Week 1 Notes

Intro to the field of Software Design (from Week 0)

1. What do we mean, when we say "Software Design is about people"?
   People collaborate together to work on the same project, and the software is designed to used by people.
2. What does "efficiency" refer to, when we talk about Object-Oriented Design?
   Each object has their own set of data, and we can reuse the same code through inheritance.

Intro to git (based on Week 1 lecture and first lab)

3. What is a Version Control System and why is it useful?

- A master repository of files exists on a server.
- People “clone” the repo to get their own local copy.
- As significant progress is made, people “push” their changes to the master repo, and “pull” other people’s changes from the master repo.
- The repo keeps track of every change, and people can revert to older versions

4. Briefly, what do each of the following git commands do?

`git clone` make a copy from the remote repository to the local machine

`git pull` sync from the remote repository to the local repository

`git add` tell git on the local repository to track the file

`git commit -m "Replace this with a message"` make a version of the currently tracked files

`git push` push all the commits from the local repository to the remote server

`git checkout` without arguments: check out the HEAD; with commit id: check out some specific commit

`git branch` without arguments: list all of the branches in your repository

`git merge` merge some branch in to the current one

5. Why is it important to make commit messages concise and informative?
   That way we will be able to have a high-level understanding of what each commit does and do not need to check out each commit to find out.
6. Why do we use branches and pull requests?
   To avoid having to constantly resolve conflicts, developers often create a branch, where they work on a new feature, then submit a pull request.

Intro to CRC (based on Week 1 lecture)

7. What is a CRC card? What is a CRC model?

CRC stands for Class, Responsibility and Collaboration. CRC model is collection of CRC cards specifying the Object-Oriented Design of a software system.

8. Briefly describe how a software development team can use CRC to design a program together?
    - Typically, you are given a specification (a written description of the requirements) for the software system.
    - After identifying the CRCs, we will have an initial set of classes and their relationships

Intro to Java (based on Java readings and quizzes)

9. Why do we need a main method?
   It is the entry point of any executable program.
10. What is a "constructor", "instance variable", "class variable", and "method"?
    constructor: to initializes the newly created object before it is used.
    instance variable: variables belong to the instance of a class, thus an object.
    class variable: there's only one copy of that variable that is shared with all instances of that class.
    method: a method is a block of code which only runs when it is called.
11. Which package is the String class in? You can find this information on the Oracle website by searching "String java" on your favourite search engine.
    java.lang

Week 0 Lab

12. List three things that stood out to you in the code reading exercise during the first lab. These can be about the code itself, differences between Java and Python, or even about how you felt the group discussion went.
    Java Compilation
    Primitives types and their wrapper classes
    Java interfaces
13. What parts of git or the GitHub website are you comfortable with now? What parts are you still unsure of? (We recommend you get in touch with us either through Piazza or office hours to address any problems you're facing!)
    Everything is fine except the creation of auto workflow which is a new feature.

Week 1 Lab

14. In the "abstract class or interface?" exercise, one of the examples was about Beverages (Tea, Juice, Coffee, etc.) and another had items that could be drunk (Tea, Coffee, Soup, etc.): what is the difference between these two scenarios, and what might be a reason to use an interface rather than an abstract class (and vice versa)?
    Tea, Juice and Coffee are all Beverages and share many similar attributes such as price and amount. However, Soup needs to be cooked so it is more a like food and requires a preparation time. The way we "drink" food and beverages are different as well as once may require a straw while the others require a spoon. Therefore, it is reasonable for Tea, Juice and Coffee but not soup share a same parent class. The "drink" method is better made into an interface, so we can simply implement the method at the parent level of those drinks that are served in different ways.
    It is reasonable to prefer classes over interfaces when the instances of such share similar attributes. For different types of classes that do not necessarily share similar attributes, we better use interfaces

to implement those methods with a same name.

15. List two things that you think demonstrate good design in the Ticket Vendor CRC model and two things that you think demonstrate bad design or will lead to issues when trying to implement the model in Java.

    PROs:

    - The Ticket classes store seat number/letter and price for different objects, because two tickets cannot have a same seat number unless it is being oversold.
    - All Shows are corresponding to a MasterManager.

    CONs:

    - Shouldn't be storing the name, date, time of show in the Ticket class. Rather, put them into the Show class and only store a reference to the show in the Ticket class, because a same show is unlikely to have different name, date and time of show.
    - The "getter for names of people that bought tickets on a given date" responsibility should be moved over to the TicketVendorSystem because it is possible for different MasterManagers to sell tickets on a same given date.