Tutorial 9

1) Refering to Tutorial 4 question 3, we must specify what should happen if it is not possible to perform `normal registration and logged-in (dealing with errors). We define the free data type called`:

RESPONSE ::= success | userNotExist | userAlreadyExist | userNotLoggedIn | userAlreadyLoggedIn | fullCapacity

and the schema

OkMessage == [rep!: RESPONSE | rep! = success]

ReegisterNewUser

$\Delta ComputerSystem$

$user? : PERSON$

$user? \notin users$

$user? \notin loggedIn$

$users' = users \cup \{user?\}$

$loggedIn' = loggedIn$

LoggedIn

$\Delta ComputerSystem$

$user? : PERSON$

$user? \in users$

$user? \notin loggedIn$

$\#loggedIn < capacity$

$user' = users$

$loggedIn' = loggedIn \cup \{user?\}$

Schema Name	Success Pre-condition	Failure Pre-condition	Remark
RegisterNewUser	$user? \notin users$	$user? \in users$	User already exist

LoggedIn	$user? \in users$ $user? \notin loggedIn$ $\#loggedIn < capacity$	$user? \notin users$ $user? \in loggedIn$ $\#loggedIn \geq capacity$	User does not exist User already logged in Full capacity

2) Message for each error > use free type definition

RESPONSE ::= success | userNotExist | userAlreadyExist | userNotLoggedIn | userAlreadyLoggedIn | fullCapacity

3) Success schema using message listed

SuccessSchema

$rep! : RESPONSE$

$rep! = success$

OR Text Form

SuccessSchema == [rep! : RESPONSE | rep! = success]

4) Error scenario schema

RegisterNewUserError

$\Xi ComputerSystem$
$user? : PERSON$

$user? \in users$

(a) Write a schema called *RegisterNewUserError* that will provide response *rep!* if it is not possible to register as a new user. Complete the registration operation as *RegistrationComplete*.

RegisterNewUserError

$\Xi ComputerSystem$
$user? : PERSON$

$user? \in users$
$rep! = userAlreadyExist$

RegistrationComplete $\triangleq$ (RegisterNewUser $\wedge$ SuccessMessage) $\vee$ RegisterNewUserError

(b) Write a schema called *LoggedInError* that will provide response *rep!* if it is not possible to logged-in into the system. Complete the logged-in operation as *LoggedInComplete.*

UserNotExist

$\Xi ComputerSystem$
$user? : PERSON$
$rep! = RESPONSE$

$user? \notin users$
$rep! = userNotExist$

AlreadyLoggedIn

$\Xi ComputerSystem$
$user? : PERSON$
$rep! = RESPONSE$

$user? \in loggedIn$
$rep! = userAlreadyLoggedIn$

FullCapacity

$\Xi ComputerSystem$
$rep! = RESPONSE$

$\#loggedIn \geqslant capacity$
$rep! = fullCapacity$

LoggedInComplete $\triangleq$ *(LoggedIn* $\wedge$ SuccessMessage) $\vee$ UserNotexist $\vee$ AlreadyLoggedIn $\vee$ FullCapacity

OR

LoggedInError

$$\Xi ComputerSystem$$
$$user? : PERSON$$
$$rep! = RESPONSE$$

$$(user? \notin users \land rep! = userNotExist)$$
$$\lor$$
$$(user? \in loggedIn \land rep! = userAlreadyLoggedIn)$$
$$\lor$$
$$(\#loggedIn \geqslant capacity \land rep! = fullCapacity)$$

LoggedInComplete $\triangleq$ *(LoggedIn* $\land$ SuccessMessage) $\lor$ LoggedInError

2) Let [ROOM] be the set of all possible hotel rooms. The hotel has a set of rooms named *accommodation* and guests may occupy some (or all) of these rooms. A specification for a hotel has a state space schema:

Hotel

$$accomodation, occupied : \mathbb{P}\ ROOM$$

$$occupied \subseteq accomodation$$

(a) Referring to the specification below, interpret and rewrite the specification for *RoomOccupy* in Z schema.

Use case:	RoomOccupy
Purpose:	To inform the system that a room is now occupied by a hotel guest.
Pre-conditions:	The room exists in the system and currently not occupied
Initiating actor:	Receptionist
Main success Scenario	1) Receptionist inputs room 2) System confirms room now is occupied 3) The accommodation in the hotel will not change 4) Exit success
Exceptions	1) Room is not in the system 1a Exit failure 2) Room is already occupied 2a Exit failure

RoomOccupy

$\Delta Hotel$

$r? : ROOM$

$r? \in accomodation$

$r? \notin occupied$

$occupied' = occupied \cup \{r?\}$

$accomodation' = accomodation$

Schema Name	Success Pre-condition	Failure pre-condition	Remark for failure
RoomOccupy	$r? \in accomodation$ $r? \notin occupied$	$r? \notin accomodation$ $r? \in occupied$	Room does not exist Room is already occupied

(b) Introduce a free type for an exception handling message called *REPORT* that has five errors handling. The error handling messages are *ok*, *alreadyAdded*, *notInSystem*, *roomOccupied* and *roomVacant*.

REPORT ::= ok | alreadyAdded | notInSystem | roomOccupied | roomVacant

(c) Construct a schema *RoomOccupiedError* where we cannot put a guest in a room, or remove the room from the system, if it is currently occupied.

RoomOccupiedError

$\Xi Hotel$

$r? : ROOM$

$report! : REPORT$

$(r? \notin accomodation \land report! = notInSystem)$

$\lor$

$(r? \in occupied \land report! = roomOccupied)$

RoomOccupyComplete = (RoomOccupy $\lor$ SuccessSchema) $\lor$ RoomOccupiedError