18641 Lessons Learned

Part 1: Things or concepts learned for application in Mini 2.

1. Object oriented design theory:
   association - passing an object by reference.
   containment - instantiating an object inside another class.
   inheritance - recycling the properties and method from parent to child
   polymorphism - single method (different meaning in different classes.
2. Inner class and protected methods:
   Class definition can be encapsulated inside another class. An instance of Outer class has to be created for creating an instance of Inner.
3. File IO: (1) read from and output to text file using BufferedReader and FileInputStream and FileOutputStream (2) load from properties file into Properties object which is like a map.
4. Serialization to and deserialization from ".ser" file.
5. Java collections:
   generics: List, Set, Map
   Example: ArrayList, HashSet,LinkedHashMap
6. Exception handling: (1) original exception can be handled by either throwing out or catching locally. (2) customized exception can be created by class extending Exception. We can new a customized exception and throw it, then use try-catch-finally block to catch and do something we want.
7. Abstract class is flexible for future extension and provide part of implementation details but let these hide from subclass's users, where a class could only extend one abstract class.
8. Interfaces which define function prototypes or constants, where a class could implement several interfaces.
9. Scalability by multi-threading and synchronized methods: synchronized methods have the same function as "synchronized(this)".
10. Synchronization by wait(), notify(), notifyAll(),join().
11. Socket programming: Socket = IP address + port number
    Packet is the communication unit. Stream is what we can used to pass information. Server has an infinite loop, keeping accepting client and using new thread to handle.
12. Java servlet: servlet can be run on Apache Tomcat after compiling. It have method prototypes for handling http requests like GET,HEAD,etc. Message and parameters can be passed from url, response can be constructed and sent back.
13. JSP programming: JSP is java servlet together with html and css features. The certain syntax is used to wrap the commands.
14. Database concepts: rules of normalization:
    1. No repeating fields
    2. Create a primary key and use this for referring data in other tables (foreign key)
    3. No interdependency between columns
15. JDBC programming: using java.sql package, where connection and statement are

prepared.

Part 2: List of design lessons learned from project 1.

1. **What is the relationship between containment and encapsulation (as applied in this project), when building components?**
   Encapsulation of objects is through containment and delegation. In this project, Automobile class contains list of OptionSet objects, where Option is inner class of OptionSet and OptionSet also contains list of Option objects. Thus containment helps encapsulation because inner class's functions are hidden from the outside – only CRUD operations are provided by containers.

2. **What are some ways to analyze data (presented in requirements) to design Objects?**
   From the data provided in project description we can try to find patterns in structuring information. For example, the relationship of containment and association. The attributes each possible object should have. Also, the invariant and variant part in different objects with the same type.

3. **What strategies can be used to design core classes, for future requirements, so that they are reusable, extensible and easily modifiable?**
   The responsibilities of different core classes should be assigned properly, so that changes made to one class should not affect others. Also, interfaces and abstract classes can be used for increasing flexibility since different subclass/implementations could be made under the same interface and abstract class. This is delegation strategy.

4. **What are good conventions for making a Java class readable?**
   The convention is for both coding and commenting.
   The code should begin with proper import, then class/interface declaration. The class body has an order of different types of attributes. Declaration first, then initialization. The naming conventions are defined separately for class, interaface, method, variable, constant and package. A reference is https://d1b10bmlvqabco.cloudfront.net/attach/ij9909ftbvo460/h6xvkl7og2l43g/ijd8dmsfkvlu/Java_Code_Conventions.pdf

5. **What are the advantages and disadvantages of reading data from sources such as text files or databases in a single pass and not use intermediary buffering?**

Pros: time and space efficient, especially when the input data is large enough.
Cons: (1) when there is error in input, no temporary buffer means no exception handling could be made for recovering what has been read. (2) When array is used but the initial size cannot be decided because of unknown number of records afterwards.

6. **What is the advantage of using Serialization? What issues can occur, when**

using Serialization with Inner classes?

Pro of Serialization: (1) can be easily serialized for storage and then de-serialized from file back to object without change. Thus the previous computation result in form of object could be maintained for future use. (2) can be sent through socket using stream.

Issue: (1) when inner class has something that cannot be serialized, the serialization of outer class could be affected. (2) After serialization, encapsulation may be meaningless because info is no longer hidden when checking serialized file.

7. **Where can following object relationships be used: encapsulation, association, containment, inheritance and polymorphism?**

encapsulation: when some inner classes don't want the outside to know their implementations but still provide part of functionalities through outer class.
association: when a class need to use another class's non-static methods.
containment: when a class need to have a list of another class's instantiations to store information.
inheritance: abstract class and subclass relationship. Subclass inherits parent class's methods and variables, but can override or create new methods and variables.
polymorphism: interface and different implementations. Strategy pattern for dealing with same problem with different strategies.

8. **How can you design objects, which are self-contained and independent?**

The objects should 1. contain fields and functions to manipulate all the data by user 2. those fields (i.e. data) should not rely on other objects or the system. The first one meets and self-contained requirement and the second one meets independent requirement.

9. **What role(s) does an interface play in building an API?**

The interface serves as a protocol or contract which is an agreed on behavior in API development. In other words, the interfaces define some constants and functions (i.e. API) to users, so users don't need to care about the interfaces are actually implemented as long as the interfaces keep unchanged.

10. **What is the best way to create a framework, for exposing a complex product, in a simple way and at the same time making your implementation extensible.**
    Create framework with interface, then abstract class with real implementations but subclass implementing all those.

11. **What is the advantage of exposing methods using different interfaces**?
    The objects should 1. contain fields and functions to manipulate all the data by user 2. those fields (i.e. data) should not rely on other objects or the system. The 1st one meets and self-contained requirement and the 2nd one meets independent requirement.

12. **Is there any advantage of creating an abstract class, which contains interface method implementations only?**
    Hide implementations but give flexibility of adding new features in sub-class.

13. **How can you create a software architecture, which addresses the needs of exception handling and recovery?**
    Create customized exception class and handling class. Do the throwing when in need and catch using customized handler.

14. **What is the advantage of exposing fix methods for exception management?**

Advantage is that exception can use fix in catch blocks.

15. **Why did we have to make the Automobile object static in ProxyAutomotive class?**

All instantiations of ProxyAutomotive class would share the same Automobile object. So that modifications could be applied to the same one.

16. **What is the advantage of adding choice variable in OptionSet class? What measures had to be implemented to expose the choice property in Auto class?**
    Pros: let outside user to make only one choice from only options provided in the set.
    Measures: getter and setter for the choice in both OptionSet class and Auto class.

17. **When implementing LinkedHashMap for Auto object in proxyAuto class, what was your consideration for managing CRUD operations on this structure? Did you end up doing the implementation of CRUD operation in proxyAuto or did you consider adding another class in Model for encapsulating Auto for the collection and then introducing an instance of this new class in proxyAuto.**

Consideration: the Singleton pattern for the only map, the concurrency issue on race condition, the null pointer issue when object cannot be found in map.
My choice: I end up doing the implementation of CRUD operation in proxyAuto.

18. **What is the best way to setup multithreading in an Enterprise Class application?**

Use synchronized methods or block on every editing method, and use thread-safe data structures.
Also, 1) have concrete class implementing Runnables and run them in the new threads; 2) Allocate thread pool to provide threads

19. **What strategy is used for synchronizing, so you end up with a scalable application?**

The primary tool for managing coordination between threads in Java programs is the synchronized keyword. Because of the rules involving cache flushing and

invalidation, a synchronized block in the Java language is generally more expensive than the critical section facilities offered by many platforms.

Hot Lock may involve multiple thread switches and system calls. When multiple threads contend for the same monitor, the JVM has to maintain a queue of threads waiting for that monitor (and this queue must be synchronized across processors), which means more time spent in the JVM or OS code and less time spent in your program code.

To avoid the hot lock problem, following suggestions may be helpful: Make synchronized blocks as short as possible. Reducing lock granularity, Avoid lock on static methods, Using lock free data structure in Java SE 5.0.

20. What implementation strategy can be used for creating a race condition for testing Multithreading?

Simulate a multithreading environment by opening multiple threads and calculate time.

21. How does Synchronization work in JVM? What are the performance consequences of Synchronizing?

It is implemented using implicit lock of objects. The performance of the program will decrease because of using synchronizing.