LESSONS LEARNED – Java for Smart Phone Development Mini 1

UNIT 1

1) What is the relationship between containment and encapsulation, when building components?

Containment in Java means that a class has fields which exist as another class. In project 1 we created a class called optionSet which had a set of option object. Containment can be thought of as a relationship between different classes i.e a class having other classes as its constituents and this one class contains all other classes.
Encapsulation is an approach in Java to design a model with modularity and abstraction. It's a mechanism of wrapping data and variables all together in a single unit. The variables of the class can be hidden from the other classes or made private which can then be accessed only through getter-setter methods of the current class. The complexity of the class is hidden which ensures that data cannot be changed directly. In the project, in Automobile class, we have optionSet field as private and in optionSet we set option inner class as private. Encapsulation also helps in maintainability.

2) What are some ways to analyze data (presented in requirements) to design Objects?

There are many ways to analyze data needed to design objects. The first step is to transform all the requirements to more specific and operable objects. In the project, the object Automobile is broken down into optionSet and further optionSet is broken down into options. Each option is defined by its name and price. Hence one class is now divided into two more classes for more specificity.
Another method is to design an object by its function. In the project we designed a class File I/O for input/output data serialization. The util package is specifically included to have all the file i/o operations in one place.
Designing an object also needs a lot of thought to be put into visibility and access privileges for dependent classes and objects. We need to consider the data field and method for each object. The easiest way is to have separate objects for different functions. To decide the visibility, we need to consider the method or the function of the designed data. In the project, in Automobile class, optionSet and option are protected. Hence the methods to manipulate these fields are also protected in Automobile class.

3) What strategies can be used to design core classes, for future requirements, so that they are reusable, extensible and easily modifiable?

Good code in terms of clarity and modularity is always a primary concern for future use. In the project, for modifying or printing options, it would be smarter to create an independent class acting

as an option in Automobile class rather than directly accessing it from Automobile. The main class has a lot of options and contained data and modifying an option through it can be messy and complicated.
All possible input methods have to be covered to make the code extensible. In the project, to access optionSet in Automobile optionSet array, we can use an index or a String name. By handling all these cases in the automobile class, data fields can be manipulated at convenience to access the input.

4) What are good conventions for making a Java class readable?

There are a plenty of ways to make Java classes more readable and clear. The packages need to be named correctly so that we can import the needed source file to access the class. It is good practice to be able to view the package organization while compiling. All the instance variables should be declared at the beginning of the code which makes it easier to add more variables when needed and also helps to understand what data is being operated on in the required class. Each variable declaration should be put in a separate line with corresponding comments to indicate why and how the variable is used in the methods. Another thing to do is have specific names depending upon what they are used for and also to have unchangeable variables that can be declared in upper case. Method names should be in camel case and in a 'to-do' style. In the project, methods like 'updateOptionSetName' clearly specify what they are doing.
We should also make sure that the line does not exceed 80 characters as it becomes too long to read. Indent the methods, declarations and loops as and when they are written for clarity and readability. Comment the code extensively as it helps in debugging and user understanding.

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?

One of the most obvious advantages of reading data from sources such as text files or databases in a single pass is that it reduces I/O overhead and saves memory by not saving temporary data. Additionally, it's easy to read data in a single pass.
The disadvantage of reading files in a single pass is that its more difficult to parse data as we do not know the size of the data structure needed apriori. The size of the resource is rarely known and it becomes difficult to allocate memory without knowing how much is needed to be stored. It may cause buffer overflows leading to unforeseen consequences if the data structure is smaller than the incoming data. Reading from a single pass can also be a waste of time and space if the wrapper class is immutable. In my opinion, using a buffer solves a lot of these problems and reading from a single pass should be done only when there is no other option.

6) What is the advantage of using Serialization? What issues can occur, when using Serialization with Inner classes?

Serialization is very useful because an object can be saved on the disk and can be restored as it becomes persistent. A serialized object can be transferred via the network because a Java serialized

object remains in the form of bytes which is portable. The transient keyword can be used to indicate that a field should not be serialized.
When serializing inner classes, one has to be careful because it is a nested non-static class. Serializing an inner class declared in a non-static context that contains implicit non-transient references to enclosing class instances results in serialization of its associated outer class instance. Furthermore, because inner classes cannot declare static members other than compile-time constant fields, they cannot use the serialPersistentField mechanism to designate serializable fields. It has to be ensured that the target class and the classes contained in it need to implement serializable interface.

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

Association defines the multiplicity between objects. Inheritance happens when there is a “has-a” relationship where one class could be an abstraction of the other. Containment is similar to dependency where a class depends on the another class but does not have an instance of the class in itself. In the project, Automobile can be abstraction of Sedan where Sedan can inherit some features of the Automobile. Polymorphism is used when there are many specific classes of a super class. In the project, Automobile can be the super class and BMW, Audi can be its sub classes. The automobile class can be instantiated by any of the sub classes.

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

Self-contained is when an object has data fields and methods to operate on this data. The method call is independent of the other objects. This can be done by first dividing the data to the lowest level such that there are no further levels below it. This puts the data into very specific categories. Then it can be grouped by meaning or function in a class with no overlap. Then methods can be added to each class to manipulate their own data.

UNIT 2

1) What role(s) does an interface play in building an API?

API or Application Program Interface is a set of functions that a user can use without knowing how it works internally. Java interface is like a window that allows the user to use it without getting into the specifics. Interface allows multi-inheritance for a class. In the project, BuildAuto class implements UpdateAuto, CreateAuto and FixAuto interfaces thereby ensuring BuildAuto inherits multiple sets of methods. These interfaces must implement the methods inside themselves and then it is used by the user from the Driver class. Driver doesn’t require to know how these methods are implemented inside the interface as it accesses them through BuildAuto which is convenient and easy.

2) 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?

By using an interface, the scope of the method for an object can be limited and when the object is instantiated within the interface, this instance will only have limited access to all methods which further simplifies the work of a simple instance.
The way to make the code extensible is to create another interface with the new required method and limit the instance within the new interface for this method without knowing about the other methods that implement the same class.
In the project we made an empty BuildAuto class which implements all the methods. Every time a new method was added – updateAuto, createAuto, we just let BuildAuto implement these interfaces in detail. But if we wanted to use a specific method, we can instantiate updateAuto up = new BuildAuto(), and the instance up will have access to updateAuto method but its method scope will be limited in updateAuto.

3) What is the advantage of exposing methods using different interfaces?

For object oriented design, interfaces help in organizing functions of different classes. It is simple to use and it provides for code extensibility. When an interface is used, we also can limit the scope of the objects - in the project if we instantiate createAuto ca = new BuildAuto(), ca will only have methods limited in createAuto i.e it cannot update or edit the car. This helps a lot in distinguishing the class function and access range.

4) Is there any advantage of creating an abstract class, which contains interface method implementations only?

There are many advantages in creating an abstract class – in the project proxyAutomobile is the abstract class. For starters, it distinguishes internal and external usage. The abstract class is internal and has accesses to the internal classes. BuildAuto is external and cannot access internal classes, so the 'extends' relationship between buildAuto and proxyAuto is like a bridge for communication. The abstract class cannot be instantiated. Hence there are no internal instances built. Moreover, the project does not require internal instances. The third advantage would be that when one external class 'extends' an internal class, it could extend its method for external use. It is also easy for a user to use both external and internal methods together if required.

5) How can you create a software architecture, which addresses the needs of exception handling and recovery?

In this project, handling an exception and recovering from it is done by creating a unique software architecture. All the possible exceptions that may occur during runtime have to be considered in a separate package called exception. Then an enumeration with the exception and an independent class with corresponding fix method is built.

6) What is the advantage of exposing fix methods for exception management?

By exposing the fix methods, we can customize exception management by defining how to handle it. In the project, if we input a wrong file, the default exception management prints the stack trace

with null pointer exception but by exposing the fix method we can define how to deal with the situation, like input a default file or ask for the input file again.

7) Why did we have to make the Automobile object static in ProxyAutomotive class?

The automobile object in proxyAutomobile is made static so that it can be shared between objects. We will not be able to update the Automobile object in ProxyAutomobile if it's not declared as a 'static' object. When an instance of buildAuto(child of ProxyAutomobile) is created, a new Automobile object will be created in the class. We do not want that to happen as the same object has to be shared between the two classes. This can be done only when automobile is made static.

8) 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?

Choice variable extends the project further by offering more customization. There are different types of clients that need to be dealt with. Some of them are big customers who just want to upload all the information and some of them want to configure their own and see the price once its configured.

9) 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 encapuslating Auto for the collection and then introducing an instance of this new class in proxyAuto.

I created a new class called Fleet for the implementation of the CRUD updating. This was introduced as a new variable in proxyAutomobile to perform operations with it.

UNIT 3

1) What is the best way to setup multithreading in an Enterprise Class application?

The best way to set up multithreading is to limit the scope of the of the multithreaded method to the object it is operated on, because it takes up too much time and memory. In the project, the method to be multithreaded is an independent class and synchronizes the access to object operation and enables multithreading with no harm to the efficiency of the code.

Race condition for multithreaded programs:
For a multithreaded program, all threads for one process are sharing the context, so there is a chance of racing between the threads. It may so happen that two different threads may want to

change the same Automobiles optionSet name to different names. This will bring in race condition and corrupt the data for this object.

2) What strategy is used for synchronizing, so you end up with a scalable application?

The scope of the synchronizing method should be limited as much as possible. For the application to be scalable, an independent class was built to hold all the multithreaded methods and synchronizing the thread with locking the Automobile object in Fleet class.

3) What implementation strategy can be used for creating a race condition for testing Multithreading?

If two threads access the same object in the same run, and if the method or objects on which the two threads are operating on are not locked, a race condition will arise. But, if for example, one Automobile is locked to one thread and then if two threads want to change its properties, it can be done with no race conditions.

4) How does Synchronization work in JVM? What are the performance consequences of Synchronizing?

Synchronization in Java is the capability to control the access of multiple threads to any shared resource. This introduces thread contention which occurs when two threads try to access the same resource simultaneously causing the Java runtime to execute one or more threads slowly, or even suspend their execution. It definitely decreases performance but it solves the race condition problem.

UNIT 4

1) How to read from a '.Properties' file?

Reading from a .Properties file is much easier than reading from a text file as everything is saved in a "Name=Value" format. We will save time as the file does not have to be parsed. To read the file, load method is used to load all the "Name = Value" pairs into an instance object and the use another method to read in the name and value.

2) What is a socket?

Socket is an abstraction of a communication link between machines over some network. In this project, client (user) and server are communicating through a socket.

Identifying a socket that's needed to communicate:
A hostname and port address are required to establish communication between the user and server. The hostname can be parsed as a string, or is typically an IP address. Port is the id of the socket which is tied to a specific usage of the socket.

InputStream and OutputStream are generally used for socket communication.

Blocking feature in ServerSocket:
When serverSocket is used, the call to the client to 'accept' causes the program to wait until the method returns. This can cause the program to hang. If we need to perform other operations, accept has to be placed in its own class.

One server handling many clients:
The server has to run infinitely long or until the user manually stops it. To implement a non-stop server class a detailed handle session method in an independent thread class is required. In the project, server is run in an infinite loop – while(true) and once it accepts a request from the client, it puts the socket into a new DefaultSocketServer thread to handle the session. Each client can then have its own thread and make the response and get the response.

Client thread working non-stop:
The client thread can also be run infinitely by using a while(true) loop. In the project, for the client, there are two while loops – the inner and the outer one. The outer loop waits for the user input and the inner while handles a specific session. If the handle method runs into exceptions that cannot be handled, the inner loop breaks whereas the outer loop will continue operating. If the client has to be stopped, the user can enter a '0' which will break the outer loop and go to the closeSession method to exit the client.

UNIT 5

1) What is the difference between JSP and Java Servlet?

JavaServer Pages is a webpage scripting language that can generate dynamic content while Servlets are Java programs that are already compiled which also creates dynamic web content. Servlets run faster compared to JSP. JSP can be compiled into Java Servlets. But it is easier to code in JSP than in Java Servlets.

2) What is Tomcat?
Apache Tomcat or Tomcat is an open-source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), and WebSocket, and provides a pure Java HTTP web server environment for Java code to run in.

3) Transferring information from the client using a HTML form

When an HTML form is used, like 'select with option', each option can be given a name and value. A form action is defined to the destination page and the method to get. Then when the form is submitted with input, the whole form is transferred to the destination page.

4) Transferring an object from the server to JSP

To perform this action, the interface HTTPsession can be used. Use session.setAttribute("objectname", object) and the page directed to can start a session to receive the object with "object name", like session.getAttribute("objectName").

5) Jumping between web pages

The 'action' in the JSP can be set as action = destination page url to jump from one page to another.

6) What is the difference between the doGet() and doPost methods?

HTTP is a request-reply protocol: the client (e.g. a web browser) sends a request, and the server (e.g. Tomcat server) responds with a reply. Each request consists of a "request-line", a series of "header" lines and optionally a "body". A typical request line has three parts:

- HTTP request method (e.g. Get)
- The URL
- The protocol version (e.g. "http/1.1")

The HTTP specification defines 8 standard HTTP request methods – GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE & CONNECT). The doGet and doPost methods in the servlet API are methods for processing HTTP GET and POST respectively. In doGet the parameters are appended to the URL and sent along with the header information. This does not happen in case of doPost. In doPost the parameters are sent separately. Since most of the web servers support only a limited amount of information to be attached to the headers, the size of the header should not exceed 1024 bytes. doPost does not have this constraint.

UNIT 6

1) What is Normalization Theory?

Normalization Theory involves relations, attributes and dependencies of attributes upon one another. With normalization, redundant and inconsistent data can be minimized.

2) What is primary key and foreign key?

A primary key is a field in the table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values. A table can have only one primary key which may consist of single or multiple fields.
A foreign key is a column (or columns) that references a column (most often the primary key) of another table. The purpose of the foreign key is to ensure referential integrity of data. In other words, only values that are supposed to appear in the database are permitted.

3) What are the three rules of normalization?

First Normal Form – First normal form (1NF) says that the arrays or other repeating fields should not be used.
Second Normal Form – For a table to qualify for second normal form, it should be first normal form (1NF) and all of the data in the table must be dependent on the value of the primary key.
Third Normal Form – For a table to qualify for third normal form, it must be in 1NF and 2NF and each of the columns in that table except those used as keys must not be interdependent.

4) What are the steps to design relational models?

There are essentially two steps in designing relational models.
First, gather the requirements – data requirements, functional, business and performance requirements.
The second step would be to develop a logical model. A logical data model is a representation of the data elements used by the client and all the relationships between the data elements. Entity relationship modelling is one of the common methods for developing a logical model.

5) What is JDBC and how can it be typically used?

JDBC is a Java API for database connectivity. JDBC enables us to write Java applications that
- Connect to a database
- Send queries and update statements to the database
- Retrieve and process the results received from the database in answer to the query

To use JDBC, import java.sql.*;
- It is required to have a database driver
- Open a connection
- Pass some queries to the database
- Process the results as needed
- Close the connection
- Deal with any errors