Advanced .NET Assignment 5

Andy Le, Sam Gershkovich, Cam Randall

Problem 1: Toys

1) The Factory pattern was used for this problem
2) The Factory pattern is best suited for creating objects that extend a base class. We know a few defined derivatives of the Toy class but there could be more. The factory will hide and manage all constructor logic so it's easy to implement new derivatives.
3) The toy class and subclasses have multiple methods and we weren't sure how to implement this into the construction method of ToyFactory.
4) https://www.dofactory.com/net/factory-method-design-pattern

Problem 2: Build a Computer

1) We used the Builder creation pattern to solve this problem
2) This pattern was chosen because a Computer object has one defined implementation consisting of different parts. The build pattern was used to implement a ComputerBuilder object that could add each part and return the created Computer object.
3) The most challenging part of this problem was trying to design constraints for error handling / input checking on the various part objects/classes. Some were fairly obvious and common while others more generic. For example, a computer case obviously can not have a negative size but could be any reasonable size so we limited it to a meter in each dimension.
4) https://www.dofactory.com/net/builder-design-pattern

Problem 3: Mailroom Organization

1) We used the Reactor behavior pattern to solve this problem
2) The reactor pattern was chosen because it best matches the problem description and functionality for handling concurrent requests. A request would be a new piece of mail and the Reactor would have two handlers: one for delivering and one for placing in review. More handles could be created as needed to match request volume
3) This problem was challenging because there isn't a lot of resources on implementing a Reactor pattern in C#. It seems like it is more meant for web-based implementations but we thought it was the best option for this problem.
4) Used course slides, videos, github to learn about Reactor implementation

Problem 4: Auction

1) The observer pattern was used to solve this problem.
2) The course slides mention this pattern as being suitable for an Auction system and we agreed that this behavioral pattern matched the system required for this problem.
3) Implementing this pattern was simple but the challenge in solving this problem was making sure we had the proper methods and properties for an Observer object that matched the functionality defined for this problem.
4) Used course slides, videos, github to learn about Observer implementation.