Lessons Learned in CIS35B (Advance Java Programming)

Lab 1

- I learned that we should create a separate package for each set of classes that together implement some functionality and are logically related to each other. In this lab, we created 3 packages namely util, model and driver.
- I learned that we should create methods out of logic that is needed in more that 1 place. Whenever applicable, such methods can be put into a separate package which only contains commonly used methods. In this lab. we did this for FileIO class.
- I learned that as much as possible utility methods should catch all exceptions within themselves and simply return true/false success status to the caller so as to make caller code simpler.
- I learned how we can write a text file that has some structure to it. With smart use of delimiters, we created a text file that contains all the information about car configuration.
- I learned how we can read a text file with a predefined structure and extract all the pieces of information from it to populate the program's in-memory data structures.
- I learned what serialization is and how it helps convert an object into a stream of bytes or vice a versa.
- I learned how we can serialize and deserialize an object and how a serialized object can be written to a file or read from a file.
- I learned how by simply implementing an interface, we can enable some functionality. In this lab, we made Automobile implement Serializable interface so that we could serialize the objects of Automobile.
- I learned how some functionality can be divided into more than 1 classes to keep code modular. In this lab, we used 3 classes namely Option, OptionSet and Automobile to handle the operations related to an automobile.
- I learned how toString() method can be overridden and used to make it easier to print details about an object.

Lab 2

- I learned how to create custom exceptions and use them to handle unexpected situations during program execution.
- I learned how finally block can be used to execute a piece of code regardless of whether exception occurred or not.
- I learned how a single exception class can be used to indicate many different types of errors in a program.
- I learned how a program can self-heal itself and gracefully continue execution even when it encounters unexpected situations during its execution.
- I learned how we should create interfaces to enforce a particular functionality on its child classes. In this lab, we created one interface that only contained methods related to creating Automobile objects, and another interface that only contained methods related to updating Automobile objects.

- I learned how we can implement all functionality for a class (e.g. BuildAuto) in a separate class (e.g. ProxyAutomobile) and how this can be useful in hiding all implementation details from the users of our classes.

Lab 3

- I learned about the 2 hierarchies in Java Collections Frameworks, one under Collection interface and another under Map interface.
- I learned about the Set collection and how it can be used to store unique elements.
- I learned about the List collection and its 2 implementations namely ArrayList and LinkedList. I learned how ArrayList can efficiently provide random access, while LinkedList can efficient provide inserts and deletes.
- I learned about the Map collection and how it can be use to store key value pairs. I also learned how to use a LinkedHashMap.
- I learned what iterators are and how to use them to iterate through the entries in a collection (e.g. LinkedHashMap).
- I learned how using ArrayList instead of arrays can sometimes make a program simpler. ArrayLists, unlike arrays, are not of fixed size and can expand themselves as needed.
- I learned how objects can be shared to save memory. In this lab, we created an ArrayList in Automobile to track user choices. We populated this ArrayList with the same OptionSet objects that we used to track Automobile configuration options.

Lab 4

- I learned how threads can be created in Java and how they can be used to perform multiple tasks concurrently to achieve better performance. In this lab, we perform each update/edit operation in a separate thread. This allow us to serve multiple user requests concurrently rather than sequentially.
- I learned why synchronization is necessary in a multithreaded program. I learned how data can get corrupted without proper synchronization on shared data.
- I learned how synchronization can be achieved in Java on methods or on objects using synchronized keyword.
- I learned how wait() and notify() methods can be used to implement complex multi-threaded scenarios where a thread needs to wait for other threads to perform some operation before it can proceed.
- I learned how we can make a thread gracefully exit before finishing its task by making the thread periodically check a shared variable that indicates whether the thread should exit.
- I learned how a class (e.g. EditOptions) can become a target of a thread by implementing Runnable interface.

Lab 5

- I learned about the 7 layers in the networking stack namely physical, data link, network, transport, session, presentation, application.
- I learned about the TCP, UDP protocols and how data transfer under these protocols happens in small packets.
- I learned what IP addresses are and how they are used to identify a host.

- I learned what domain name service is and how it helps translate a domain name to corresponding IP address.
- I learned what ports are and how they are used to identify a process on a host.
- I learned what sockets are and how they can be used to send data from one program to another over network.
- I learned how a program (i.e. a server) can listen on a port for requests from other programs and then respond to those requests using the established connection.
- I learned how objects can be serialized and sent to other programs over network. I learned how these objects can then be deserialized to construct a copy of the serialized object.
- I learned how a complex interaction between server and client programs can be achieved by establishing a protocol for communication and then making both server and client adhere to that protocol.
- I learned how all the logic related to working with a special set of classes (e.g. Socket, ServerSocket) can be contained within a class (e.g. DefaultSocketClient) to simplify rest of the code that uses these classes.
- I learned how in a client-server application, client need not have any data but can still provide a rich functionality by retrieving data as and when needed from a server.
- I learned how in a client-server application, server can just act as a data store and need not have any logic related to user interaction.
- I learned how we can represent some data in more than 1 format (e.g. Properties file and Automobile file) and still be able to extract all the information from it as long as the program completely knows the layout of the data in each format.

Lab 6

- I learned about the HTTP protocol and its most commonly used GET and POST operations.
- I learned how to create forms in HTML and how they can be used to send data from a user to a web server.
- I learned how to use Apache Tomcat to set up a web server.
- I learned how to use Apache Tomcat with Eclipse to create a server that responds to HTTP requests. I also learned how and when to create Dynamic Web Projects in Eclipse.
- I learned about servlets and how to write servlets that programmatically construct desired HTML.
- I learned how the init() method of servlets can be used to perform one-time setup. In this lab, I created all Automobile objects in init() method of the top level servlet.
- I learned how to map servlets to specific URLs (in web.xml) so that a servlet receives only a specific kind of request.
- I learned how a servlet can extract information filled in a form by user from the URL to serve the request.
- I learned how to add static resources like images or text files to a server's execution environment.