3F6 Software Engineering and Design: 2015 Solutions

Dr. Richard E. Turner and Dr. Elena Punskaya

14 May 2015

1 (a) "Describe the purpose of *Interfaces* in *Object Oriented Design*. Explain why implementing multiple *Interfaces* by a single class might be useful.[15%]"

Bookwork

(b) (i) "Identify the principal *Classes* and their *Relationships* for the *Object Oriented* system that implements the `Stock View` functionality. Illustrate the design with the help of a *Class Diagram*. Consider the use of *Interfaces* to encapsulate different types of functionality provided by the *Class* `Market`. [40%]"

One of the possible solutions is presented in Fig. 1. Any reasonable naming conventions can be used.

(ii) "Draw a *Sequence Diagram* for the scenario in which the `Stock View` is updated in response to a transaction executed in the market. [30%]"

One of the possible solutions is presented in Fig. 2. Any reasonable naming conventions can be used.

(iii) "Discuss how *Agile Software Development* techniques can be used in the project described above. [15%]"

Most of the *Agile* techniques can be used, in particular, the following might be listed:

- using "stories" to capture the requirements and implement them incrementally
- test design and requirements via incremental deployments
- using source code repository and continuous integration (automated builds)
- test in the clone of the production environment
- automated testing, unit test, usability studies

It should also be noted that continuous deployment may not be appropriate given that any issue in the production system will have a high cost.

Figure 1: Class Diagram

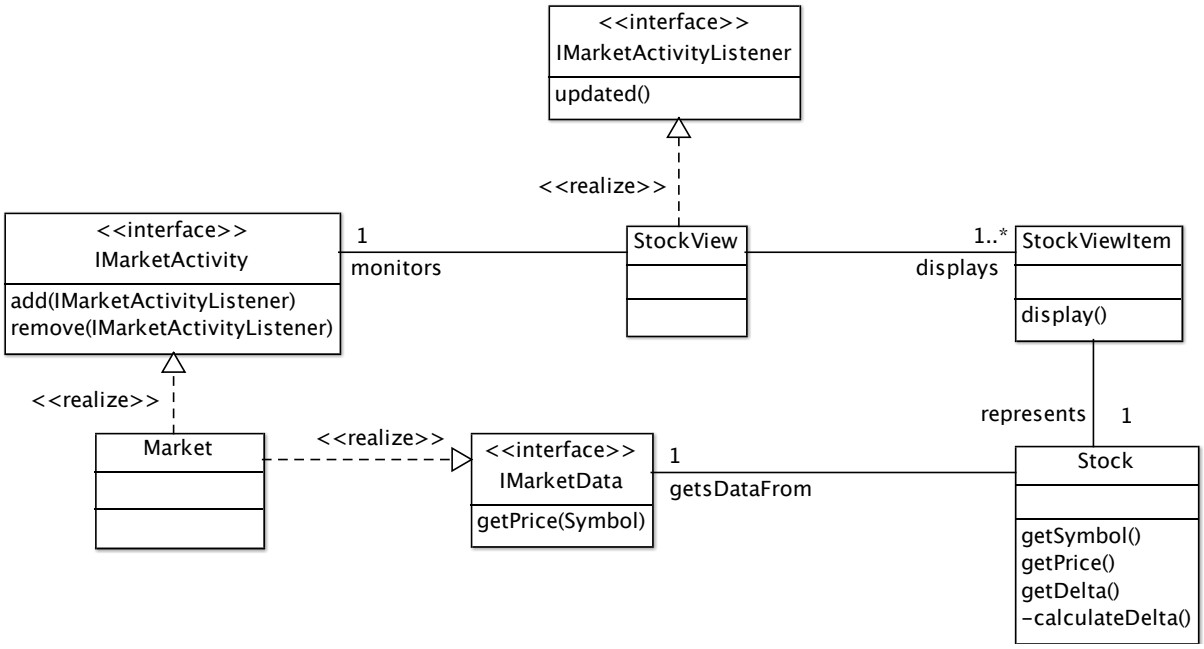
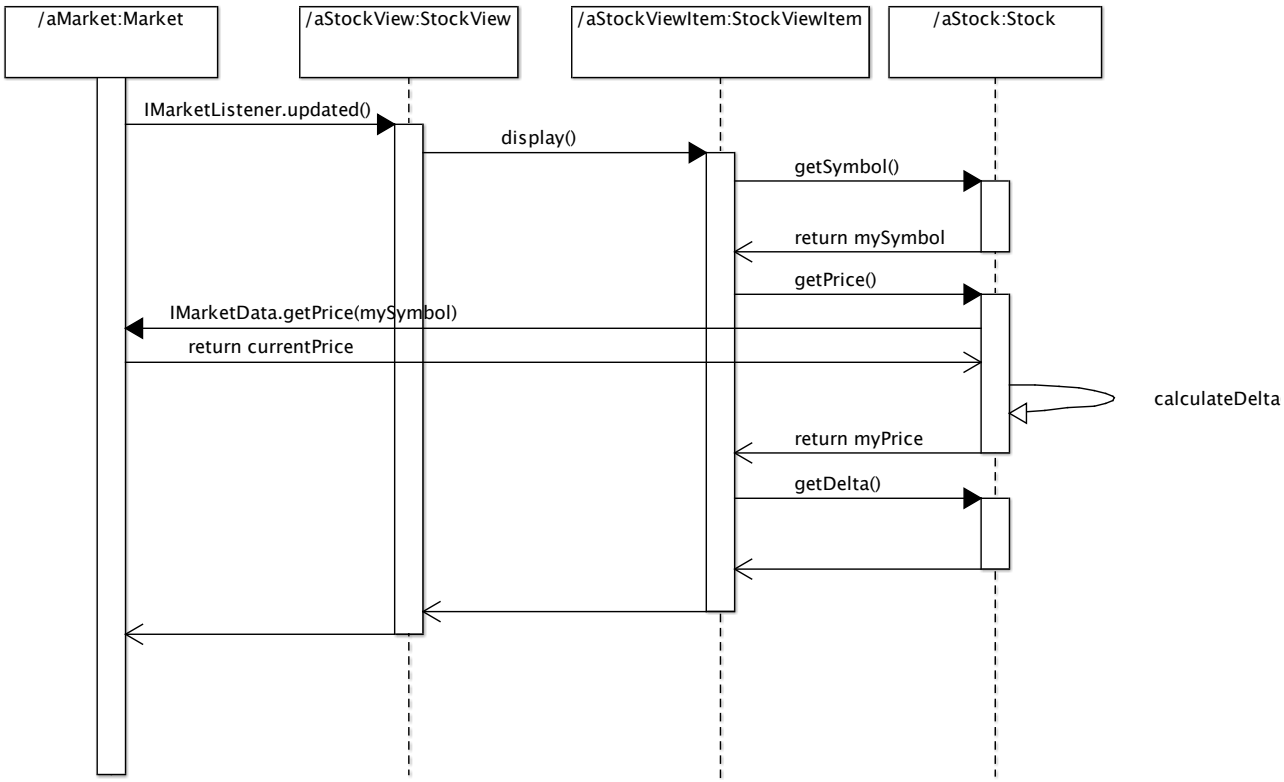


Figure 2: Sequence Diagram

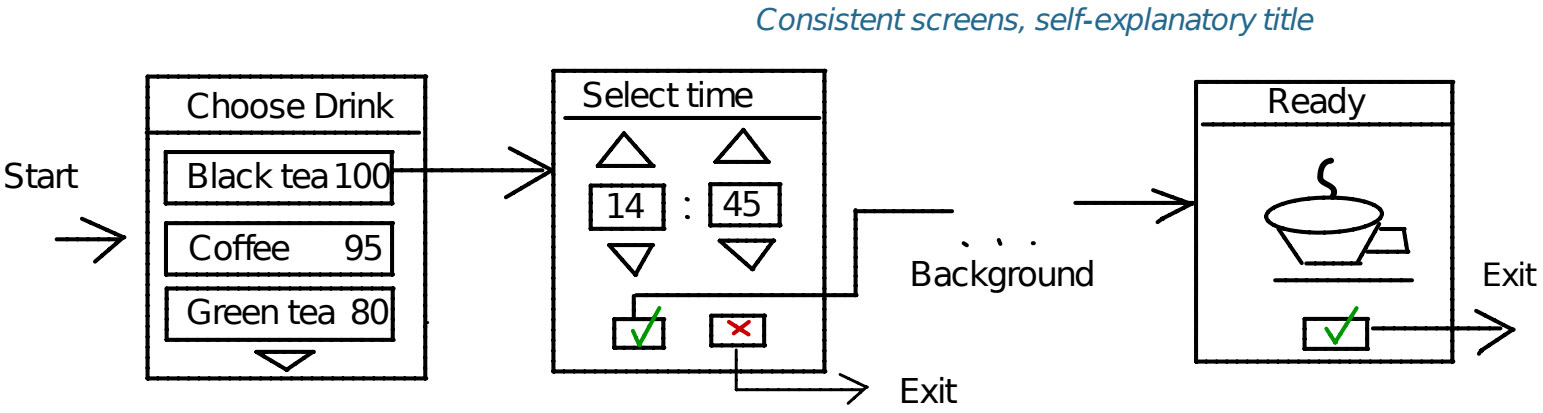


Question 2

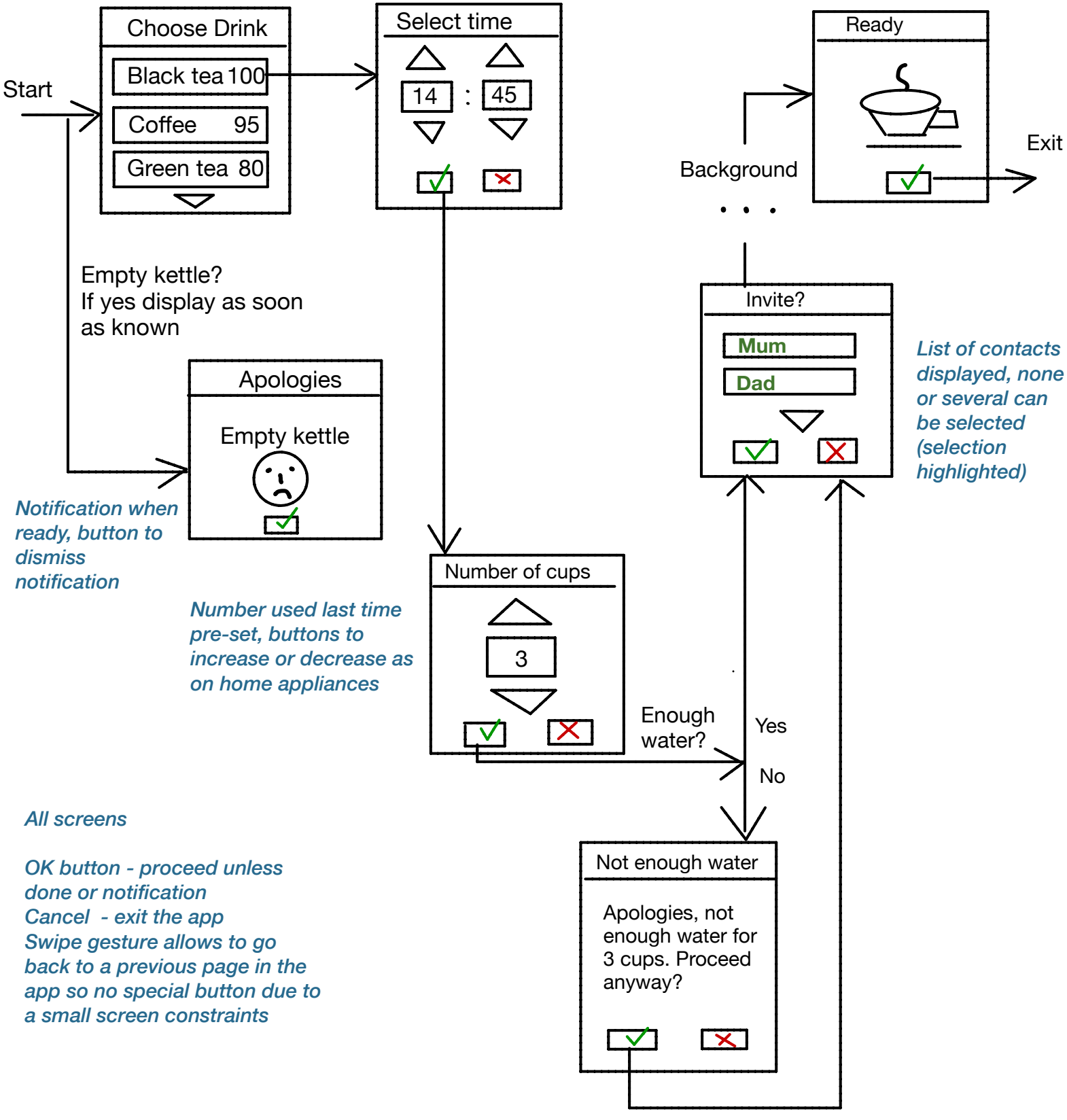
a) Bookwork

b)

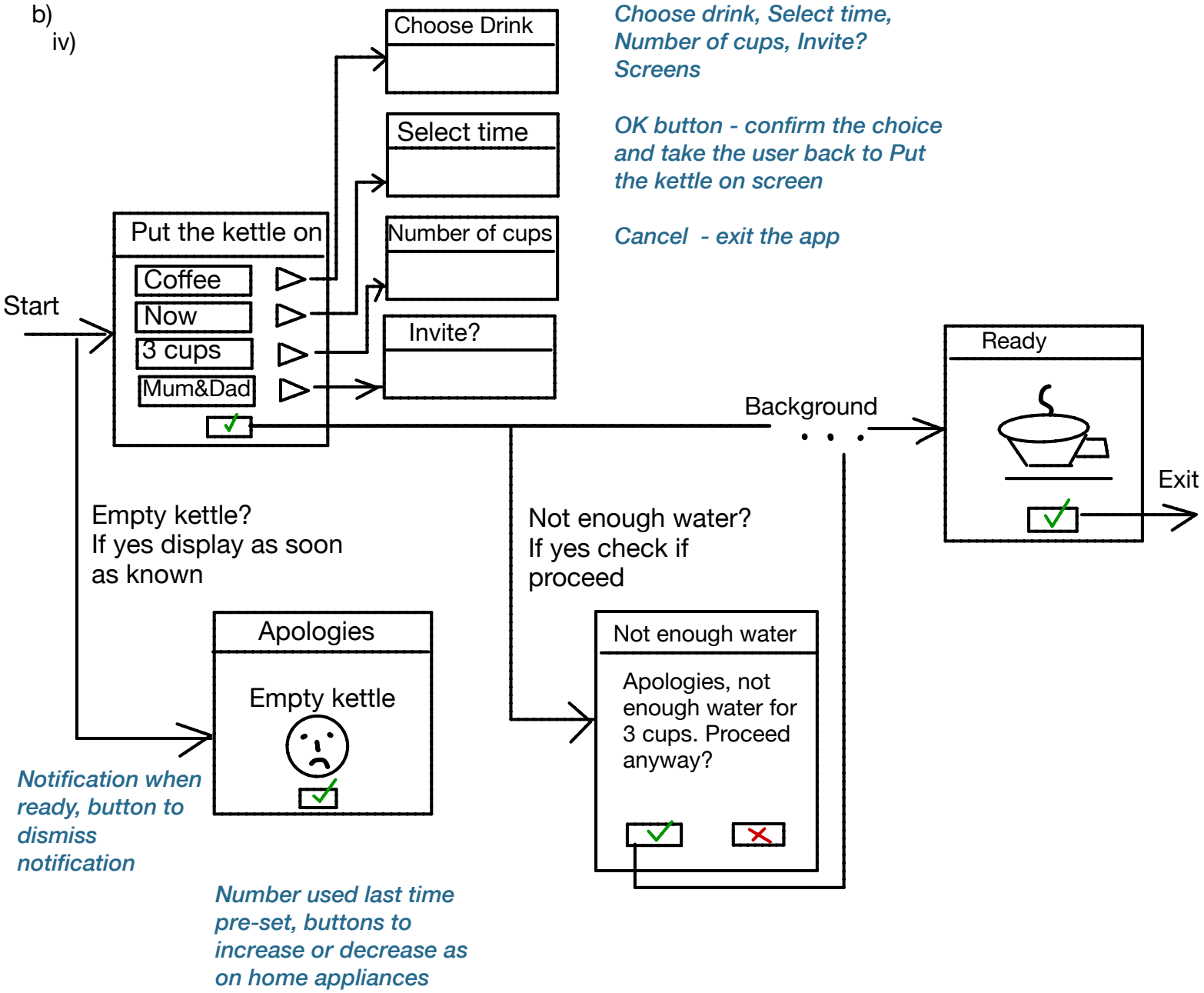
i) One of the possible solutions is presented below



b)
ii-iii)


Start
Choose Drink
Black tea 100
Coffee 95
Green tea 80
Select time
14 : 45
Ready
Exit
Background
Empty kettle?
If yes display as soon as known
Apologies
Empty kettle
Notification when ready, button to dismiss notification
Invite?
Mum
Dad
List of contacts displayed, none or several can be selected (selection highlighted)
Number of cups
3
Number used last time pre-set, buttons to increase or decrease as on home appliances
Enough water?
Yes
No
Not enough water
Apologies, not enough water for 3 cups. Proceed anyway?
All screens
OK button - proceed unless done or notification
Cancel - exit the app
Swipe gesture allows to go back to a previous page in the app so no special button due to a small screen constraints

b)
iv)



Choose drink, Select time, Number of cups, Invite? Screens

OK button - confirm the choice and take the user back to Put the kettle on screen

Cancel - exit the app

Notification when ready, button to dismiss notification

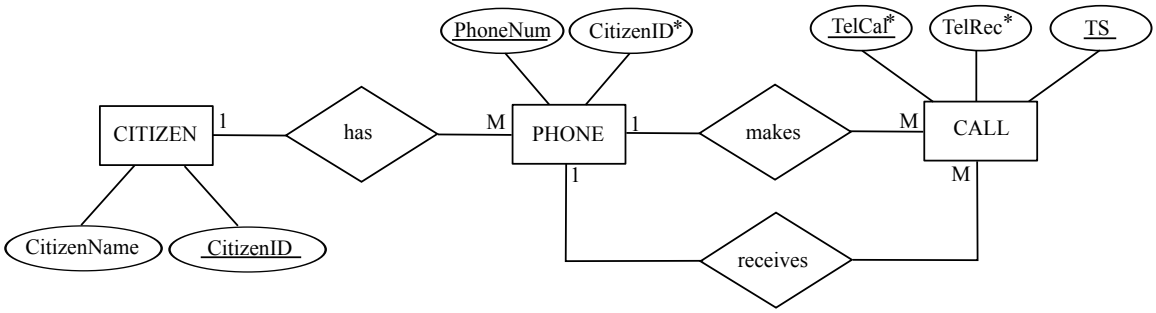
Number used last time pre-set, buttons to increase or decrease as on home appliances

All screens

OK button - proceed unless done or notification
Cancel - exit the app

Swipe gesture allows to go back to a previous page in the app so no special button due to a small screen constraints

3 (a) "Evaluate the database design and suggest how to improve it. Draw an *Entity-Relationship Diagram* to illustrate your answer. [15%]"



(b) "Show the updated design of the tables including any new Entities and Attributes added in the answer to part (a). Identify the Primary and Foreign keys used. [15%]"

The new tables are:
CITIZEN [CitizenID,Name]
primary key: CitizenID
PHONE [CitizenID,PhoneNum]
primary key: PhoneNum foreign key: CitizenID
CALL [TelCal,TelRec,TS]
primary key: (TelCal,TS) or (TelRec,TS) foreign key: TelCal and TelRec (both reference PhoneNum)

(c) "Design a query to return all of the telephone numbers that Alice has made calls from. Express your answer using relational algebra or SQL code and explain your solution. [10%]"

SELECT PHONE.PhoneNum FROM CITIZEN JOIN PHONE ON CITIZEN.CitizenID = PHONE.CitizenID WHERE CITIZEN.Name = 'Alice';

(d) "Design a query to return the names of the citizens who have called Alice. Express your answer using relational algebra or SQL code and explain your solution. [25%]"

One method uses the solution to part (c)

SELECT DISTINCT CITIZEN.Name FROM CITIZEN JOIN PHONE ON CITIZEN.CitizenID = PHONE.CitizenID JOIN CALL ON PHONE.PhoneNum = CALL.TelCal WHERE CALL.TelRec IN (SELECT PHONE.PhoneNum FROM CITIZEN JOIN PHONE ON CITIZEN.CitizenID = PHONE.CitizenID WHERE CITIZEN.Name = 'Alice');

Alternatively, using the rename operator:

SELECT DISTINCT CITIZEN.Name FROM PHONE JOIN CALL ON PHONE.PhoneNum = CALL.TelCal JOIN PHONE AS X ON X.PhoneNum = CALL.TelRec JOIN CITIZEN ON CITIZEN.CitizenID = PHONE.CitizenID JOIN CITIZEN AS Y ON Y.CitizenID = X.CitizenID WHERE Y.Name = 'Alice';

(e) "The government wants to know the names of the citizens who were called by citizens who were themselves called by Alice. Design a suitable query and express your answer using relational algebra or SQL code. Explain your solution. [25%]"

SELECT DISTINCT Y.Name FROM PHONE JOIN CALL AS C1 ON PHONE.PhoneNum = C1.TelCal JOIN CALL AS C2 ON C1.TelRec = C2.TelCal JOIN PHONE AS X ON X.PhoneNum = C2.TelRec JOIN CITIZEN ON CITIZEN.CitizenID = PHONE.CitizenID JOIN CITIZEN AS Y ON Y.CitizenID = X.CitizenID WHERE CITIZEN.Name = 'Alice';

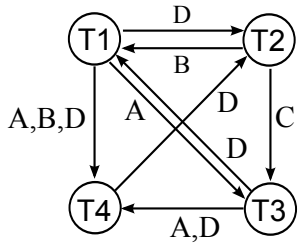
(f) "The government wants to carry out many queries similar to the one in part (e). Describe features that could be added to the database to accelerate such queries and detail potential disadvantages they may have. [10%]"

Secondary indices could be added to the database. Three secondary indices could be added to TelRec and TelCal in CALL, and also PhoneNum in PHONE. It might also be useful to add a secondary index to Name in CITIZEN. Adding these indices will accelerate queries, but they will slow down additions/modifications/deletions. This might be especially problematic for the CALL relation as new calls will continually be added to it.

4 (a) "Explain the importance of the *serialisability* of a schedule in concurrency control. Define the *serialisation graph* and explain how it is used to determine whether a schedule is serialisable. [20%]"

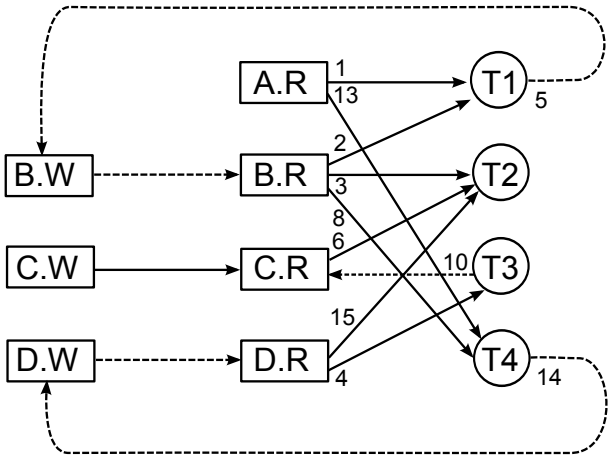
Bookwork

(b) i. "Draw the serialisation graph for the schedule and determine whether it is serialisable. For this part of the question assume that no form of concurrency control is used. [15%]"

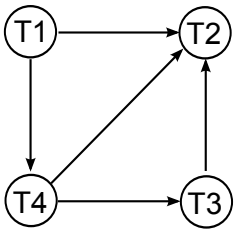


There are directed loops in the serialisation graph, and so the schedule is not serialisable.

ii. "Draw a *resource allocation graph* for this sequence of transactions immediately after action number 15. [30%]"



iii. "Draw the corresponding *wait-for-graph*. State which transactions will complete and the order in which they do so. Comment on the efficacy of the concurrency control protocol for this schedule.[20%]"



T2 commits, followed by T3 (this requires inspection of the intended actions for T3 after time-step 10 to ensure deadlock does not occur), then T4 commits and finally T1. The locking protocol has turned a non-serialisable intended schedule into a serialisable schedule.

(c) "What are the limitations of lock-based concurrency control methods? Outline an alternative concurrency control protocol and explain in which situations it is likely to outperform lock-based methods. [15%]"

Bookwork

Principal Assessor's comments:

Q1 Universal modelling language, design patterns, and interfaces. The least popular question in the exam, but nevertheless well answered by the majority of candidates. A number of candidates did not identify the observer design pattern as the most logical solution to part (b). Good solutions used a design that could enable separate software teams to develop the component parts in parallel without knowing the specifics of the other teams implementation.

Q2 User interfaces. A popular question that was well answered by most. Many candidates were able to identify and devise a sensible design for the smart watch screens, but some failed to discuss the relationship between the screens and program flow.

Q3 Database design and queries. This question was well answered in the main. Some candidates struggled to devise sensible entity relationship diagrams and this led to overly complex databases for which it was cumbersome to design queries. Many candidates failed to identify secondary indices as the most sensible solution to part f.

Q4 Concurrency control. This question polarised candidates with many doing very well (there were two essentially perfect solutions) and many doing poorly. The candidates who struggled a) did not know what a serialisation graph was, and b) failed to identify when transactions would wait (and therefore stop acquiring locks) when constructing the resource allocation diagram.