University of
South Australia

# INFS 2044

## Workshop 3b

# Preparation

- Read the required readings
- Watch the Week 3 Lecture
- Bring a copy of the workshop instructions (this document) to the workshop

# Where We Are At

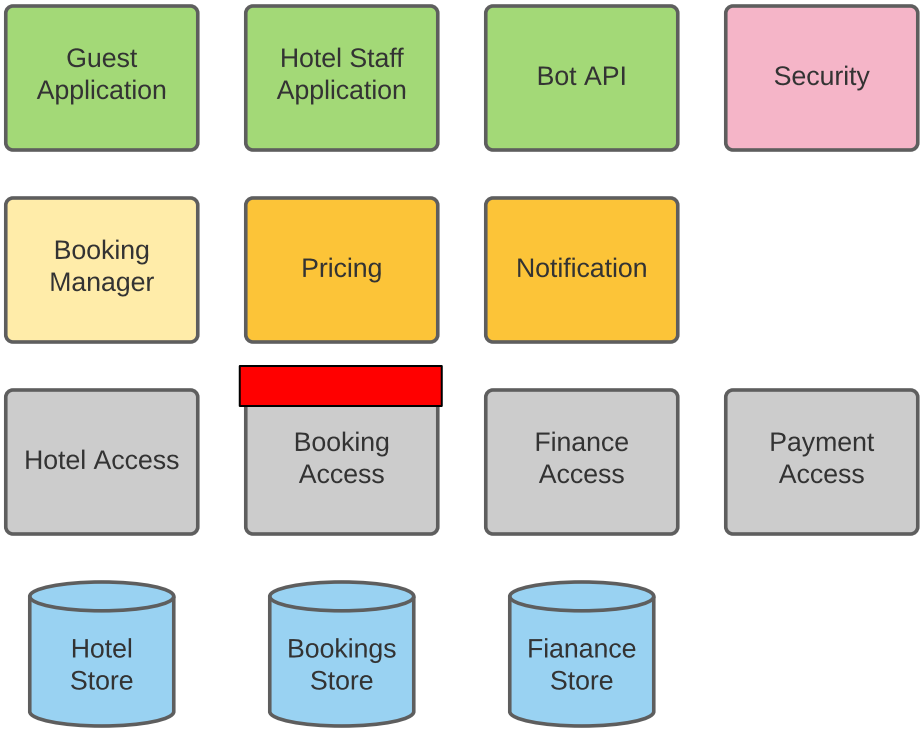
- Designed system-level operations
- Drew System Sequence Diagrams

# Learning Objectives

- Design component interfaces
- Draw Sequence Diagrams to show interactions
- Correctly orient dependencies between components

# Task 1. Assess Interface

- Assess the interface for the *BookingAccess* component shown on the following slides.
- Does the interface hide implementation details?
- Are the operations at an appropriate level of abstraction?


Guest Application
Hotel Staff Application
Bot API
Security
Booking Manager
Pricing
Notification
Hotel Access
Booking Access
Finance Access
Payment Access
Hotel Store
Bookings Store
Fianance Store

# BookingAccess Interface #1

- bookings(): list(str) # returns a list of booking IDs
- findBookings(filter:str): list(str) # filter is an SQL query
- createBooking(): str
- setBookingRoomID(bookingID:str, roomID: str)
- setBookingCheckIn(bookingID:str, dt:Date)
- setBookingCheckOut(bookingID:str, dt:Date)
- setBookingGuest(bookingID:str, guestID: str)
- cancelBooking(bookingID:str)

# Task 2. Interaction Diagram

- Draw a Sequence Diagram for the interaction between two classes/components defined in the Python code fragments shown on the subsequent slides.

# Booking Manager in Python

```
class BookingManager:
  def findRooms(...): ...
  def calculateDays(…):…

  def getBasePrice(roomID, inDate, outDate):
    roomInfo = self.hotelAccess.getRoomDetails(roomID)
    days = self.calculateDays(inDate,outDate)
    return roomInfo.dailyRate * days

  def createBooking(roomID,...):
    totalPrice = self.pricingPolicy.getTotalPrice(self, roomID, inDate, outDate)
    ...
    bookingID = self.bookingAccess.createBooking(roomID,...)
    return bookingID
```

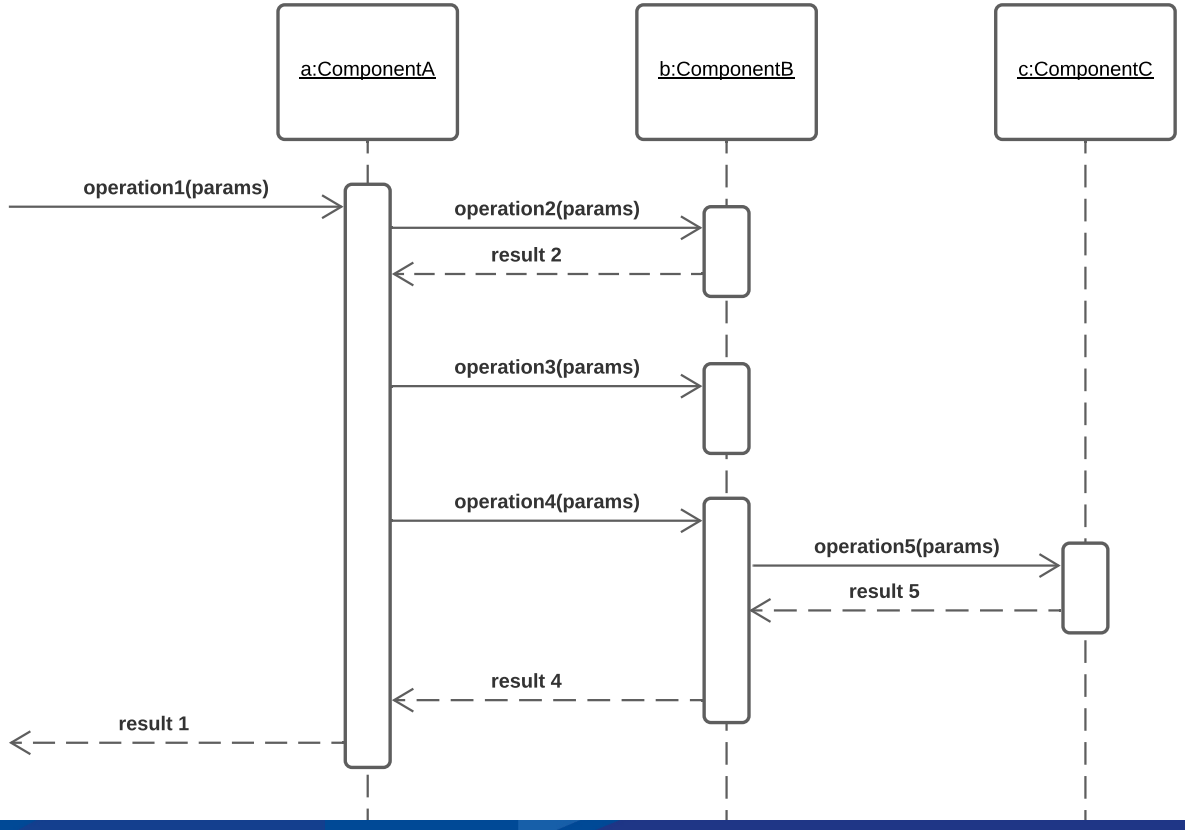
# Pricing Policy in Python

```
class PercentDiscountPricingPolicy:

  def __init__(percentDiscount):
    self.percentDiscount = percentDiscount

  def getTotalPrice(manager, roomID, inDate, outDate):
    basePrice = manager.getBasePrice(roomID, inDate, outDate)
    discountedPrice = basePrice * self.percentDiscount
    return discountedPrice
```

# Sequence Diagram Example



# Task 3. Assess Boundaries

- Identify any issues that may be present in the interaction design defined in Task 2.
- Revise the design to create a better interface for the two components

# Task 4. Assess Abstractions

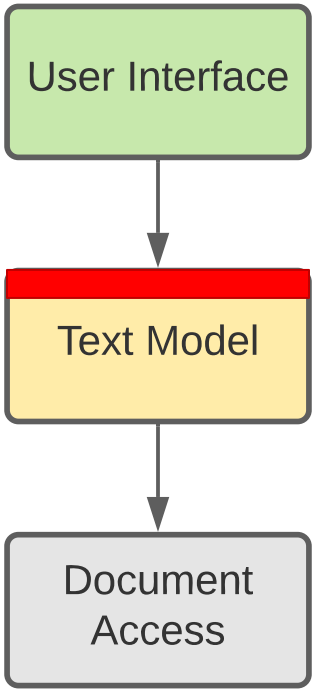
- Consider the design of a simple text editor shown on the following slides

- Discuss advantages and disadvantages of the given interface design.
- Would the component be easily reusable?
- Is the interface at the right level of abstraction?
- Is the module deep or shallow?

# Text Editor Use Cases

- Load, edit, save plain text documents
- Search & replace text in the current document

# Text Editor Architecture



# Text Editor Model Interface

- backspace(cursor:Cursor)
- delete(cursor:Cursor)
- deleteSelection(selection:Selection)

- Cursor represents the position of the cursor
- Selection represents the range of text that is selected

# You Should Know

- Detect information leakage
- Assess the quality of abstraction in an interface
- Define interfaces at an appropriate level of abstraction
- Correctly orient dependencies between components

# Activities this Week

- Complete Quiz 3

University of
South Australia