IS 217 Exam 1

Submit the exam by uploading it to Moodle. For 10 Bonus points you can submit the code portions of the exam through Github by placing a link to a repository in you exam submission.

**Questions: 1 - 5 – 5 Points Each**

**1-2 paragraphs per answer**

1. What is a design pattern?  
   A design pattern is like a blueprint for a specific code structure that has been used to solve   
   common problems found in software programming and design. Each design pattern is   
   unique in the way it handles the data flow through the code. They standardize ways to  
   handle dataflow in a consistent manner, enabling faster production times and easy to use  
   solutions to problems.
2. How does the design pattern improve your ability to communicate with other programmers?  
   Because design patterns are consistent in the way they are implemented each time, it  
   is easy to convey what strategies are used in the flow of data. This enables programmers  
   to better communicate how the code is manipulating the data. Instead of having to try to  
   describe exactly how a code structure is handling data, one can just say the name of the  
   design pattern that was used. Granted the other programmer knows the design pattern,  
   he will quickly be able to understand the flow of the code.
3. How does a design pattern help you solve programmatic challenges?  
   Throughout the history of programming, solutions to problems have been consistently solved  
   in similar manners. The solutions to these programmatic challenges have been recorded,  
   named, and implemented many times throughout the years. Design patterns bloomed because   
   programmers often run into the same type of programmatic challenges. Thanks to this history,  
   programmers can now rely on many design patterns to help them through many, many  
   programmatic challenges the encounter.
4. Is a design pattern a law or guideline? Why do you think so?  
   A design pattern is a guideline. There are practically an infinite amount of solutions to any   
   one problem in the world of programming. These design patterns are meant to be just  
   that, a pattern. The intricate details of the pattern don’t have to be exactly the same,  
   because each program is unique in its data handling, and each pattern can have many  
   different variations.
5. Do you think that you have improved as a programmer because of learning about them? How?  
   Absolutely. They have shown me how greater minds than mine work. Instead of reinventing  
   the wheel, I can now look at blueprints of the wheel and perhaps even improve upon them.  
   They are inspiring when implemented correctly, and help speed up development time by   
   tenfold.

**Questions: 6 - 13 – 5 Points Each**

**1-2 paragraphs per answer**

**Explain the following design patterns in terms of their purpose and what type of pattern it is (Creational, Behavioral, or Structural)**

1. Factory Pattern  
   A factory pattern is a creational pattern. Its purpose is to act as a factory does; it creates   
   objects of similar design in mass amounts. The factory is set up in a way that promotes  
   the creation of data elements specifically tailored to the programs requirements.
2. Observer Pattern  
   The observer pattern is a behavioral pattern. Its purpose is to watch other objects   
   and report changes when they happen inside of the other objects. It does this by   
   maintaining a list of subjects and providing callback features whenever a specific  
   occurrence happens in any of the subjects within its list.
3. Decorator Pattern  
   The decorator pattern is a structural pattern. It is used to add functionality to an  
   object on-the-fly. Basically it dynamically adds new behavior to an object whenever   
   needed at runtime.
4. Singleton Pattern  
   The singleton pattern is a creational pattern. The singleton pattern is used when   
   the instantiation of only one object is necessary. It insures that one and only one   
   object of this structure can be created in the entire program. It needs global scope  
   access to insure this happens.
5. Constructor Pattern  
   The constructor pattern is a creational design pattern. Constructors are used in   
   normal OOP as a function that is run immediately following the instantiation of   
   the object. JavaScript must explicitly create this functionality for its objects.   
   Following the creation of an object, a constructor() function is called and all of   
   the code within that function will then be run before anything else in the object.
6. Prototype Pattern  
   The prototype pattern is a creational pattern. Prototypes are similar to parent   
   classes that are to be inherited in normal OOP. A prototype is fully instantiated   
   before its use and can then be used as a blueprint model for other objects that  
   wish to inherit the functionality of the prototype.
7. Pub Sub Pattern  
   The pub sub pattern is a behavioral pattern. Pub stands for publisher, while   
   sub stands for subscriber. In this pattern a subscriber, or observer, maintains   
   a list of objects, or publishers, to watch. As the publishers publish content,   
   the subscribers interpret the content in a number of different ways (usually   
   just formatting and displaying the content).
8. Fly Weight Pattern  
   A fly weight pattern is a structural pattern. It is used to efficiently share   
   information needed by other objects in the program. For example a   
   fly-weight object could be used to handle data from a database and then   
   feed that information to the objects that require it.

**Questions: 14-17 – 5 Points Each**

**In JavaScript write an example of each pattern:**

1. Singleton Pattern
2. Factory pattern
3. Prototype Pattern
4. Constructor Pattern
5. Decorator Pattern

**Questions: 19-20 – 5 Points Each**

1. What is the difference between a language that uses classical inheritance vs. a language that uses prototypical inheritance? Can you name an example of a language for each respectively?  
   A language that uses classical inheritance, like Java, has all of its objects laid   
   out as a solid class structure which can then be inherited from by child classes   
   directly. A language of prototypical inheritance, like JavaScript, has to explicitly   
   declare functions as prototypes which act as child methods.
2. Name two design patterns that work together well and provide an example of how you would use them in words or code.  
   A decorator pattern and factory pattern could work well together. The factory   
   pattern would be responsible for pumping out objects with similar functionality,   
   and the decorator pattern could then add functionality to either the individually   
   created objects themselves, or it could even add functionality onto the factory   
   itself.