**3250 Semester Notes**

**Git**

Branching- Need to start branching. A branch for each person, not working on MASTER

* Can have branches of branches (Have to merge when on lower level)
* Git Branch + name of file – creates a brand new reference, not looking at it, but its created
* Switching – Everything I see and do happens to master, to get out of branch use “git checkout name of file”

**Scrum**

* Maximize teams abilities
* Work on requirements ASAP
* Product owner, Find out which user story if the most important and make sure that is known in the presentation
* Keep product backlogs visible to everyone (working on project)
* Sprint backlog is the list of work the dev team must address during the next sprint
  + Comes from the product backlog and select priority of the product backlogs
  + Once a sprint backlog is committed, no additional work can be added unless the team says so
* Three Pillars
  + Transparency
  + Inspection
  + Adaptation
* Five Values
  + Commitment: Team members commit to achieving team goal each and every sprint
  + Courage: Courage to work through conflict and challenges together to do the right thing
  + Focus: Focus exclusively on the teams goals and sprint backlogs. No work done other than backlog’
  + Openness: Team members and stakeholders agree to be transparent about their work and challenges
  + Respect: Team members respect each other to be technically capable and to work with good intent
* Roles
  + Product Owner
    - Relates to customer
  + Scrum Master
    - Unblocks dev team
  + Dev team
* Retrospective
  + Reflect on previous sprint, go over what can be improved

**The Real Syllabus**

* Important stuff
  + Act confidently and secure
  + Respect rights and properties of others
  + Work cooperatively
  + Courteously respects authority; polite
  + Practices self-control
  + Accepts responsibility for behavior
  + Resolves conflict appropriately
  + Exhibits positive attitude toward learning
  + Participates in class discussions
  + Engaged listener
  + Follows all directions
  + Completes work in timely manner
  + Works independently
  + Responsible for property
  + Asks for help when needed
  + Produces neat, accurate, quality work
  + Positive work ethic
  + Good organization skill
  + Demonstrates technology skills
  + Uses library media to enhance learning

AEIP

A- Abstraction

Grouping your code into sections, like classes, objects & APIs where the complex processing is happening “behind the scenes”, and all that you need to know or all that communicating objects need to know are a small number of input or output variables

E- Encapsulation

Hides internal details of one object from another

I- Inheritance

Instead of rewriting code, have classes inherit from super classes. This means your program will require less code and be easier to add onto or update.

P- Polymorphism

The ability of one object to be generic and used in many different ways. Taking a different form is called polymorphism

Design Patterns (incomplete- refer to Design Patterns PDF for further info)

General / abstract factories: the dependency Inversion Principle

General rules:

* No variable should hold a reference to a concrete class.
* No class should derive from a concrete class.
* No method should override an implemented method of any of it base classes.

Dependency Injection:

* Where one object (or static method) supplies the dependencies of another object.
* A dependency is an object that can be used(a service)
* An injection is the passing of a dependency to a dependent object.
* Passing the service to the client rather than allowing the client to find it.
* Inject information into an object

Inversion of Control (\*crucial to perform)

* A system that records changes to a file or a set of files so that you can recall specific versions later on (like GitHub)

Abstract Factory Pattern:

* Creation through inheritance
* Creates object of single types

**Midterm Review**

* Crucial that we version control
  + Not capable of handling the complexity of the code we are writing
* Communication
* Design Patterns
* High Level software engineering, haven’t gotten a lot better at it.
  + When he showed up how many lines there were per item.
  + Test driven development
  + Waterfall model (good for small software projects)
* Scrum
  + Retrospective
  + Roles
  + Scrum guide (download that shit)
* Design Patterns
  + Interfaces
  + Big 4 of OO, AEIP
* Observer patterns
  + Subject to which we subscribe
    - Each observer implements and interface, when subject changes the observer is activated
  + Event driven
* Decorator Pattern
  + Open close principle
  + Wrap object inside another object of the same type
* Factory Pattern/No abstract pattern
* AEIP
  + Abstraction
    - Grouping your code into sections, like classes, objects & APIs where the complex processing is happening “behind the scenes”, and all that you need to know or all that communicating objects need to know are a small number of input or output variables
  + Encapsulation
    - Hides internal details of one object from another
  + Inheritance
    - Instead of rewriting code, have classes inherit from super classes. This means your program will require less code and be easier to add onto or update.
  + Polymorphism
    - The ability of one object to be generic and used in many different ways. Taking a different form is called polymorphism