Lessons learned

Name: Subramanian
Andrew Id: snatara1

#Item1: What is the difference between Data Encapsulation and data hiding
Data hiding refers to parts of the program that can change but must not be exposed directly to outside world. The manner in which the data can change is left to the implementation that is hidden. Data Encapsulation promotes data hiding by providing access specifiers, interfaces to achieve it in object oriented systems.

#Item2: What are the ways to analyze data (presented in requirements) to design Objects?
Analysis of data can take various forms such as classes, methods or variables. Classes represent a real world object in software that has state and demonstrates behavior. The variables/data members encapsulate the state using access specifiers such as private/protected/public while the methods implement the behaviors that operate on these variables/state.

#Item3: What is use of encapsulation, association, aggregation and composition
Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. For this reason, encapsulation is also referred to as data hiding
Association is a relationship where all object have their own lifecycle and there is no owner.
Aggregation is a specialized form of Association where all object have their own lifecycle but there is ownership and child object can not belongs to another parent object.
Composition is a specialized form of Aggregation and we can call this as a “death” relationship. It is a strong type of Aggregation. Child object dose not have their lifecycle and if parent object deletes all child object will also be deleted.

#Item4: Composition vs inheritance: which to prefer
The general guideline in object oriented systems is to prefer composition to inheritance as it is more malleable / easy to modify later, but do not use a compose-always approach. With composition, it's easy to change behavior on the fly with Dependency Injection / Setters while inheritance is more rigid. Favoring composition over inheritance helps to create flexible and maintainable code.

#Item5: When is inheritance preferred?
Inheritance is preferred when you have strict "is a" relationship between classes. When a set of classes demonstrate and derive the same behavior from a parent class, an inheritance hierarchy can be introduced.

#Item4: When to use inner classes
Inner classes allow you to define one class inside another class – which is why they are called "inner" classes. Inner classes are typically used, when a class is an internal implementation detail of another class rather than a part of its external interface. Usually inner classes are data only classes.

#Item6: Separate interfaces from implementation
The design paradigm allows changing the implementation independently of the interface. This helps deal with changing requirements.

#Item7: Enforce non-instantiability using a private constructor
When a class has a private constructor that means that the class cannot be instantiated as an object by the external world. This functionality is useful for cases such as classes holding static content, constants etc that do not change in the course of the life cycle of the application. Having a private constuctors also ensures that the class cannot be extended for inheritance purposes.

#Item8: Prefer interfaces to abstract classes
Interfaces are better than abstract classes for the below reasons:
1. Existing classes can be easily retrofitted to implement a new interface.
2. Interfaces allow the construction of nonhierarchical type frameworks
3. Interfaces are ideal for defining mixin classes. A mixin class is a type that a class can implement in addition to its "primary type" to declare that it provides some optional behavior.
4. Interfaces can enable safe and powerful functionalities when used as wrapper classes.

#Item9: Java Conventions for code readability
Following are some good recommendations for code readability in java:

1. Declare all instance variables in the beginning of the class.
2. Naming of variables and methods should closely resemble to their real world counterparts
3. Declare variables in the beginning of the functions or code blocks.
4. Do not extend code over 80 characters in a single line.
5. Comment the code generously

6. Bundle the functionality into a java package so that it can be reused anywhere.

#Item10: What are the advantages of reading data from files and databases without intermediary buffering.

Reading data without buffering is efficient, as it will take less time and use no resources. However this can also expose issues such as overflow, if the data set is large and requires mandatory staging in order to read and make sense out of it.

#Item11: Advantages of serialization

Serialization helps to save the data to disk and make it persistent. Serialization is also a key feature in network communication where data over the wire is the serialized form of the object in bits.

#Item12: Role an interface plays in building an API

Interface gives a name to the API and hides the implementation because any class that implements this interface can implement in its own way. It also allows to change the implementation without changing the signature of the method and so the callers are not affected if there is any change in the implementation, so they are not affected by it.

#Item13: Making automobile object static in ProxyAutomobile class

If it is not declared static, then it cannot be updated in the Proxy Automobile class, because when an instance for BuildAuto (child of ProxyAutomobile)is created, a new Automobile object will be created in the class if it is not static. So this requires Automobiles in ProxyAutomobile to be static so that it can be shared between objects.

#Item14: Advantages of exposing fix methods for exception management

By exposing fix methods for exception management instead of using default exception handling method, one can define how to handle the exception. With the default exception handling, a wrong Automobile object or filename may result in program termination if not handled whereas exposing fix method, one can define how the situation can be handled either by having a default automobile name or a backup default file.

#Item15: Using checked exceptions for recoverable conditions

The purpose of checked exception is to use exceptions for conditions from which the caller can reasonably be expected to recover. By throwing a checked exception, you force the caller to handle the exception in a catch clause or to propagate it outward.

Each checked exception that a method is declared to throw is therefore a potent indication to the API user that the associated condition is a possible outcome of invoking the method.

#Item16: What is the advantage of LinkedHashMap over HashMap?
LinkedHashMap and HashMap both support key-value pair insertion and search. They all support constant time insert and search. But the HashMap class does not have order, while LinkedHashMap keeps an order of time for when you insert this object. This is very effective for insert time based iteration in this data structure.

#Item17: Best way to setup multithreading in an Enterprise class application
The best way to setup multithreading is to scope it to the object that is going to be operated upon. This is because, all synchronization constructs used for multithreading come at its own cost and can add up as a performance cost to the end application. So making multi-threading as narrow and fine grained is the recommended approach so as to not harm the efficiency of the solution developed to solve the problem at hand.

#Item18: Race conditions and how to avoid them.
Race conditions are inevitable when multiple entities try to access and modify a shared resource regardless of the order in which they intend to change the state. This can lead to surprising results as interleaving of operations can lead to the shared resource providing an inconsistent view to the world, and lead to bigger problems of affecting the functionalities that depend on the state. The widely adopted ways of locks, condition variables and other synchronization constructs mitigate race conditions and preserve write ordering such that the information is cohesive and consistent at all times for consumption.

#Item19: How does synchronization work in JVM? What are the performance consequences of synchronization?
Synchronization in Java can be performed at two levels, one at the object level and the other at the method level of the object. Synchronization can have a negative performance impact on the application as it allows only one execution context at a time, while the others need to wait for this to finish. However this is a tradeoff that one has to pay to avoid race conditions.

#Item20: Do not synchronize every method of the object.
Synchronization must be adopted wisely, as it can slow down the program. It needs to be used for protecting critical sections involving shared resources with a high probability of race conditions.

#Item21: Best Strategy for synchronization:
Best strategy for synchronization is to ensure that it was sprinkled very generously in the code, but is centered on a narrow piece of functionality or code, which protects shared resources being modified in a unordered fashion. Using synchronization constructs just because they are available will slow down the program and can have major performance impacts.

#Item22: How to create a race condition to test multithreading
Simulating race conditions without synchronization constructs in place is quite easy. If two java threads modify a global variable in a unordered manner, race conditions occur quite freely and this can be demonstrated by printing the value of the global variable. Every execution of the program will display a inconsistent and unexpected value of the global variable in the course of execution.

#Item23: What is producer-consumer problem? How to handle it?
Producer-consumer problem is a classical problem in multithreaded programming. Producer produces resources while consumer consumers the resources. The need is to make sure that the consumer consumes only after the producer produces the items and the producer produces the items only after the consumer consumes the items. Synchronization is needed between the producer and consumer to ensure that they follow the behavior mentioned.

#Item24: What is meaning of deadlock in multithreading programming?
Deadlock is a scenario where execution of each thread blocks each other; thereby no thread can make progress in execution. So in essence both the threads are stuck and this can leave the application is hung state.

#Item25: What is the difference between wait and sleep in java?
Both Wait and sleep pause the thread execution but the difference lies in what context they are called. Wait is called from a synchronized context and will release the lock before pausing while sleep is called in a normal context and will keep the lock before pausing. Calling sleep with a held lock can be detrimental to the performance of the application.

#Item26: What is an immutable object in a java? How is it useful?
Immutable objects are those, which do not change/cannot, change once they are created. This comes useful in multi-threaded scenarios as immutable objects do not

need protection and hence no synchronization constructs are necessary, so no negative impact to the application.

#Item27: What is a properties Java class? How is it useful?
Properties class is a subclass of Hashtable from the Collections framework. It is used to maintain lists of values in which key is a String and the value are also a string. Properties objects are very useful in representing entities, which can be expressed as a key value pair. They are quite useful in manipulating java properties files.

#Item28: How to read a java properties file?
A Java properties file can be read by parsing the file and the results can be stored easily in a java properties object. This is because the file contents are in key value pair format, which fits nicely into the workings of a properties object.

#Item29: What is a client server model of computing?
The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

#Item30: What is the most efficient way for a server to handle multiple clients?
The most efficient way for a server to handle multiple clients is to spawn an independent thread to handle request/responses with each client. This will ensure concurrency and also no client is kept waiting while the server is satisfying the request from a different client.

#Item31: What is a socket?
A socket is an endpoint of communication. It could be implement TCP or UDP protocol functionality. It has not network presence and is present only in the client and the server, serving as endpoints of communication and hence has not role in the routing layer.

#Item32: What is the difference between port and a socket?
A port is a virtualization identifier defining a service endpoint while a socket is a communication endpoint.

#Item33: How is a socket identified in the system?
A socket at the server is identified by IP address and a port number.
A socket at the client is identified by IP address.

#Item34: What are blocking and non-blocking sockets?
For blocking sockets, socket APIs or the function that operate on the sockets like accept/receive can block indefinitely waiting for input/data, while for non blocking sockets these APIs return immediately when no input or data and do not wait forever.

#Item35: How to transfer data through a socket?
Data is transferred through socket using streaming protocols. In java we have the inputstream and outputstream classes that handle the data communication through the java socket.

#Item36: What is a java servlet? Where is it widely used?
A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET

#Item37: What is JSP? What is rationale for using JSP?
JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP essentially represents a HTML page which can be implemented in multiple languages but usually with Java.

#Item38: What is a webserver? What are the popular webservers available?
A web server is an information technology that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. Some popular webservers are Apache Tomcat, Microsoft IIS, IBM Lotus, NGinx etc.

#Item39: What is a web container?
Web container (also known as a Servlet container) is the component of a web server that interacts with Java servlets.

#Item40: what is the difference between a webserver, web container and an application server?
In Java: Web Container is used to manage the components like servlets; JSP. It is a part of the web server.
Web Server is a server, which is capable of handling HTTP requests sent by a client and respond back with a HTTP response.

Application Server or App Server: can handle all application operations between users and an organization's back end business applications.

#Item41: How to display a webpage using Java?
In the Java servlet class, the doGet and the doPost methods handle the input while the response.getWriter handles the output to write to the web page.

#Item42: what is the interface between a java servlet and JSP?
The implmentiaons of the methods of interface HTTPSession is used to transfer data between servlet and JSP.

#Item43: How to connect MySQL via java?
Java connects to mysql using the jdbc package. It provides robust and sophisticated means to connect to the database.

#Item44: What is the difference between SQL and a MySQL databases?
Some key differences noted as below:

1. SQL databases are table based databases whereas NoSQL databases are document based, key-value pairs, graph databases or wide-column stores.
2. SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data
3. SQL databases are vertically scalable whereas the NoSQL databases are horizontally scalable.
4. SQL databases emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance)

#Item45: What is normalization and why do we need it?
Normalization is the process of organizing the columns (attributes) and tables (relations) of a relational database to minimize data redundancy.
The objective is to isolate data so that additions, deletions, and modifications of an attribute can be made in just one table and then propagated through the rest of the database using the defined foreign keys. This will remove the inconsistencies seen in redundant information residing in multiple rows of the tables without normalization.

#Item46: Why do we need to be careful about normalization?
Under-normalization will cause excessive repetition of data; over-normalization will cause excessive joins across too many tables. Both of them will get
worse performance

#Item47: What is primary key and foreign key?
A primary key is a field in the table, which identifies a unique row of the table.
A primary key cannot have NULL values and must be unique.
A table can have only one primary key, which may be a single filed or composition of multiple fields.
A foreign key is field that references another field in another table. The purpose of foreign key is referential integrity of data.

#Item48: What are the three rules of normalization?
First Normal Form: As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

Second Normal Form: As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second normal form.

Third Normal Form: Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this transitive functional dependency should be removed from the table and also the table must be in Second Normal form.

#Item49: What are the rules to design a relational database?
Designing a database using a relational model involves a two step process:
1.The first step is gathering requirements and sees its implication on scale and performance.
2. Develop a logical model to represent the data and relationships among the data so that data can be easily queried for create/delete/modify purposes. Most popular modeling used is the entity relationship modeling.

#Item50: What are the limitations of a relational database?

Some of the limitations of relational databases:

1. There are some forms of data and information that the relational database cannot accommodate easily and adequately. Examples are unstructured data, spatial data, multimedia data etc.
2. Complex queries require sophisticated processing power. A database with huge data sources or data with complex relationships may require more powerful servers to return results within an acceptable response time.
3. Database is bound to rigid schema and not flexible to changes during runtime.
4. Complex relational database systems can lead to these databases becoming "islands of information" where the information cannot be shared easily from one large system to another.
5. Logical and physical modeling can never be separated.