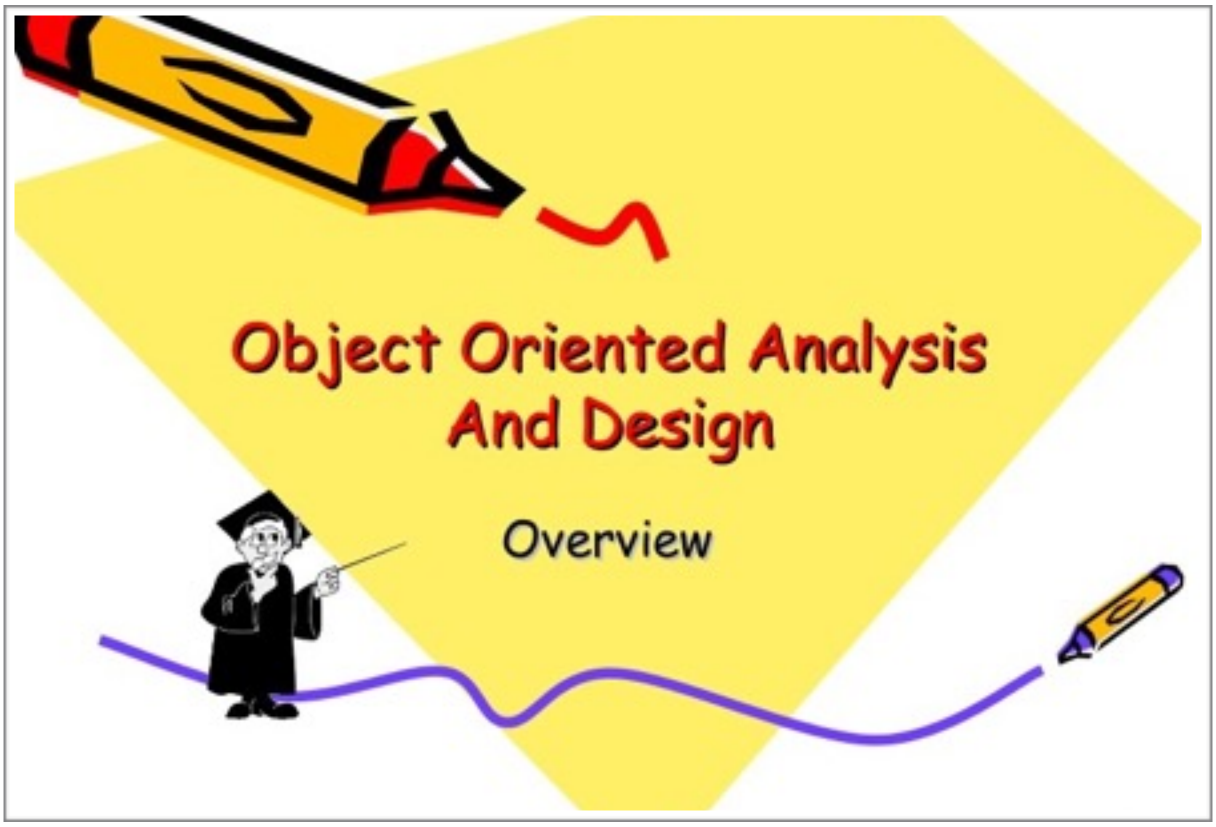
Lessons Learned of Mini1



Yi Cao (Andrew Id: yc2)

Spring of 2016

Lessons Learned of 18641

yi cao (andrew id :yc2)

Abstract

Mini 1 of Java smart phone(18641) mainly focused on the OOAD(object-orientated analysis design). From mini1, I got familiar with the basic concepts and terminologies of object–oriented systems. Through assignment1 to assignment2 , I learned what is object, how to build object, the relations of objects and FileIO. Through project unit1 to unit6, I learned the key concepts on design pattern, multithreads, database, web socket, servlet, JSP, network protocol, serialize, and the key points of OOAD including encapsulation, association, containment, inheritance and polymorphism. This article would include my understandings or lessons learned from mini1.

PART 1: OOAD

1. what is object: Objects can be modeled according to the needs of the application. An object may have a physical existence or a conceptual existence.
2. what is class: A class represents a collection of objects having same characteristic properties that exhibit common behavior. It include the operations and attributes of object.
3. what is encapsulate: It is the process of binding both attributes and methods together within a class. Through encapsulation, the internal details of a class can be hidden from outside. And customers can only access the elements of class outside.
4. what is inherent: It permits new classes to be created out of existing classes by extending and refining its capabilities.
5. what is polymorphism: a object could be implemented in many forms but with a common interface. Inheritance would be one of the important way to implement polymorphism.
6. what is association: It describes the relations between objects with common structure or common behavior.
7. what is aggregation: if A is an aggregation of B, then A is part of B.
8. what is composition: if A is a composition of B, then A is part of B but if B doesn't exist, the A would not exist any more.
9. what is UML: UML is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented design methodology.
10. what is the key points of OOD: low coupling and high cohesion.
11. what is the advantage of serialization: serialize could help to save information to memory and keep it from losing.

PART 2: Design pattern, multithreads, database, web
12. What role(s) does an interface play in building an API: Interface encapsulates all operations of the API, but it doesn't have function body to explain the operations in a detail. It has to be implemented by proxy classes to explain the details of the operations
13. What is the difference between abstract class and interface: abstract classes are a re-use mechanism for the class hierarchy for behavior and structure which isn't complete on its own. Interfaces are mechanism for specification of requirements on a module (e.g. class) independently of the concrete implementation. All other differences are technical details, important is different usage
14. what is the advantage of interface and abstract class: the interface and abstract class could be implemented in many forms to realize different functions
15. how to addresses the needs of exception handling and recovery: we could write a package to contain all customer exceptions and its handling or recovery.
16. What are good conventions for making a Java class readable: documentations are very important to make the Java class readable.
17. Why we need exception handling: When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore these exceptions are to be handled.
18. What is the difference between the LinkedArrayList and array: In project1, we mainly utilize LinkedArrayList as our datatype, because ArrayList is more fast to add and delete.
19. Facade design pattern: The BuildAuto has no specific operations inside but it is an outer interface for customers to call and it almost contain all CRUD operations for models. It is a kind of "Facade Design Pattern". Customers can only call functions to operate but do not the implementation details. It is good to maintain the safety of the whole system.
20. Adapter Design Pattern: adapter design pattern means Converting the interface of a legacy class into a different interface expected by the client so that the client and legacy class work together with any changes to the client or the legacy class. In project1, the CRUD operations are implement this design pattern so that even the legacy class change, customer can still CRUD as usual.

21. Why we need multithread: Multi threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application. Thus, the speed would be much faster than before.
22. A good way to set up multithreading: we can use synchronization solve concurrency issues so that that only one thread can access the resource at a given point in time.
23. What can use to test multithreading: we can use Thread Pool to test whether the system has concurrency issues after applying multithreading.
24. What is http protocol: Http is a protocol based on TCP. User enters information into form fields or controls. The browser then packages the data, opens an HTTP connection and sends the data to a server or an email address.
25. What is servlet and why we need servlet: Servlets provide a component-based, platform-independent method for building Web-based applications.
26. What is JSP and why we need JSP: JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
27. How Synchronization works in JVM: JVM puts a lock on the monitor of the object or the piece of the synchronized code, which gives it exclusive access to that part of code or object.
28. What is database normalization: Database normalization is the process of organizing the columns and tables of a relational database to minimize data redundancy.
29. Why we need database normalization: Normalization involves decomposing a table into less redundant tables without losing information; defining foreign keys in the old table referencing the primary keys of the new ones. 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.
30. Why we'd better to create table from text file: we read commands from text file or properties file to create tables because we can adjust the process of creating table easily by changing content of input file.
31. What is relation table: relation table mainly means tables have foreign key and reference key and so that those table would not be that redundant.

32. Why we have to define the primary key for a table: If we have a primary key, then we could use the primary key as the unique identity for the row element, it would avoid conflicts.
33. What is database transaction: A transaction is a unit of work that is performed against a database.
34. What are the common steps to design database:
*Determine the data to be stored in the database.
*Determine the relationships between the different data elements.
*Superimpose a logical structure upon the data on the basis of these relationships
35. What does sql mean: Structured Query Language; it is used to manage data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).
36. What is transaction ACID:
ATOMICITY: A transaction should be done or undone completely and unambiguously
CONSISTENCY: A transaction should preserve all the invariant properties (such as integrity constraints) defined on the data
ISOLATION: Each transaction should appear to execute independently of other transactions
DURABILITY: The effects of a completed transaction should always be persistent.
37. How to select the primary key and foreign key of relation db: as for project1, I choose the tables' id as primary key or foreign key, because id is unique and would be valid to avoid conflicts
38. How to delete a relation db with cascade: if we delete the upper db which serves as reference for lower db, then the lower db would be deleted as the upper db deleted.
39. How to print the table to the console: when we create tables, add models, update models or delete models in the db, in order to display tables in the console to verify our operations, we should know the elements of those tables and print them to console in a standard format.
40. What kind of data type can be saved in db: db can save almost all kinds of data type, the important thing is how to save those datatypes , how to index them and how to query them.

PART3: Overall understanding

41. what is the disadvantage of reading file like text files or database in a single pass and not use intermediary buffering: If we read file in a single pass, the speed of accessing data would be much slower than using intermediary buffering.

42. Good conventions for making a Java class readable: When we design java class, it would be better to document java files. Besides, we should try to module our design classes so that the logic would be more clear.

43. Strategies to design core classes so that they are reusable, extensible and easily modifiable: We could use interface or abstract classes to make our design reusable, extensible and easily modifiable;

44. Ways to analyze data to design objects: when we analyze data, we should focus on what are actors, what are boundary classes, what are interfaces, what are controllers so that when we implement, we could deploy our design more easily.

45. The relationship between abstraction and encapsulation:
Abstraction: Just like a Abstract of a document.Similarly in Java / OOP, an abstract can probably define a type. Encapsulation: Just like the english definition, hide something through some accessibility rules. In Java achieved by the access modifiers applied to both the data and the behavior.

46. Compile time polymorphism: It means the method overloading in java. In simple terms we can say that a class can have more than one methods with same name but with different number of arguments or different types of arguments or both.

47. Runtime polymorphism: Method overriding is a perfect example of runtime polymorphism. In this kind of polymorphism, reference of class X can hold object of class X or an object of any sub classes of class X

48. Issues when using Serialization with inner classes: 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. And synthetic fields generated by Java compilers to implement inner classes are implementation dependent and may vary between compilers; inner classes cannot declare static members other than compile-time constant fields.

49. Java heap: The heap is created on virtual machine start-up;The heap is the run-time data area from which memory for all class instances and arrays is allocated.

50. Java stack: Each thread has its own stack that holds a frame for each method executing on that thread. The stack extends vector. The stack is not directly manipulated, except to push and pop frame objects, and therefore the frame objects may be allocated in the Heap and the memory does not need to be contiguous.

Overall:
Those two assignments and project1 of mini1 help me get familiar with Java SE, Database, Java Web, Java OOAD and many other important technique details. From this course, I find only through practices can the theories be solid and only through bug tries can finally get bug-free and elegant projects.