NATIONAL UNIVERSITY OF SINGAPORE

CS2113/T – SOFTWARE ENGINEERING AND OBJECT ORIENTED PROGRAMMING

(Semester 2: AY2018/19)

Part 2

Time Allowed: 1 Hour 30 min

INSTRUCTIONS TO STUDENTS

1. Please write your Student Number only. Do not write your name.
2. This assessment paper contains **FIVE** questions and comprises **EIGHT** printed pages.
3. You are required to answer **ALL** questions.
4. Write your answers in the space provided.
5. This is an **OPEN BOOK** assessment.
6. You may **use pencils** to write answers; but write legibly.

STUDENT NO:	**A**								

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/6	
Q2	/6	
Q3	/6	
Q4	/4	
Q5	/8	
Total	/30	

Wallet-gator – The payment wallet aggregator software.

You are developing a payment wallet aggregator software which is to be deployed in NUS. There are many payment wallet services (e.g., NETS, PayLah, GrabPay) that are operational in the campus. In version 1.0 of your app the customers can scan a QR code thereby selecting the appropriate wallet to make the payment. In future, you also plan to support automatic selection of the wallet service based on the discounts/offers that are available. Your software will be deployed as a mobile app. The user first has to register the wallet on the app before he can select it for making the payment.

1. [6 marks]

(a) Give the use case to scan a QR code and identify the wallet service. Include at least two extensions. [4 marks]

Usecase: Identify wallet service

1. User starts Wallet-gator.
2. Wallet-gator launches the scan screen.
3. User scans the QR code.
4. Wallet-gator identifies the wallet service based on the QR code scanned.
5. Use case ends.

Extensions:

3a. Camera view is blocked
3a1. Wallet-gator notifies the user to unblock the camera
3a2. User unblocks the camera
3a3. Use case continues from 3.

3b. User cancels the operation
3b1. Wallet-gator confirms the cancelled transaction
3b2. Use case ends.

4a. QR code belongs to an un-registered gateway/ QR code is invalid
4a1. Wallet-gator shows invalid QR code/gateway selected
4a2. Use case resumes from 3.

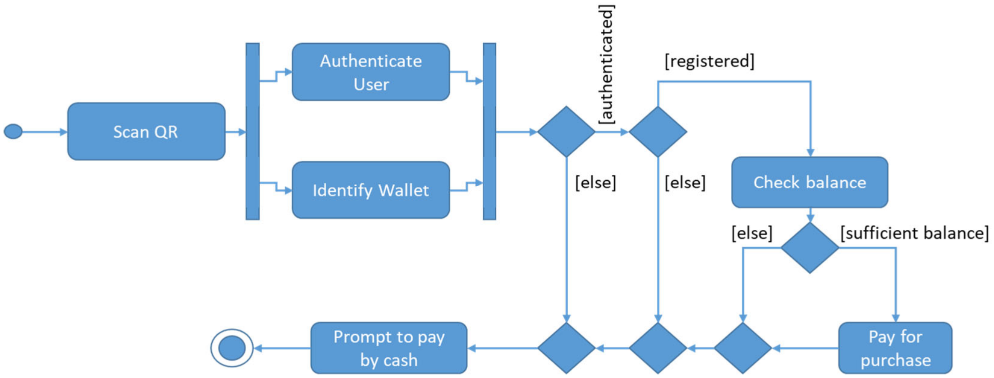
(b) Give one *must have* user story each from the perspective of the shopkeeper and the customer. [2 marks]

As a user I should be able to scan the QR code at the shop so that I can make the payment.

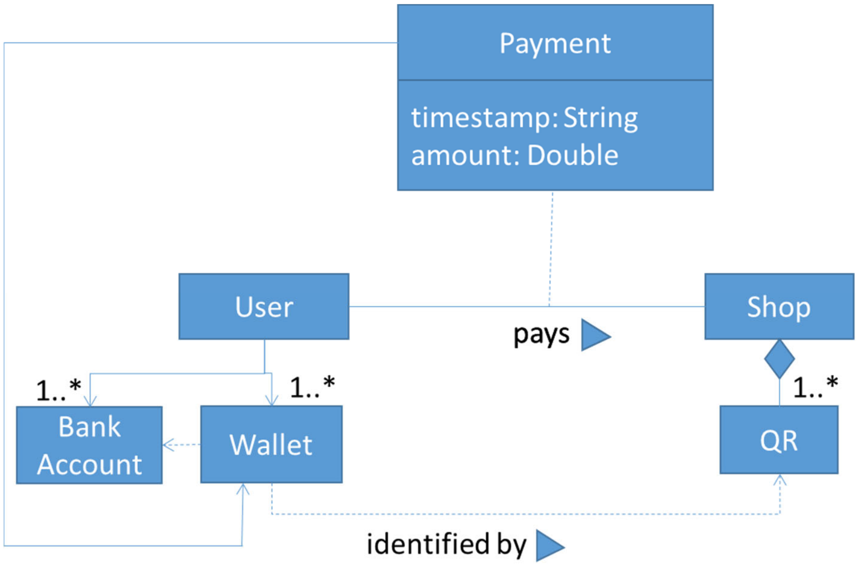
As a shop keeper I should be able to register a new QR code so that I can receive payment using a different wallet service.

2. [6 marks] Upon scanning the QR code, the app authenticates the user and in parallel identifies the wallet service selected. If the wallet selected is already registered by the user, the app checks for the balance amount. If the balance is sufficient the app proceeds to allow the user to pay for the purchase. Alternatively, in case of error conditions such as unable to authenticate the user or if the wallet is not registered or the selected wallet has insufficient funds, the app prompts the user to pay for the purchase with cash.

Model the workflow of the process described above using an appropriate UML diagram.



3. [6 marks] The user has to register at least one wallet before (s)he can use Wallet-gator. (S)he should also associate at least one bank account with Wallet-gator to enable recharging the wallet from within the app. The wallet service is identified by the QR code scanned at the shop. A shop can have multiple wallet services registered (i.e., multiple QR codes) to receive the payment. The user pays a shopkeeper by scanning the registered QR code; the payment details are recorded by Wallet-gator. The details include the wallet service used, the timestamp (String) of the transaction and the amount (Double) paid.
   (a) Model the structure of the described above using an appropriate UML diagram.
   (b) Include all details (such as navigabilities, multiplicities etc.,) given in the description in your answer



2. [4 marks]
   (a) Which design pattern do you think will be most useful in implementing the payment mechanism? Justify your answer briefly. [2 marks]
   Façade - for each gateway, by scanning just the compatible QR code, the internal system should resolve this to the correct wallet and make the payment.

   (b) You intend to support other means of payment like Apple Pay and Samsung Pay in your solution. Which design principle will help you the most? Justify briefly. [2 marks]
   OCP; core is fixed, but can extend.

3. [8 marks]
   Consider the code below to answer sub-questions (a) and (b). (Assume all methods used are known to the object and all necessary imports are included).

```
/**
 * This method reads the QR image and gets the associated text for
 * further processing.
 * @param file_path can be the location of the socket connecting
 * to the camera or a file location
 * @param hint_map provides format instruction to the reader
 * @throws
 * @return
 */
public String readQR(String file_path, Map hint_map)
    throws IOException, NotFoundException, FormatException {
    try {
        BinaryBitmap binaryBitmap = readBitmap(file_path);
    } catch (Exception e) {
      showToUser("An error occurred");
    } catch (IOException ioE) {
      throw new IOException();
    }
    Result qrCodeResult = new MultiFormatReader().decode(binaryBitmap, hint_map);
    return qrCodeResult.getText();
}
```

(a) Based on the coding standards and concepts learned in CS2113/T, suggest at least 4 improvements to improve the code quality. [2 marks]

- Start of comment: Returns the text associated with …
- file_path/hint_map – violations
- empty @throws/@return
- Exception caught earlier than IOException
- Indentation error in the catch blocks
- No logging/assertion etc.,

(b) Does the code above violate Law of Demeter? Justify your answer. [2 marks]

- No it doesn't/ MultiFormatReader is created in the method, so can use the decode method.

(c) The QR code data is an alpha-numeric string that is 40 to 50 characters long and it contains at least one alphabet and one number. Wallet-gator has a method `isValidQR()` to validate the QR code data. Identify the equivalence partitions to test the `isValidQR()` method. [4 marks]

- Equivalence partitions:
    - NULL string
    - <40 characters
    - >50 characters
    - 40-50 characters, non-alpha-numeric (e.g. punctuations and non-ascii)
    - 40-50 characters, alphabet only
    - 40-50 characters, numeric only
    - 40-50 characters, alpha-numeric