Team 11

Name: Ke Wu

Andrew ID: kewu

1. Technical Point: Java files should be packaged to different packages based on their function. Those java files that implement the same function should be packaged together.

Example: We put all activity java files in ui package, and all entity classes (Course.java, User.java) in entities package.

1. Technical Point: Abstraction: Object-Oriented Programming is considering all things as objects. A object modularizes some data and functions and can be used as template for creating such modules on demand.

Example: We defined entity classes in entities package. The presentation tier will call methods of entities (objects), these methods will do the real job (e.g. fetch data from database)

1. Technical Point: Encapsulation: objects can interact with each other without knowing the details of each other's data or code. The data hiding helps the programmer to build secure programs.

Example: In the BackendConnector.java, other objects can just call the methods of BackendConnector (e.g. BackendConnector.getMembersByCourse()) without knowing details of how it is implemented (e.g. how it communicate with server).

1. Technical Point: Inheritance: object can extend from other objects which make it possible to code-reusing. It is a "is-a" relationship between classes.

Example: All activities in our project is inherited from AppCompatActivity and there are many methods are inherited from AppCompatActivity directly (e.g. onPause(), onResume(), etc.).

1. Technical Point: Static variable and static methods are used for a class to describe the thing that is shared by every instance of the class.

Example: in startActivityForResult() and onActivityResult(), there are some static request codes to indicate which intent is returned.

1. Technical Point: Public, private and protected give access control of a class (scope management). It helps encapsulation.

Example: In Course and User class, you can not access its attribute directly. Instead, get() methods must be used to get data.

1. Technical Point: Exception handling decreases the probability that program crash. make program robust.

Example: We defined many Exceptions (e.g. DatabaseException, InputInvalidException) to indicate errors.

1. Technical Point: Based on exception handling, we can build self-healing software so that the program is more robust.

Example: After the ui layer (activities and fragments) catch exceptions, it will prompt some dialogs to tell user there are some errors so user can re-input.

1. Technical Point: Abstract method gives method name and parameter without the implementation details. It is useful to describe a behavior in an abstract way. It can be used to achieve polymorphism.

Example: In the server side, there is a abstract class DBConnector that contains some abstract methods. And the ClassDiscuzDBConnector implements these abstract methods.

1. Technical Point: Interface is similar with abstract method. But interface can be used between different classes. It captures similarities between unrelated classes without forcing a class relationship.

Example: there is an interface PromptDialog and all exception classes implement the interface.

1. Technical Point: Inner class is a class that defined inside another class. Inner classes can be hidden from other classes in the same package --> another kind of encapsulation.

Example: AsyncTask class ans onClickListener class are inner class of Activity class.

1. Technical Point: Serializable is a special interface for class. The interface makes it possible to write a instance of the class to a file and read the instance back from the file.

Example: We make some classes (e.g. QBDialog) serializable so that it can be put into Bundle (e.g. args.putSerializable(EXTRA\_DIALOG, param1)).

1. Technical Point: Collection groups multiple elements into a single unit, it can be used to store, retrieve and manipulate a set of instances with same type (class).

Example: We use ArrrayList<Course> to get all courses that one user have registered**.**

1. Technical Point: Map is a special class that can be used as hash table. It gives O(1) time to look up.

Example: We use HashMap<Integer, Integer> to map userid to chatid in ChatAdapter.

1. Technical Point: Iterator can traverse a collection form start to end and delete elements from collection safely.

Example: We use iterator to traverse ArrrayList<Course>.

1. Technical Point: Java multithreading provides the opportunity to run multiple things at the same time. It can be used, for example, making multi-clients to run simultaneously.

Example: When the program need to fetch data from database, the ui thread will new an AsyncTask to do the job (doInBackground()).

1. Technical Point: Threads can communicate with others using PipedOutputStream and PipedInputStream. Communication between threads is important for the cooperation of threads.

Example: When an AsyncTask finish it job, it will tell ui thread (onPostExecute()).

1. Technical Point: Socket makes it possible for Java program to send/receive data to/from network

Example: BackendConnector connects to server side using HTTP GET and POST, which is based on TCP socket.

1. Technical Point: Based on socket programming, java programs can be divided into client and server. It separate different modules of program completely.

Example: In our project, there are client side (android side) and server side.

1. Technical Point: There are more failures in socket programming (network error). It is important to include exception handling when network comes in.

Example: In BackendConnector.java, our program will check the result returned from server side to detect error.

1. Technical Point: When server accept a connection with a client, it should make a new thread to deal with the connection. Server should not communicate with the client by itself, because it must be ready to accept connection from other clients.

Example: In server side, we use servlet. When a new request comes in, it will new a thread to deal with the request.

1. Technical Point: Servlet provides a interface for server program to deal with HTTP requests. It separate Response-generating program from other parts of server.

Example: In server side, we use servlet. We write different functions to deal with different request from client side.

1. Technical Point: Servlet provides different methods for different types of requests. It gives more flexibility for server to generate response based on types of requests.

Example: In server side, we use servlet. We write different functions to deal with GET request and POST request.

1. Technical Point: Database provides a better way for data saving and looking up, especially when data scale becomes large.

Example: In server side, we have a database to store the data of user, course, and user\_select\_course data.

1. Technical Point: Database is also a good choice when multiple threads need to read/write data simultaneously.

Example: Multiple clients can connect to server and asking for data simultaneously. Then there will be multiple threads of server to fetch data from database.

1. Technical Point: Normalization is import in database schema designing. With normalization, we can minimize redundant and inconsistent data and avoid update anomalies.

Example: We design our database schema based on normalization.

1. Technical Point: Build ER model will help database designing. It helps us to understand relationships between data in a clear and precise format.

Example: We design E-R diagram to help database designing.

1. Technical Point: JDBC provides an interface for Java program to connect and interact with database. A common way to create a JDBCAdapter class to invoke JDBC-related functions.

Example: In the server side, we use JDBC to connect to database.

1. Technical Point: JDBCAdapter should include some helper function to make it for other methods to interact with database. JDBCAdapter should also be responsible to transform database response to common Java data structure.

Example: In the server side, we use have a DBConnector class which is a JDBCAdapter.

1. Technical Point: Web programming provides another way for clients to communicate with server (through HTTP). In some situations, it is user-friendly than socket programming.

Example: In server side, we use servlet so that client side can send request to server based on HTTP.

1. Technical Point: Select statement provides methods for querying in a database. Combined with other clauses, select statement can be very powerful for querying.

Example: we use select to get the registered courses of one user, or all users that registered one course.

1. Technical Point: Before programming, a class diagram should be drawn. Class diagram is extremely useful for helping us understand the relationship between classes and what data/methods each class should contain.

Example: we have drawn a class diagram when we design our app, it really helps.

1. Technical Point: There are a couple of relationships between classes, including association, containment, encapsulation, inheritance and implement. Classes with different relationships work together in different way.

Example: In our project, the relationship between classes include association (Activity-Activity), containment (Activity-onClickListener), encapsulation (entities), inheritance (Activity-AppCompatActivity), etc. All of these classes work together in different way.

1. Technical Point: When design class, abstract class and interface should be considered to provide flexibility so that the class can be reused.

Example: In our project, we have a interface PromptDialog. It provides flexibility so that other classes can implement it as needed.

1. Technical Point: Log file should be used to record errors of running program.

Example: In our project, we use Log.i() and Log.d() to help us development.

1. Technical Point: All programs should be test sufficiently to make sure programs always work correctly and meet the specifications.

Example: We take many test cases to test our program, including normal case and corner case.

1. Technical Point: Template (generics) makes it possible to reuse code without copy-paste where the only difference is just the type of data.

Example: We use ArrayList<Course> to get all courses that one user have registered and we use ArrayList<User> to get all users that register the same course.

1. Technical Point: An Android Virtual Device (AVD) is an emulator that can be used to emulate real android machine, it is very useful for development.

Example: Before we got the Nexus 7, we use AVD to debug and develop our app. Even after we got the Nexus 7, only one member will use it to develop and the other people will use AVD.

1. Technical Point: Activity is the basic component of android application. It can be used to show view of application.

Example: We use a lot of activities to construct ui of our application.

1. Technical Point: Menu component can be used for some buttons that is rarely used in android application.

Example: We put “setting” and “focus” buttons in menu. They are not used as frequent as other buttons (like chat).

1. Technical Point: Fragment is useful to divide an activity to several parts so that only some parts will be replaced or changed when user give some input.

Example: We use two fragments in HomepageActivity so that user that switch between course schedule page and chat page without starting new activity.

1. Technical Point: Google map service is a powerful service for android application. It can be used to show map and mark location.

Example: We use Google map in our project to show the detailed location (e.g. which building) of course classroom.

1. Technical Point: Multimedia makes android application colorful, with multimedia user can take data of voice, image and video besides just plain text.

Example: In our project, user can upload avatar (from gallery or camera). When user chat with others, the chat message will not only show chat text but also the avatar of the message sender.

1. Technical Point: when android application need to fetch data from database, it should not do it by itself. Instead, it should new an AsyncTask so that ui thread can continue to respond to user’s input.

Example: In our project, when we need to talk to server side (e.g. call function of BackendConnector) we will new an AsyncTask to do it.

1. Technical Point: Polymorphism: the behavior may exhibit different behavior in different instances, it depends on the types of data used in the operation. It is important so that different instance will exhibit different behavior.

Example: In the android project, all activity is inherited from AppCompatActivity. However, they will override some methods (e.g. onCreate()) so that different activity will have behavior.

1. Technical Point: For scalability, instead of hard coded string and value in the code. We should put them in a separate file so that it’s easy to modify.

Example: In our project, we put all string value in res/values/strings so that it is easy to modify.

1. Technical Point: A ListView and an array adapter can be used so that it is easy to show a list of items.

Example: In our project, we use ListView and customized adapter to show courses list, chat members list, etc.

1. Technical Point: Content provider is the local database of android application which can be accessed by SQLite. It can be used to store local data of application.

Example: In out project, we use content provider and SQLite to store local data such as the profile of the user that login to application.

1. Technical Point: An android should be divided into presentation tier, application tier, integration tier, content provider, etc. Separating tiers are good for design and development.

Example: Our project is divided into presentation tier, application tier, integration tier, content provider and server side.

1. Technical Point: For proper design, program should not call the constructor of a fragment class directly. Instead, it should call a “static factory method” like newInstance().

Example: In our project, we implement a newInstance() for ChatFragment, ChatPageFragment so that when a new fragment is needed newInstance() method will be invoked instead of constructor method.