Lessons Learnt

1. Package creation and usage. It helped in bringing the clarity within the project. The interdependence between the packages could be determined neatly.
2. Final keyword usage with respect to variable. This helps in changing the value only at 1 location. If tomorrow, the design value of a variable changes, the value has to be changed only at one location and not at multiple locations.
3. Final keyword usage with respect to method.
4. Static field usage for class instance. They can be directly referred by class name and are common to all objects. This was specifically used for common repositories or ServerSocket in socket connections. They are created just once and used again and again by many entities.
5. Public, private and protected access specifier usage wrt Automobile, optionSet and option classes. Protected and private help in isolation and clear ways have to be defined to access these variables and methods. This helps a lot in multithreading since you have entire workflow available.
6. Class diagrams helped us to form a better approach to solve a problem. Initial analysis helped in segregating the problem into smaller individual tasks and made solution simpler and cleaner.
7. Creating custom exceptions. They offer a very neat way of throwing user defined errors which might not be defined by java. They are especially very useful in sanitizing the input data on which decisions are made later.
8. Multithreading improves system performance, memory utilization and cpu usage.
9. Multithreading by extending Thread. This is a simpler and concise way of making threads. Mostly used where application response time is faster.
10. Multithreading by implementing Runnable. Was especially useful since in Java we can extend only 1 class. Multithreading by implementing Runnable was helpful in servlets as they extend HttpServlet class already. So I couldn't extend a class for Thread if it was already extending HttpServlet.
11. Synchronization of resources by using Keyword synchronized. This helped a lot in multithreading. When multiple threads work on common resource or depository each ready to modify the contents of common resource, it helps in modifying one at a time so that data is not corrupted. Although the sequence is still unpredictable, but at least data is not corrupted.
12. Interfaces for isolation. For example, while constructing a BuildAuto object, only its CreateAuto methods should be accessed. Similarly for updating only UpdateAuto methods should be accessed. Also interface programming is a good way of programming. For example, we have linkedhashmap, treemap all interfaces for say class map. Say tomorrow a bug is found in treemap, then I can with a lot of ease replace it with linkedhashmap. Definitely the testing cycle restarts and some changes will still be required, but it is still better than complete redo.

13. API construction helped in neat isolation of a specific set of methods associated with a class. For example, we can divide entire functionality of a particular class into sections and access each section via an interface. There is no smudging between 2 sections. Each section can work independently. Also it is easily extendible.
14. An interface helps in providing isolation between methods of an API. Any method related to a particular set of functions can only be accessed by an external user if it is present in that particular API. For example an API implements 2 interfaces A and B. Then an object created through A cannot access methods described in B.
15. Properties file format for easy loading of data. This way helps a lot in easy extraction and loading of data from a file to a database. Since the format is constant, the user never has to open the file and look at specific parsing to be done for each file. A common method for all files with easy data extraction.
16. Client Server model implementation helped a lot in data transfer. Today with iot being one of the most sort after fields, understanding the concept of sockets, clients and server was very helpful.
17. Protocol establishment between the client and server. When client and server are in same project, the execution steps can be sequential. But that is not a practical use case. Most of the times, the server and client are miles apart. The methods written to access all kind of data are common, so it is important to properly typecast the data to be received. Also not everyone should start talking at the same time. So a protocol in this case is very important.
18. Initially the server and client were listening on separate localhost and ports in spite of mentioning localhost as the string in the socket methods. Then it was tested and same source host inet address and port was passed while socket endpoint creation.
19. Serialization for data transfer. Serialization is a nice way of sending any kind of data to any place whether it's a file or over a socket.
20. It is important in serialization that if the object passed is of a class type, the class path in both the projects should be same. In my client and server initially the path of Automobile was different in Unit5. Because of this, my client always gave an exception of ClassNotFound on reading the object from ObjectOutputUser. This eventually lead to nullpointer exception.
21. LinkedHashMap implementation. Iteration through LinkedHashMap. LinkedHashMap provide a very clean and concise way of accessing a key, value pair. This kind of database is very useful for searching a needle in a haystack of data. Also they allow any kind of data to be put in them with proper typecasting during declaration.
22. ArrayList implementation. Iteration through ArrayList. ArrayList offers a much better way of implementation over an array. A very important advantage is that an initial size is not required for ArrayList. This saves a lot of memory space. The value can be assigned as and when the need arises.

23. OOPs concepts of inheritance and encapsulation. The concept of inheritance helps a lot in removing the code repetition. A lot of common functionality between classes can be shifted to a common class. Also it helps a lot in testing. We just have to check that a certain section of code works perfectly and then just isolate it.
24. Inner class usage can be done when a certain set of data is common. But its access is to be restricted to just the class that encapsulated it.
25. JDBC connection to connect to MySQL. MySQL is a very popular RDBMS. It is also very useful in storing data and is used by a lot of people in the industry. Even though I restarted my server many times, the data in mysql (or any other) database was always available for testing. This saved a lot of initial loading time. Also for huge data, storing data in some server and connecting with it is better than keeping everything in jvm. Also if one machine goes down, other machines still have continuity with database and can provide with the information.
26. Consistency between the data in database and linkedhashmap is very important. Just updating one and not other is a very bad design. The client interacts with linkedhashmap, and hence it should have the most updated data. Also the linkedhashmap is updated at start up from the database, so proper changes should be replicated over here as well.
27. MySQL queries execution through java code. This helps in reducing the overhead of switching between the java and sql code. One place controls all the operations.
28. Reading the contents of a file in java. Files are one of the most common source of input. Most of the data is stored in a particular format in text files. So learning how to read a file using bufferedReader helps to safely and easily load data from a file into a database.
29. Parsing a string based on a delimiter using StringTokenizer. A unique identifier helps a lot in reducing the amount of storage. It also helps in separating different kind of data within a file. Learning how to properly parse a String is important.
30. Solving and identifying causes of NullPointerException. Nullpointer exception is one of the most frequently encountered errors in java. Ability to identify and solve it quickly is a task in itself. But this course has taught me to find it fast and solve it quickly.
31. For serialization and deserialization, the object has to be typecasted each time. Serialization process converts the objects into bits and hence must be storing data about object structure as well.
32. Creating servelets and creating jsp files using tomcat for web development. Servlets provide an easy way of posting data on webpages. They also provide a very clean way of interacting with the user, wherever he might be present.
33. doGet and doPost methods in servelets. They help to cleanly obtain data to and fro from web.
34. While working with ObjectInputStream and ObjectOutputStream in openConnection, the order of their calling is important. Initially I was calling ObjectInputStream first. But it was giving me wrong results and hence null pointer exception. But calling ObjectOutputStream first followed by its flush solved the problem.

35. BufferedReader is synchronized.
36. The advantage of exposing methods using different interfaces is that a user cannot access and modify any other field or method using 1 object it has caught hold of. Apart from isolation it is important from security point of view. If a malicious user has access to 1 object, it cannot destroy the entire functionality. Using API and interface logic, at most only 1 section will be affected.
37. An abstract class or interface are like contracts and help in unifying the implementation across various different entities.
38. Advantage of implementing interface methods in abstract class are that all those methods which are common to all classes can be defined in abstract class. All classes which then inherit this abstract class can then define all the methods implemented by interface but not by abstract class. And since abstract class implements interface, it will force the sub class to implement those methods.
39. Custom exception handling is very important for any kind of system. A running system should not only report exceptions but also self- heal either by taking default value or by notifying the user and terminating the system.
40. For an inner class to be serializable its outer class must be serializable as well.
41. By exposing the fix method in custom exception handling, we can define the range of exceptions associated with a kind. Also we can call the fix method to decide which exceptions we want to solve and which are ok to live with. For example, in java null pointer exception terminates the system but classNotFound simply prints and goes ahead.
42. The Automobile object was made static in ProxyAutomotive class, so that irrespective of how many objects of BuildAuto are made, everyone is able to access the entire automobile repository. The automobile data is common and should be accessible to all at all times.
43. The choice variable was introduced in OptionSet class so that anyone with an object of Automobile can access it with just 1 layer of method call. Also all the choices can iterated together at one go instead of going back and forth between Option and OptionSet continuously.
44. I did CRUD operations in Automobile. I had to define separate methods to access each element in OptionSet and Option. This helped in solving the workflow method. There is a definite method of changing any value and that is possible only through Automobile.
45. Strategy that can be used for creating a race condition for testing are creating more threads and printing at various steps with thread id of printing thread attached. This can show us how each thread gets scheduled.
46. Synchronization although improves the chances of data not becoming garbage but causes a bottleneck as at a time only 1 thread can work on the resource. So the synchronization should be as small as possible.
47. By using encapsulation we wrapped the data and code into a single unit. This allowed us to efficiently manage the code and data together without any handling issues.

48. Commenting in code helps in the long run to easily understand logic implemented. Also very useful for future reference by other employees.
49. Good coding practices of line indentation, braces matching are very important, meaningful name of the methods and variables are very important.
50. Unbuffered input/output requests handle one byte at a time and hence are very time and resource consuming. Buffered input/output interact with the disk in larger-sized chunks hence it is faster and more efficient in the case of large files.