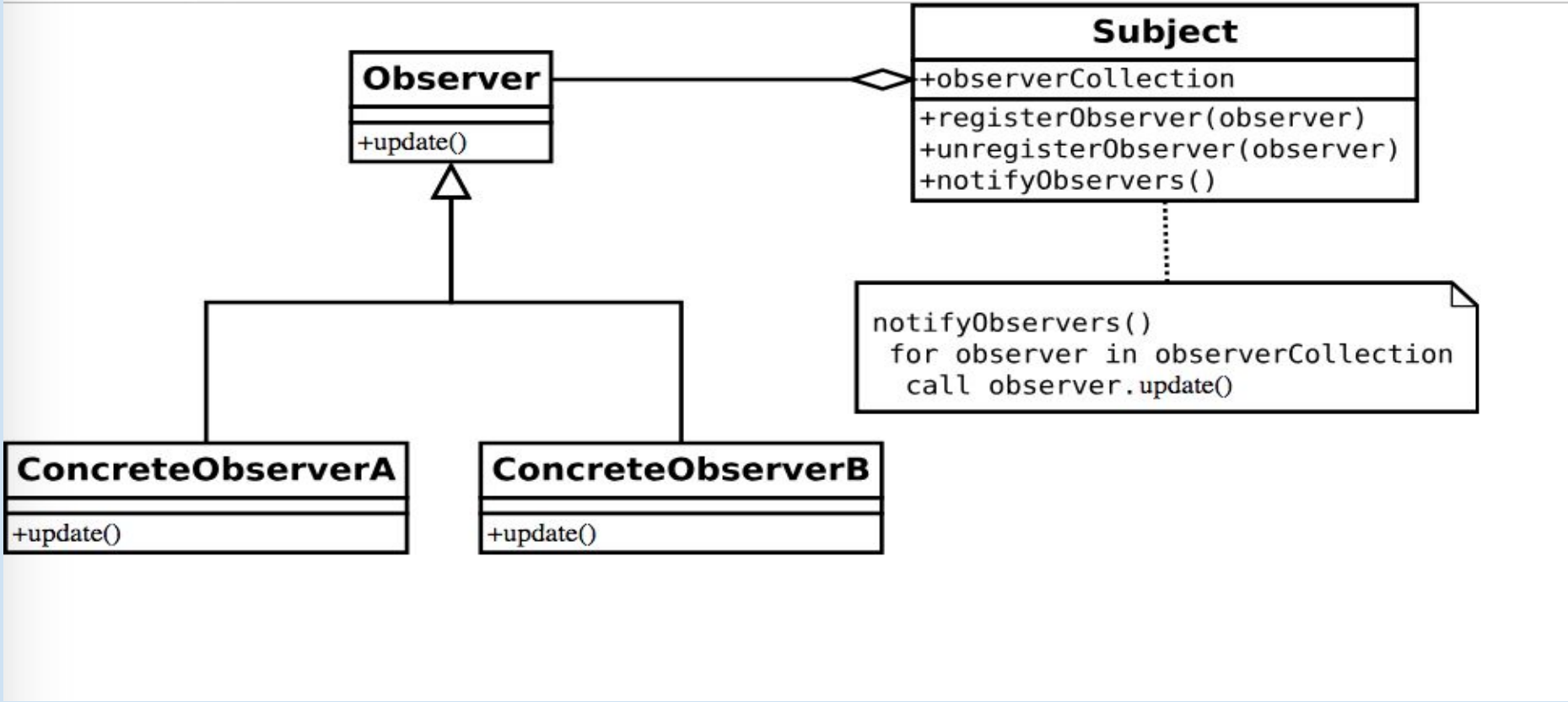
CS342 Software Design

- HW #3 due today
- Extra Credit due today
- Project #3 due Monday
- Teams for project #4, beginning of next week
- Today: Observer and Agile

Observer Pattern

- Magazine subscriptions: 1 publisher with many subscribers
- Email listserv: subscribe or unsubscribe
- Users of an app: all get notified of an update by developer
- SmartPhone and laptop updates: your registered with the maker and receive updates
- In Java, Button and Listener

Observer Pattern:



Observer Pattern: Advantages

Loose Coupling!!!

- Subject only knows that observer implement Observer interface.Nothing more.
- There is no need to modify Subject to add or remove observers.
- We can reuse subject and observer classes independently of each other.

Observer Pattern: Disadvantages

Possible Memory Leaks!

Strong Reference: Widget w = new Widget();

- A Widget object is created on the heap. As long as w is active, that Widget can not be GC.

Observer Pattern: Disadvantages

Possible Memory Leaks!

Weak Reference: WeakReference<Widget> wr = new WeakReference<Widget>(new Widget());

- Can be GC even when being used.
- Can not assume that the object still exists
- String s = wr.get(); if(s != null) { // great! Still exists}

Observer Pattern: Disadvantages

Possible Memory Leaks! “Lapsed Listener Problem”

Observers register and unregister themselves with the subject.

Memory leak happens when the observer fails to unsubscribe to the subject. The subject holds a strong reference to the observer and keeps it alive in the heap.

Fix the problem by having the subject just keep weak references instead.

Agile Software Development:

Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it.

Manifesto for Agile Software Development:

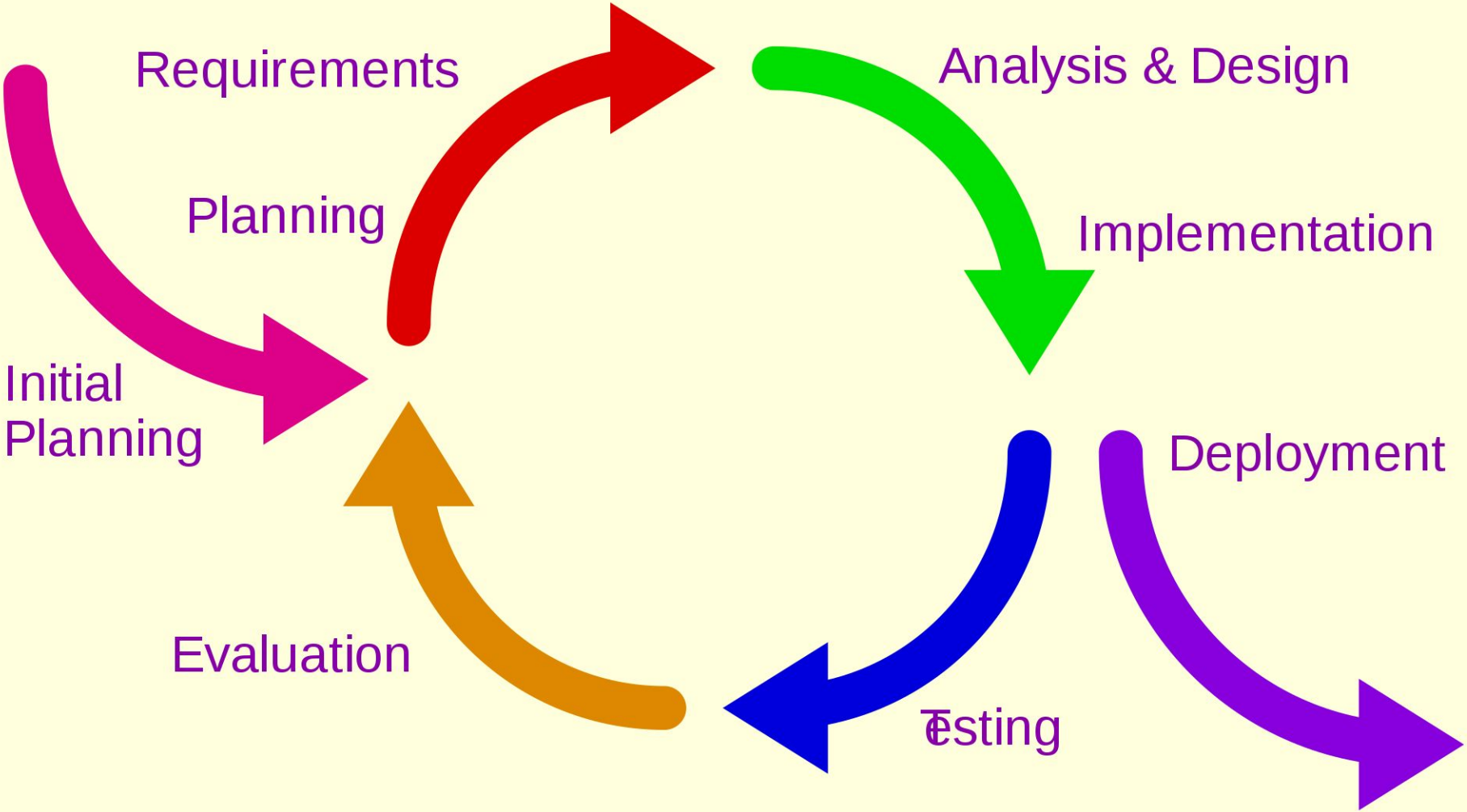
- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Individuals and interactions over processes and tools.

Working software over comprehensive documentation.

Customer collaboration over contract negotiation.

Responding to change over following a plan.


Requirements
Analysis & Design
Planning
Implementation
Initial Planning
Deployment
Evaluation
Testing

12 Principles Behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

12 Principles Behind the Agile Manifesto

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.