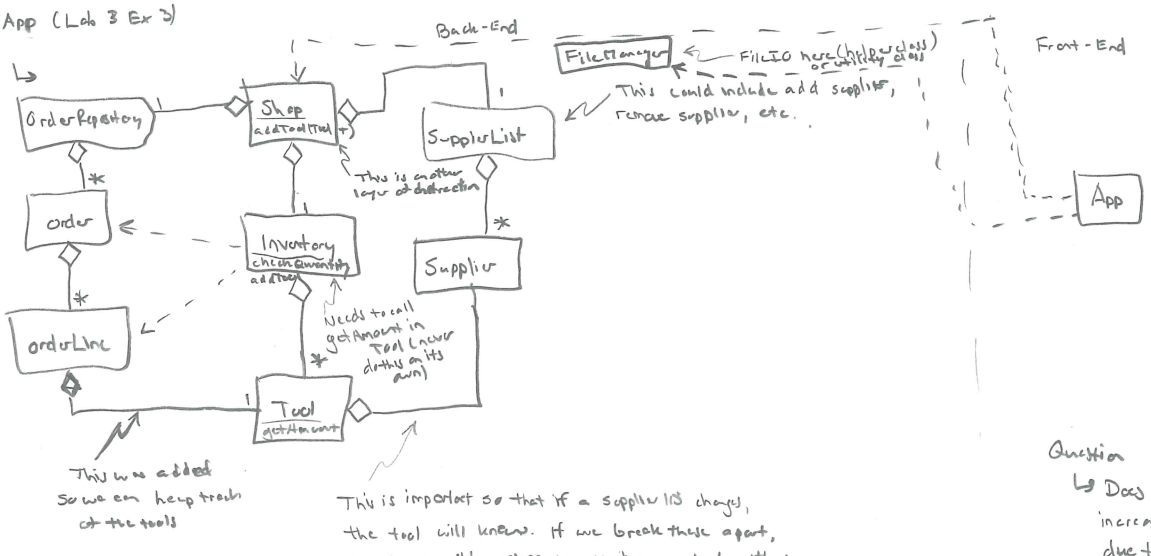
ENSF 593/594
Oct. 8, 2019

Testing

↳ Testability is a great measure of the modularity of a program

↳ Always keep in mind that software design is not open-ended, and design choices need to be justified

Shop App (Lab 3 Ex 3)



Question

↳ Does dependency relationships cause increased coupling? I had avoided them due to coupling, but then transferring data became problematic

↳ It seems that no, since he seemed unconcerned about having mult. dependencies on the FileManager

↳ what impacts coupling?

↳ Having multiple classes that depend on a single class (in terms of strong or relationships) increases coupling.

↳ For the lab, do we use both has and own?

Class Diagrams

↳ A Class Diagram is a static views of the system

↳ A dependency does not hold data, and therefore does not typically get shown

↳ We want to always stay away from composition, because it places too much responsibility on the parent class

↳ The front-end should not be loading data!

↳ A helper class should be included, which takes responsibility away from the front-end

↳ The class Diagram shows only where the data resides, and not how the data is passed around

↳ Therefore dependencies should not be shown, but should be used if needed!

↳ Decrease coupling as much as possible

↳ we can totally have dependencies in the back-end

↳ Avoid dependencies from the front-end

↳ we can chain addTool(Tool...) functions together

* From chat with Dr. M:

↳ we can and should have dependencies in our backend, primarily to pass data around.

↳ if we find we have too many dependencies from one class, we should add another layer of abstraction (more classes beneath)

↳ Dependencies are used to pass data around code so that we don't have to use primatives (like I did)

Lab

↳ Implement with minimal decoupling, but having cyclical coupling is ok, typically as long as each class only has one or two classes that depend on it.

↳ Follow his diagrams exactly, or justify ours (justify deviations)

↳ For instance, we won't necessarily need an orderrepository or supplierlist (but I will have it)

Code (Shop App)

```
Shop {
    Inventory theInventory;
    Supplier sup;
    Order theOrder;
}
App {                                  A branch of (static, even) FileIO
    main() {                             ↳ The helper constructs to objects
        Helper h = new Helper(...);
        *Create array of tools
        Inventory i = new Inventory(tool array)
    }
}
```