Architecture Reconstruction: SOOMLA Android Store System

Kennzahl	*Matrikelnummer*	*FAMILIENNAME* *Vorname*

Questionnaire

Question 1:
How many years of programming experience do you have?

Question 2:
How many years of Java programming experience do you have?

Question 3:
Do you have commercial programming experience? If yes, how many years?

Question 4:
Do you have experience in programming computer games? If yes, how many years?

Question 5:
Do you have experience in android programming? If yes, how many years?

SOOMLA Android Store Architecture

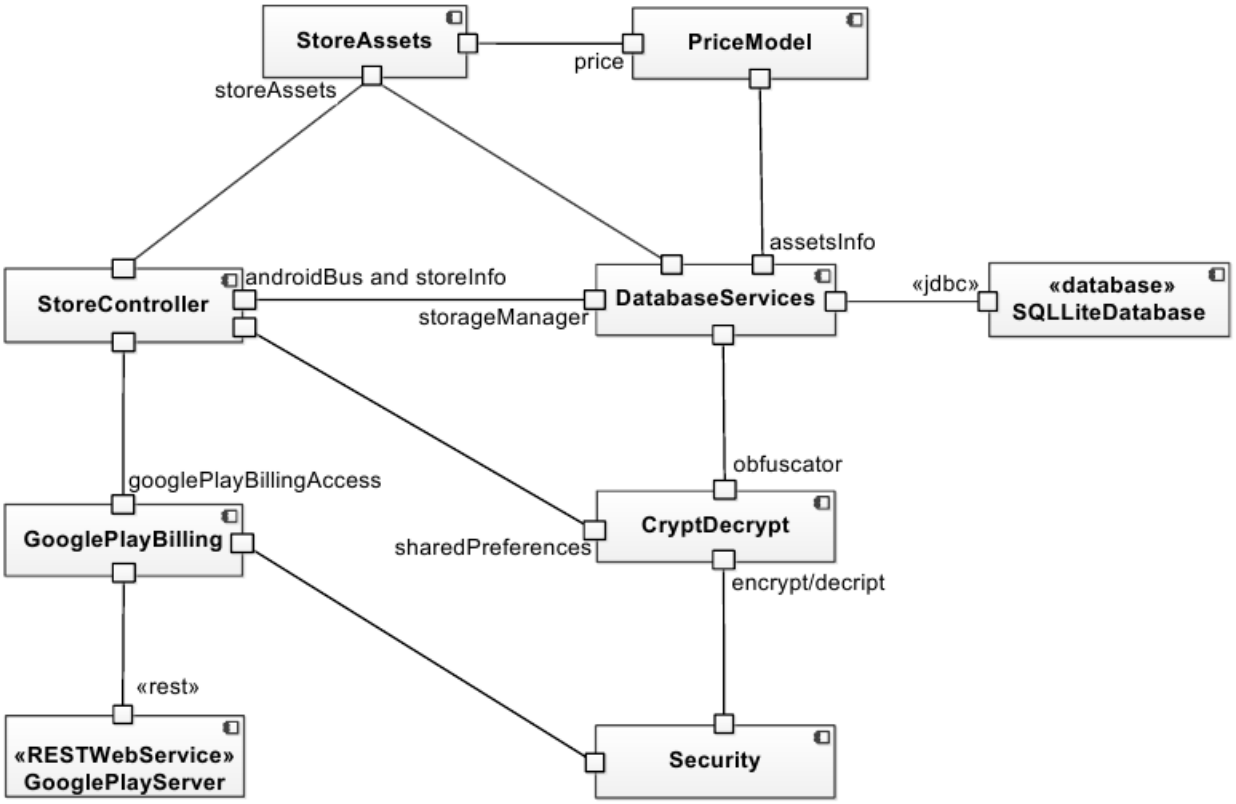

Figure 1. SOOMLA Android Store Architecture

The SOOMLA Android Store is an open source framework for supporting virtual economy in mobile games. This framework allows mobile game developers to easier implement virtual currencies (tokens, coins, gems, etc.), virtual goods and in-app purchases. The in-app purchase functionality supports two types of virtual goods: consumables and non- consumables. The main difference between these two is their durability. For consumable items the developer expects the user to consume the virtual goods over time and possibly to purchase them again. The *VirtualCurrency* is a form of consumable item. A specified amount of *VirtualCurrency* can be spent to accumulate *VirtualItems* (items available in the store for purchase). In case of insufficient *VirtualCurrency*, the user needs to purchase a *VirtualCurrencyPack* (e.g., 10 coins pack, 20 coins pack, etc.) which holds the *VirtualCurrency* and has a price (the cost of the pack). Non consumables, on the other hand, are expected to last forever and so this type of goods can be used to implement extra levels, a remove ads feature, or upgrading to a premium version of the game.

The high-level architecture design (shown in Figure 1) of the SOOMLA Android Store comprises of seven components, namely ***StoreAssets, PriceModel, StoreController, DatabaseServices, GooglePlayBilling, Security*** and ***CryptDecrypt***. In addition, there are two external components modeled: ***GooglePlayServer***, the REST Web Services running at Google, and ***SQLLiteDatabase***, the used database accessed over JDBC.

StoreAssets describes the virtual currency, virtual goods, and their classification. **PriceModel**

describes the model that explains how the prices of virtual items are formed. **StoreController**

provides the runtime functionality of the Android store and contains up-to-date store information. This component handles the purchasing and equipment of virtual items. In addition, the component initiates purchase and refund events, and performs expected actions in response.

DatabaseServices comprises of classes that communicate with the database. In particular, the classes enclosed inside this component perform the initialization of the database and implement retrieve, add, and remove operations for store assets in the database. The classes of this component also contain SQL queries to create database table as well as to retrieve, set, and overwrite the information in the database.

GooglePlayBilling simplifies in-app billing API which is a Google play service that lets you sell virtual goods from inside your applications. It also handles the responses from the *GooglePlayServer* regarding the corresponding billing requests. The *GooglePlayServer* is accessed using REST Web Services.

Security verifies the information during purchasing. It checks whether signature from the server matches the computed signature on the data and deals with obfuscation of values before saving to database and when retrieving from database.

CryptDecrypt contains classes that provides encrypt/decrypt services to obfuscate *GooglePlayBilling* information and to encrypt/decrypt the data stored to or retrieved from the database.

Traceability Links of Architectural Components to the Code-base Classes

The following traceability links summarize the relations of components in the SOOMLA Android Store Architecture to classes in the implementation code (that is, the component consists of the related implementation classes). This helps you to understand the links between the architecture and source code.

Component Name	Related Implementation Classes
StoreAssets	VirtualCategory.java, VirtualGood.java, VirtualCurrency.java, VirtualCurrencyPack.java, AbstractVirtualItem.java, VirtualItemNotFoundException.java, GoogleMarketItem.java, NonConsumeableItem.java, JSONConsts.java, VirtualGoodEquippedEvent.java, VirtualGoodUnEquippedEvent.java, CurrencyBalanceChangedEvent.java, GoodBalanceChangedEvent.java, IStoreAssets
StoreController	StoreInfo.java, StoreController.java, GoodPurchasedEvent.java, GoodPurchaseStartedEvent.java, MarketRefundEvent.java, MarketPurchaseEvent.java, MarketPurchaseStartedEvent.java, AndroidBus.java, BusProvider.java, InsufficientFundsException.java, NotEnoughGoodsException.java
DatabaseServices	StorageManager.java, KeyValueStorage.java, VirtualCurrencyStorage.java, VirtualGoodsStorage.java, NonConsumeableItemStorage.java, StoreFrontInfo.java, KeyValueDatabase.java (nested class: DatabaseHelper), StoreInventory.java
GooglePlayBilling	BillingReceiver.java, BillingService.java (nested classes: CheckBillingSupported, GetPurchaseInformation, ConfirmNotifications, BillingRequest, RequestPurchase, RestoreTransactions), PurchaseObserver.java, ResponseHandler.java, Consts.java
CryptDecrypt	AESObfuscator.java (nested class: ValidationException), Base64.java, Base64DecoderException.java, ObscuredSharedPreferences.java
PriceModel	AbstractPriceModel.java, StaticPriceModel.java, BalanceDrivenPriceModel.java
Security	Security.java (nested class: VerifiedPurchase)
GooglePlayServer	None (external, running at Google, accessed using REST Web Services)
SQLLiteDatabase	None (external, running in database server, accessed using JDBC)

Exploring the SOOMLA Android Store system

In this assignment you will explore the relationships between the classes within the corresponding components in the SOOMLA Android Store system architecture and the roles of the identified relationships. You will also explore the relationships between classes that are located in different components in the architecture and the roles of those relationships. There are 4 true/false statements for each component and you have to check the right answers among them. Please, study the statements carefully, especially in the case of bigger components where answering the question/statement requires studying of several classes.

Hint: You can get the required information from the following sources:

1. Component architecture and traceability links (can be used to study which components are interconnected and which classes they contain)
2. Source code (can be used to study the relations between the classes)
3. Comments in the source code (can help/explain the roles of the classes, relationships between classes, class methods, etc.)
4. You obtain the browser based source code access. You can use the possibility to search for other classes in the source code of the given class by simple using of CTRL+F function that will give you (highlight) all the places where the specified class that you are searching for appears. You can also open many tabs with different classes that will facilitate your search.

For browser-based source code access please use the next URL:
https://swa.univie.ac.at/soomla/

Important: Please indicate the time when you start and finish exploring each component. You can write it in the format **hour**: **minute** (for example 15:24). There are many time slots for each component so that you can write the time several times, for example if you would like to get back and study the given component a second or third time. We put the reminder at the beginning of each question.
Based on this we can show you how much time you needed to study each component, and it will help us to adapt the assignment to other students in the future in terms of the number of studied components, components size, etc.

Question 1: Exploring the component <u>Security</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True □ b) False □
   The class Security uses the class VerifiedPurchase as a nested class that holds verified purchase information.

2. a) True □ b) False □
   The class Security uses the classes Base64 and Base64DecoderException in order to decode the data from base64 notation and to catch an exception in case that base64 decoding is failed.

3. a) True □ b) False □
   The class Security uses the class AESObfuscator in order to obfuscate (make unclear) of values before saving to database (DB) and when retrieving from DB.

4. a) True □ b) False □
   The class Security uses the class ObscuredSharedPreferences in order to generate base64 encoded representation of the public key during the purchase process verification.

Question 2: Exploring the component <u>CryptDecrypt</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True □ b) False □
   The class AESObfuscator uses the class Base64 to encode or decode the data using base64 notation during the obfuscation of values (making the values unclear).

2. a) True □ b) False □
   The class AESObfuscator uses the classes ValidationException and Base64DecoderException to indicate that an error occurred during validating the integrity of data managed by the AESObfuscator and to indicate that the given virtual item that need to be obfuscated is not defined in the IStoreAssets.

3. a) True □ b) False □
   The class AESObfuscator uses the class ObscuredSharedPreferences to create the shared preferences (public, private keys) and to send them to the Android Market.

4. a) True □ b) False □
   The class Base64 uses the class Base64DecoderException to raise an event in case that base64 decoding was successful.

Question 3: Exploring the component <u>DataBaseServices</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True ☐ b) False ☐
   The class StorageManager uses the classes VirtualGoodsStorage, VirtualCurrencyStorage, NonConsumableItemsStorage, KeyValueStorage, AESObfuscator and KeyValDatabase to: create all relevant classes for the storage, get the database that provide basic key-value storage above SQLite, obfuscate the key-value pairs of data before saving to the database and after retrieving from the database, and store the virtual items in the database tables and the tables of the corresponding storage classes.

2. a) True ☐ b) False ☐
   The classes StoreFrontInfo, VirtualGoodsStorage, VirtualCurrencyStorage, and NonConsumableItemsStorage use the classes KeyValDatabase and StorageManager to: update the amount of the given virtual item in the storage, get the database and obfuscate the key-value pairs and the store metadata (JSON file) before saving it in the database.

3. a) True ☐ b) False ☐
   The class StoreInventory uses the classes StorageManager, VirtualCurrencyStorage, VirtualGoodStorage, VirtualGood, VirtualCurrency, and StoreInfo to be able to manipulate with the given virtual items and to get their definitions during providing the help to the storage related classes in managing store operations (get balances or remove, add items).

4. a) True ☐ b) False ☐
   The classes VirtualCurrencyStorage and VirtualGoodsStorage use the classes BusProvider, AESObfuscator, CurrencyBalanceChangedEvent (VirtualCurrencyStorage) or GoodBalanceChangedEvent (VirtualGoodsStorage) to: obfuscate the key-value pairs of data before saving them to the database and after retrieving them from the database, register for the given events by obtaining the bus, raise those events in case of the items equipping and increasing or decreasing their balances. Equipped items are those that are bought once and can be kept forever while the user can additionally equip them for specific usage.

Question 4: Exploring the component <u>StoreAssets</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True □ b) False □
   The classes VirtualGood and VirtualCurrencyPack can catch the VirtualItemNotFoundException to indicate that virtual category and virtual currency cannot be found in the StoreInfo definitions. The VirtualGood and VirtualCurrencyPack classes are also related to the class StoreInfo in order to get the information related to the virtual category or the virtual currency that is needed for the generation of virtual items instances from the JSON objects. The class VirtualGood uses the class AbstractPriceModel to associate the given good with a price model, to get the good price, and to convert the associated price model to the JSON object and vice-versa.

2. a) True □ b) False □
   The classes VirtualGood, VirtualCurrency, VirtualCurrencyPack, NonConsumableItem use the class AbstractVirtualItem to inherit the common features of all virtual items: generating instances form their JSON object representations, conversion to the JSON objects, etc. Two of the provided virtual items have their representations in Google Play (GoogleMarketItem) and each product of those two items can have status MANAGED, UNMANAGED and SUBSCRIPTION. Only one of the provided virtual items can be categorized using VirtualCategory where one virtual category can be associated with many items.

3. a) True □ b) False □
   The classes AbstractVirtualItem, GoogleMarketItem, VirtualCategory, VirtualGood, and VirtualCurrencyPack use the class JSONConsts for accessing of the all static final String constants defined in that class during the conversion to/from JSON objects from/to the given virtual item objects. The event classes in the component GoodBalanceChangedEvent, CurrencyBalanceChangedEvent, VirtualGoodEquipedEvent, and VirtualGoodUnEquipedEvent contain corresponding currency and good items and are raised within the component in order to indicate changes in balances of the given virtual items and changes of the virtual goods' equipping status.

4. a) True □ b) False □
   The classes AbstractVirtualItem, GoogleMarketItem, VirtualCategory, VirtualGood, and VirtualCurrencyPack use the class JSONConsts for accessing of the static final String constants defined in that class during the conversion to/from JSON objects

from/to the given virtual item objects. The event classes in the component GoodBalanceChangedEvent, CurrencyBalanceChangedEvent, VirtualGoodEquipedEvent, and VirtualGoodUnEquipedEvent contain corresponding currency and good items and are raised within the component in order to indicate changes of the virtual goods' equipping status when their balances significantly decrease.

Question 5: Exploring the component <u>StoreControler</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True ☐ b) False ☐
   The class StoreControler uses the classes GoodPurchasedEvent, MarketPurchaseEvent, GoodPurchaseStartedEvent, MarketPurchaseStartedEvent, and MarketRefundEvent to raise events related to the following actions: starting the currency pack purchase, indicating that virtual good is purchased and the balance of that good is changed in the storage, indicating that purchase state is changed and a user is charged for the order, indicating that virtual good purchasing is started, and indicating that used received the refund for the order.

2. a) True ☐ b) False ☐
   The class StoreControler uses the classes VirtualGood, VirtualCurrency, VirtualCurrencyPack in order to be able to manipulate with the given virtual items in the purchasing process, and it uses the classes StorageManager, VirtualCurrencyStorage, VirtualGoodStorage, and NonConsumableItemsStorage in order to: update the amount of the corresponding items, create new virtual items that are required, and equip the virtual goods.

3. a) True ☐ b) False ☐
   The class StoreControler uses the classes BillingService, PurchaseObserver, RequestPurchase, and Consts from the component GooglePlayBilling in order to: inherit the possibility of making UI changes according to the various purchase events, access the service that will send the massages to Android Market on behalf of the application, deal with the response from Android Market for the given purchase request, and use and update the global constants of possible purchase states and response code information used through the application to support the in-app billing.

4. a) True ☐ b) False ☐
   The class StoreInfo uses the classes StorageManager, KeyValDatabase and AESObfuscator in order to: put the store info in the database as JSON objects, obfuscate the key-value pairs of the given store assets before saving it to the database, and delete the un-obfuscated key-value pairs. In case of error during the validation of data integrity, it catches ValidationException.

Question 6: Exploring the component <u>GooglePlayBilling</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True □ b) False □
   The classes RestoreTransactions, RequestPurchase, BillingService and CheckBillingSupported use the class ResponseHandler in order to deal with the response information from the Android Market related to: transactions management, reporting errors and acknowledgements, starting of the activity for the user to buy an item, and purchase state changes and notifications about the availability of MarketBillingService.

2. a) True □ b) False □
   The usual data flow in the component GooglePlayBilling can be represented using the next sequence of relationships: class BillingRequest – Android Market – class BillingReceiver – class BillingService – class ResponseHandler – class PurchaseObserver. The sequence can be explained as follows: The class BillingRequest sends messages to Android Market using MarketBillingService, then the class BillingReceiver receives and forwards all received messages for handling the further communication with Android Market to the BillingService class, then BillingService notifies the application about purchase state changes using the ResponseHandler class which at the end updates the UI using the received information from the Android Market (posting appropriate events, updating currency balances, items, etc.).

3. a) True □ b) False □
   The classes BillingService, RestoreTransactions and GetPurchaseInformation use the class Security to improve the security during the purchasing process by: checking if the data was signed with the given signature, generating random numbers (Nonce) that are used for signing the data during getting the purchase information from Android Market and signing the transactions that are sent from Android Market, and verifying that restore transactions use the appropriate binary to hexadecimal format conversion.

4. a) True □ b) False □
   The classes BillingService, RestoreTransactions, ConfirmNotifications, RequestPurchase and CheckBillingSupported use the class Consts in order to access: the possible response code states defined by the Android Market (*RESULT_OK*, *RESULT_ERROR*, etc.), the intent actions defined by the application that are sent from BillingReceiver to BillingService, intent actions received by the Android Market that

cannot be changed, intent action used to bind to the MarketBillingService, and other global constants for creating Request bundles (packets).

Question 7: Exploring the component <u>PriceModel</u>

Please, write the stop time in the previously studied component in case you forgot and the start time in this component.

Time slots:

Start:												
Stop:												

Please tick the true checkbox for those of the following statements that are completely true for this component. If the statement is not completely true tick the false checkbox.

1. a) True ☐ b) False ☐
   The classes StaticPriceModel and BalanceDrivenPriceModel use the class JSONConsts to create the JSON constants according to the given price model.

2. a) True ☐ b) False ☐
   The class BalanceDrivenPriceModel uses the classes StorageManager and VirtualGoodStorage in order to create a new virtual good and save it to the corresponding storage.

3. a) True ☐ b) False ☐
   The class BalanceDrivenPriceModel uses the classes StorageManager and VirtualGoodStorage to remove the given virtual good from the corresponding storage.

4. a) True ☐ b) False ☐
   The class AbstractPriceModel uses the classes StaticPriceModel and BalanceDrivenPriceModel in order to create the appropriate StaticPriceModel and BalanceDrivenPriceModel objects with a given JSON object.