Simple Factory: .50 points. I covered the fact that Simple Factories are used to create/instantiate objects, but I failed to include that they then return this object and that the conditional is determined by method parameters.

Factory Method: .3 points. I covered the fact that this class is a method that is used to call constructors and create objects, but I incorrectly stated that this pattern doesn't use inheritance.

Abstract Factory: 1 point. I covered the fact that Abstract Factories create families of objects by holding invariable information in the superclass and variable information in the subclass.

Prototype: 1 point. I covered the fact that prototypes use an object as a reference when creating new objects and that it reduces the likelihood of error versus making the object from scratch.

Builder: 1 point. I covered the fact that Builders are used to form complex objects by combining smaller objects. I also used the analogy of building a house to supplement my answer.

Singleton: 1 point. Mentioned that Singletons have private constructors which force there to only be one instantiation of a certain object, and that they provide global access to this object.

Object pool: 0 points. Missed.

Adapter: 1 point. Mentioned that the Adapter is used to allow 2 objects to communicate. Also made an analogy to a USB to USB C adapter in real life.

Bridge: 1 point. Mentioned that bridges are used to increase the separation between the user and the code, and talked about how the code is easier to change without affecting the user when using a bridge.

Composite: 1 point. Mentioned that the Composite pattern allows object to be composed in tree structures, with the Composite pattern consisting of 2 or more of these objects while getting all of the functionality of both objects. Also referenced the seat/chair analogy used in the Harry Potter book.

Decorator: 1 point. Mentioned that the Decorator is used to add layers of functionality to an object. Made an analogy to a Russian nesting doll.

Facade: .70 points. I covered the fact that the facade can be used to hide extra, unwanted functionality from a user, but I failed to mention that it separates a client from a subsystem of components. I also threw in an analogy to the window of a store.

Flyweight: 1 point. I mentioned that flyweights allow many objects to be instantiated by allowing them to share common traits, thus only having to store the object-specific ones. I used the example of bullets in an FPS game to demonstrate this.

State: 1 point. I covered the fact that the State pattern allows objects to change functionality without changing itself, and the fact that States are allowed to change. I included the example of buttons in a payment portal for an online shop.

Template Method: 1 point. Covered that the Template Method stores invariable parts of an algorithm in the Superclass, and individualizes certain steps for each usage case in its subclass. I gave an example of processing different types of files or using an algorithm on a collection to demonstrate this.

Observer: .70 points. While I said that Observers listen to see if an action happened, I didn't say that it uses a subscription-based broadcasting to notify listeners. I used the analogy of an EventHandler to demonstrate this.

Iterator: 1 point. Mentioned that iterators allow you to loop through objects without exposing any information about their implementation.

Chain of Responsibility: 1 point. I mentioned that the Chain of Responsibility pattern allows objects to be passed up a chain of objects until they are correctly handled. I included the example of an ActionListener/EventHandler, and included the analogy of call escalation in customer support.

Mediator: .5 points. I mentioned that the mediator is used to control the order of actions performed in a complex software system, but I failed to mention that it reduces dependency and that it restricts direct connection between 2 or more classes.

Visitor: 1 point. Mentioned how Visitors create hooks that allow for the use of external code to be implemented in a program while breaking encapsulation.

Proxy: 1 point. Mentioned that Proxies are interfaces for interacting with "real" objects. I used the real life example of an ATM being a proxy for your bank account.

Memento: .85 points. Mentioned that it saves a state of your code and that it is commonly used in the implementation undo/redo functionality, but I failed to mention that it does this without revealing the details of its implementation.

Interpreter: 1 point. I mentioned that Interpreters take input and convert it to output that your code can understand. I also included the example of how it is commonly used in compilers to convert the code from a certain language to Machine code.

Command: 1 point. I mentioned that the Command pattern is used to store one or more commands in the form of a request in an object.

Strategy: 1 point. I mentioned how Strategy patterns contain Subclasses for different usage cases that contain instructions for handling each case, and that they run all the way through unlike the State pattern. I also explained how the State pattern can transition through different states, but the strategy does not, instead going all the way to the end of its specific set of instructions then stopping.

Null Object: 1 point. Mentioned how the Null Object is a placeholder for nothing, and is usually used to implement the “do nothing” function in a program. I gave an example of this in the UnsupportedOperationException found in Homework 2 (LispList) to explain this.

Model-View Controller: NA point. I mentioned how the MVC is a compound pattern that splits the code up into 3 pieces: the model, the view(s), and the controller(s).

Total Score: 22.55/26 (B, 86.73%)