Homework7-CS3365-Spring2018

Using multi-threaded programming, you are required to implement a hotel system, which provides a check-out with cash payment use case depicted in Fig. 1 and Fig. 2. Fig. 1 depicts the communication diagram for the check-out with cash payment use case and Fig. 2 depicts the simplified software architecture that consists of the Hotel Client and Hotel Server subsystems. In the software architecture (Fig. 2), the Desk Clerk Interface object and Check-out Manager object are active objects (i.e., threads) and they communicates messages using synchronous message communication, which is implemented by means of a message buffer and response connector class (Fig. 3 and Fig. 4). Fig. 5 depicts the specification of Check-out Manager object. Fig. 6 and Fig. 7 depict the specifications of classes in Fig. 2.

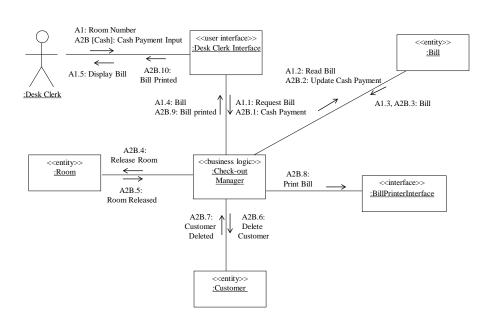


Fig. 1 Communication Diagram for Check-out Customer use case – Check-out with Cash Payment

<<subsystem>> :HotelClient checkOut(in Room#, out Bil payByCash(in Amount, out Bil Printed <<user interface>> <<entity>> :Desk Clerk Interface :Bill readBill(in Room#, out Bill updateCashPayment(in Room#, in Amount, out Bill) :Desk Clerk **^**--requestBill(in Room#, out Bill)
payByCash(in Amount, out BillPrinted <<business logic>> printBill(in Bill) <<entity>>
<u>:Room</u> :Check-out <<interface>> Manager :BillPrinterInterface releaseRoom(in Room#, out RoomReleased) deleteCustomer(in Room#, out CustomerDeleted) <<entity>>
:Customer <<subsystem>> :HotelServer

Fig. 2 Simplified Software Architecture for Hotel System - Check-out with Cash Payment

Fig. 3 message buffer and response connector

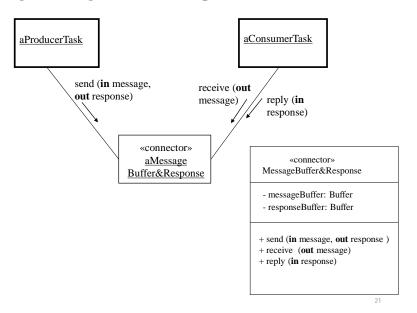


Fig. 4 Message buffer & response connector

```
monitor MessageBuffer&Response
 -- Encapsulates a message buffer that holds at most one message
 -- and a response buffer that holds at most one response.
 -- Monitor operations are executed mutually exclusively.
private messageBufferFull : Boolean = false;
private responseBufferFull : Boolean = false;
public send (in message, out response)
 place message in buffer;
  messageBufferFull := true;
  notify;
  while responseBufferFull = false do wait;
  remove response from response buffer;
  responseBufferFull := false;
end send;
public receive (out message)
  while messageBufferFull = false do wait;
  remove message from buffer;
  messageBufferFull := false;
end receive;
public reply (in response)
  Place response in response buffer;
  responseBufferFull := true;
     notify;
end reply;
end MessageBuffer&Response;
```

22

Fig. 5 Specification of Check out Manager active object

```
loop
receive Message from Hotel Message Buffer and Response;
Extract message name and message parameters from message;
case Message of
    requestBill:
        requestBill (in Room#, out Bill);
        reply (Bill);
PayByCash:
        payByCash (in Amount, out BillPrinted)
        reply (BillPrinted);
end case;
end loop;
```

Fig. 6 Class specification

<user interface>>
Desk Clerk Interface

checkOut(in Room#, out Bill)

payByCash(in Amount, out BillPrinted)

<interface>>
BillPrinterInterface

printBill(in Bill)

Fig. 7 Class specification



RoomNo	Status
301	Occupied
302	Available
303	Occupied

- RoomNo	
- Name	
- PhoneNo	



RoomNo	Bill	Total	ReferenceNo
301	John's Bill	100	
303	Sam's Bill	200	

RoomNo	Name	PhoneNo
301	John	806-333-1234
303	Sam	806-333-4567