EBU6304 Software Engineering

Exercises

Set 2

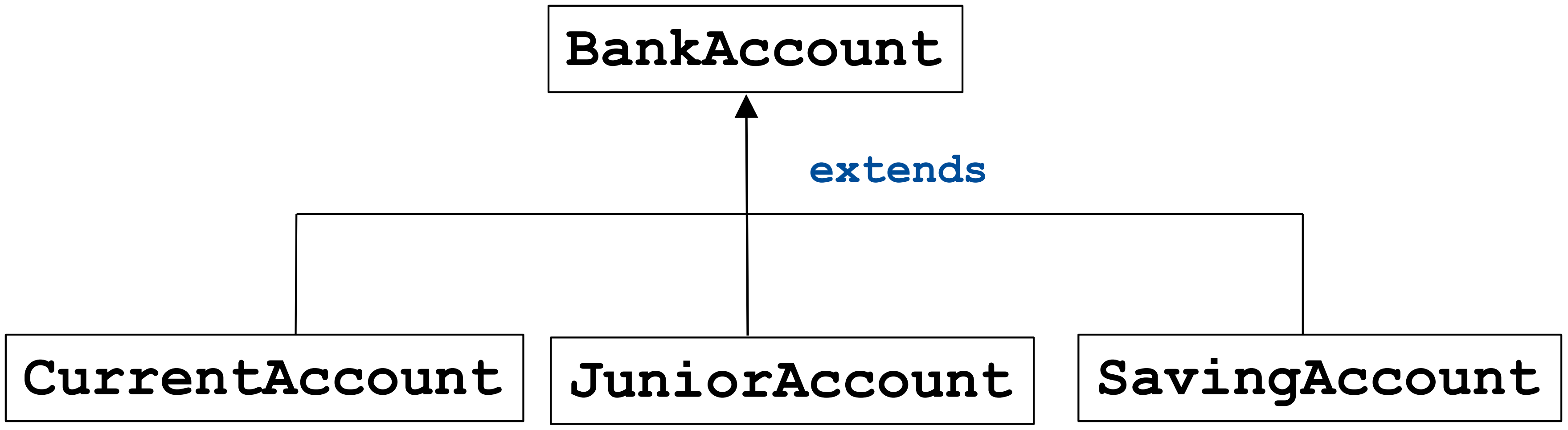
Analysis and Design

Q1

Describe the relationship between the Bank Account, Current Account, Junior Account and Saving Account with a UML class diagram.

Q1 model answer

Current Account, Junior Account and Saving Account are all child classes of a Bank Account. It is Inheritance relationship.



Q2

Read the following description in a user story:

"A digital library system is being developed for QMUL library. It is to be used by QMUL staff, QMUL students and public members. Some users can read books inside the library, but cannot borrow them, whereas other users can read and borrow books. Library admin staff can check in/check out books. Librarians can perform all of the above duties plus order new books."

This description is not well written. You need to do **analysis** to better understand the requirements. Identify **Entity** classes from the description and draw a UML class diagram to show their relationships.

Q2 work out…

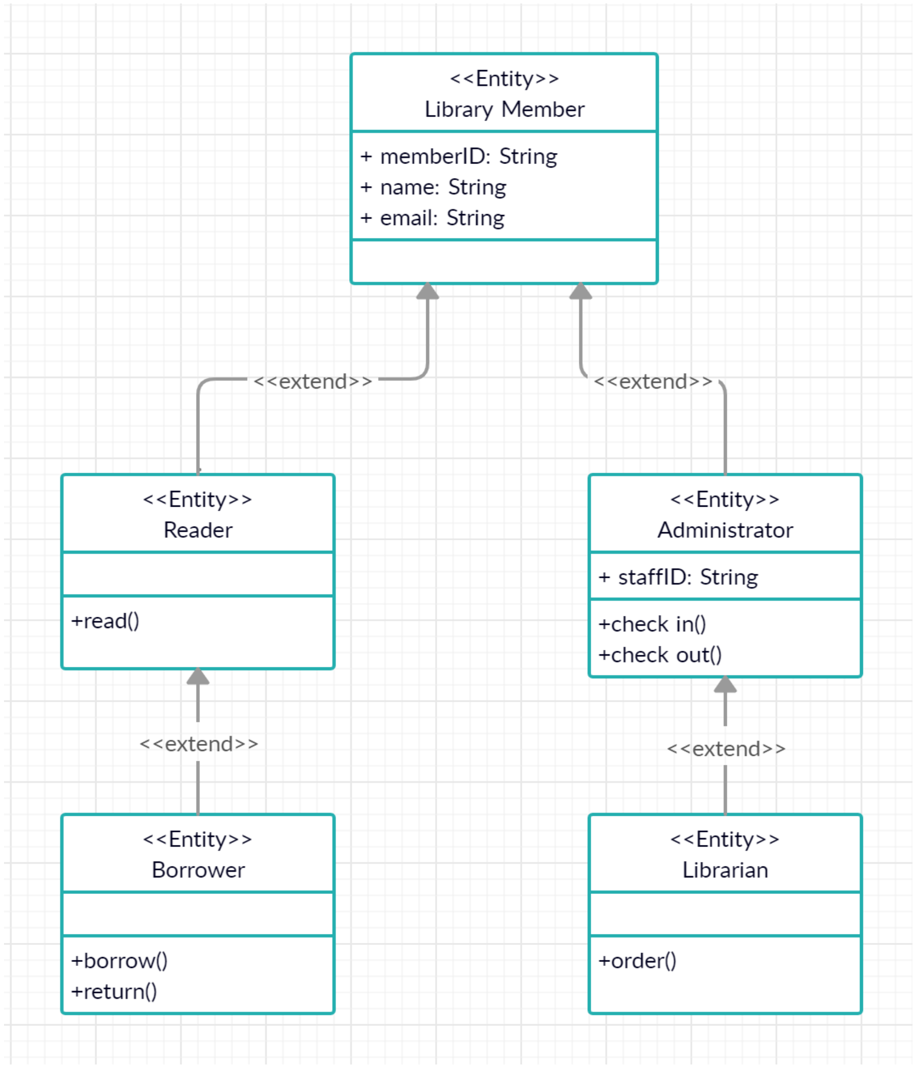
To get started, highlight nouns and verbs. Nouns could be candidates of class or attributes; verb could be candidates of methods.

As a business analyst, we need to identify roles and how the roles use the system differently, identify missing class, attributes and operations.

Q2 work out…

A digital library system is being developed for QMUL library. It is to be used by QMUL staff, QMUL students and public members. Some users can read books inside the library, but cannot borrow them, whereas other users can read and borrow books. Library admin staff can check-in/check-out books. Librarians can perform all of the above duties plus order new books.

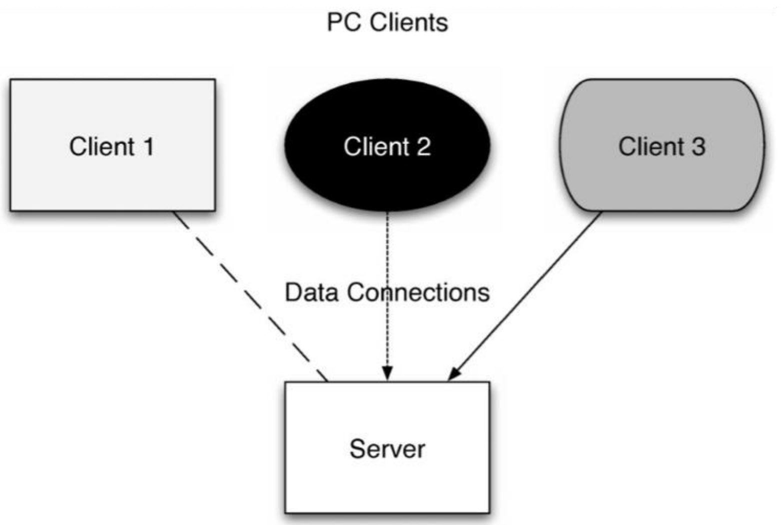
Q2 model answer



Software Architecture

Q1

"Box and arrow" diagrams sketched on a whiteboard are often sufficient to start reasoning about the architecture of a system. However, some issues can arise if one is too relaxed about the diagrams' notation. Consider for example the diagram below: can you think of any issue or potential source of confusion in it?



Q1 model answer

The issue with the sample diagram is that the boxes and arrows are drawn with different shapes, colours, and styles without explicating what is the significant of these differences.

All three connections are of labelled “data connection” so it’s not clear why they are drawn using different styles. All clients are labelled as “PC clients” so it’s not clear why they’re drawn using different shapes and colours.

These are unnecessary embellishments that have no meaning and only cause confusion.

Q2

Many APIs provided by companies are labelled as "RESTful". For example, the Twitter API is often called "Twitter REST API" [1], however it's hard to understand if this API is really satisfying the 6 (5 compulsory + 1 optional) architectural constraints of the REST architecture. Search the Internet for some evidences of why the Twitter API <u>may or may not be</u> a truly RESTful API.

(*Hint: there are some interesting discussions on this topic on stackoverflow*)

References:

[1] https://www.w3resource.com/API/twitter-rest-api/

Q2 model answer

Let’s recall the 6 architectural constraints of the REST architecture: 1) Client-server, 2) Cacheability, 3) Uniform interface, 4) Statelessness, 5) Layered system, and 6) Code-on-demand. Let’s ignore the last one, as it’s optional.

“Some well-known APIs that claim to be RESTful fail this sub-constraint. For example, Twitter’s REST APIs uses *RPC-like* URIs like **statuses/destroy/:id** and it’s the same with Flickr. The problem is that they break the Uniform Interface requirement, which adds unnecessary complexity to their APIs” [1]

Q2 model answer

For an API to be RESTful it needs to treat URI as what they are meant to be (i.e., resource identifiers) and not remote procedure calls, and use instead the correct HTTP methods instead [2,3] (e.g., GET, POST, and DELETE [4]).

References

[1] https://www.kennethlange.com/what-are-restful-web-services/
[2] https://stackoverflow.com/questions/29151523/are-rest-apis-really-restful
[3] https://docs.google.com/presentation/d/16bqtQz_sa_YgO6wlC8p-YK-ZTMWmrvyZiub7FupTT5M/edit

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Testing

Q1

Why does software have bugs despite developers' carefully following software engineering methodologies?

Q1 model answer

Perfect software development assumes software ships with no bugs, good enough software accepts that some bugs will still remain in order to ship for a deadline.

Developing software that is ‘perfect’ costs a lot in time and money and at some point software has to be released in a dynamic world/market. It can be difficult to replicate all the situations which may trigger bugs.

Q2 Test case design

The table below shows the discount rates for theatre tickets. The current price of an Adult ticket is £100. Children under age 6 are not allowed to get admission into the theatre.

Age	**Discount**
Adult (age 26-64)	No discount
Young person (age 6-25)	20% discount
Concession (age 65+)	10% discount

You need to test the software component which is used to calculate the price. This component takes the age as input, and outputs the price. Design Test Cases using **Partition Testing**.

Q2 work out…

Understand partition testing:

Input data and output results often fall into different partition where all members of a partition are related. Test cases should be chosen from each partition, typically boundary values and mid-range values.

Q2 model answer

Partition 1: Adult age 26-64

Input:

Lower Boundary value: 26

Upper Boundary value: 64

Middle range value: 45

Expected output: £100

Partition 2: Young person age 6-25

Input:

Lower Boundary value: 6

Upper Boundary value: 25

Middle range value: 17

Expected output: £80

Partition 3: Concession age 65+

Input:

Lower Boundary value: 65

Upper Boundary value: none

Middle range value: 80

Expected output: £90

Optional: Partition 4: invalid inputs

Input: <6

Expected output: error message e.g. must over age 6

Input: non integer/non digits/negative value

Expected output: error message e.g. invalid input

Test Driven Development

Q1

We use TDD approach to test the price calculation component. Fill in the missing code of this test program.

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class TicketTest {
        @Test
        public void testCalPrice(){
                Ticket t1 = new Ticket(100);
                assertEquals(________, t1.getPrice(26));
                assertEquals(________, t1.getPrice(64));
                assertEquals(________, t1.getPrice(45));
                assertEquals(80.0,  _______________);
                assertEquals(80.0,  _______________);
                assertEquals(80.0,  _______________);
                assertEquals(________, ___________);
                assertEquals(________, ___________);
        }
}
```

Q1 model answer

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class TicketTest {
        @Test
        public void testCalPrice(){
                Ticket t1 = new Ticket(100);
                assertEquals(__100.0___, t1.getPrice(26));
                assertEquals(__100.0___, t1.getPrice(64));
                assertEquals(__100.0___, t1.getPrice(45));
                assertEquals(80.0, t1.getPrice(6));
                assertEquals(80.0, t1.getPrice(25));
                assertEquals(80.0, t1.getPrice(17);
                assertEquals(__90.0___, t1.getPrice(65));
                assertEquals(__90.0___, t1.getPrice(80));
        }
}
```

Q2

Write the product program to pass the test.

Q2 model answer

```
public class Ticket {
        private double fullPrice;
        public Ticket(double fullPrice){
                this.fullPrice = fullPrice;
        }
        public double getPrice(int age){
                //code to validate age>=6 (throw exception)
                if (age>=6&&age<=25){
                        return fullPrice*0.8;
                }
                else if (age>=65){
                        return fullPrice*0.9;
                }
                else return fullPrice;
        }
```