Week 2

Java (based on course notes, Quercus quizzes, and GitHub Classroom exercises)

1. What does it mean for a method to be overloaded? What does it mean to override a method? And how does this compare to Python?
   Overloading: same method name in a class, different signature (different types/counts of arguments)
   Overriding: in a child class, same method with same signature as the parent class with a different implementation.
2. What is polymorphism? Give an example of polymorphism.
   An object can be interpreted as an instance of different types.
   People -> Student -> ForthYearStudent
   An object created by "New ForthYearStudent()" is not only an instance of ForthYearStudent, but also Student and People.
3. Give two examples of casting: one of upcasting, and one of downcasting.

```
Student s = ForthYearStudent(); // upcasting
People p = s; // upcasting
ForthYearStudent f = (ForthYearStudent)s; // downcasting
```

4. What can interfaces enforce? What can they not enforce?
   An interface can enforce the method prototypes to be implemented on all classes that implements the interface. It cannot enforce any instance variables to be declared.
5. When would we use an interface over a parent class? How do we know which one to use?
   When different classes share similar properties but are not in parent-child relationships. Use classes when the methods do not need to present in another non-child/cousin class. Otherwise, use interfaces.

Git

6. This week, the coding exercise used Git with MarkUs. What were the differences between submitting to MarkUs, and submitting to GitHub?
   - MarkUs might refuse some commit if there is some file not 'required' by the assignment, or files under a structure not created by MarkUs are tracked and committed.
   - GitHub provides more features such as pull request, code review.
   - GitHub has a modern interface while MarkUs seems pretty old-school.

File System (File Structure Quiz / case study / https://en.wikipedia.org/wiki/Path_(computing))

7. What is a path?
   A path is a string of characters used to uniquely identify a location in a directory structure.
8. How can you find the path of a given file on your computer?
   ```
   pwd # then append the file name with a '/' in between
   ```

Case Study (lecture and case study code on GitHub)

9. List at least one example of too little or too much documentation in a class or interface.
   IShellState.java: no comment at all
   There is no such thing as "too much" documentation. More high-quality comments help maintenance of the code. Even if the comments affects readability of the code, we can always hide them in an IDE with a single-click.
10. Locate the main method in the code. Is the main method too short? Too long? What is a benefit of having fewer lines of code in main?
    Java-Shell/src/main/java/driver/JShell.java: the length is about appropriate.
    Having few lines in main means most of the low-level logic is abstracted into different classes, and only the high-level logic is present in main() which helps maintainability.

11. Identify an interface in the code and explain what you think its purpose is.
    IShellState: it defines what actions the Java shell state should be able to accomplish.
12. How would one add a new command to the program? Try to identify everywhere in the code where something would need to either change or be added.
    Add a class that inherits class Command. Then in Java-Shell/src/main/java/constants/Constants.java , add the command name into the Hashtable (dictionary).
13. When writing our code, we typically want to keep things as general as possible. Directory.java has an example of a too-specific return type (ArrayList). What could the return type be replaced with?
    public ArrayList getChildren() is too-specific.
    We should ask for a children index in the arguments and return a FileSystemObject .

Test-Driven Development (lecture)

14. What are some benefits of writing tests first?
    - We can tell other how a method is supposed to work by the tests to prevent others breaking your code
    - We can get a direction by implementing the methods so that we can write the code faster
15. What is unit testing? Why shouldn't we just test the whole program all at once?
    Setup, call the method, teardown. To test functions separately from each other in case some of them fails.

16. What are some features of IntelliJ that help developers write / make use of junit tests during development?
    We can select which method we want to test.

Lab 3

17. What are some strategies or actions that might make working in a team easier for this course? (e.g. communication strategies, methods of code review, organization)
    communication strategies: provide constructive feedback on teammate's code
    methods of code review: look for coding style and the things that do not confirm to our CRC designs
    organization: everyone should be responsive for her/his part
18. List two problems you found in your group member's code during the Trader exercise. What benefits are there to code reviews? Do you think code reviews are worth the time investment?
    i. Naming convention: class name that doesn't start with a capital letter
    ii. Instance variable (property) management: some variable was declared as 'final' which is not changeable once some value is assigned.

    Benefits: Everybody helps to catch errors and contribute with their own knowledge. The project will go into a shape everybody agrees on, so the time is worth investing.