University of
South Australia

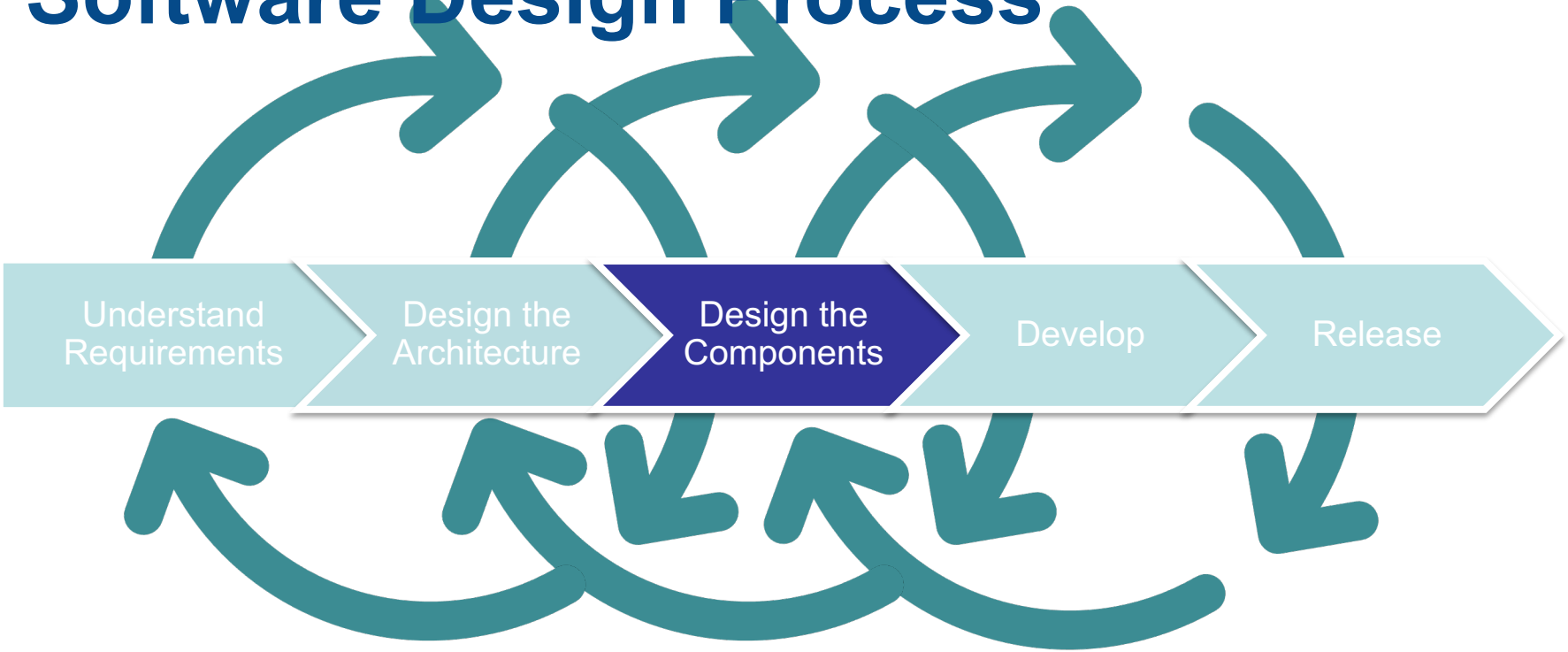
INFS 2044

Workshop 4a Answers

Preparation

- Read the required readings
- Watch the Week 4 Lecture
- Bring a copy of the workshop instructions (this document) to the workshop
- Bring a copy of the Find a Meeting Script to the workshop

Software Design Process



Where We Are At

- Designed system-level and component interfaces
- Drew Sequence Diagrams

Learning Objectives

- Understand object interactions
- Document implementation design using UML diagrams
- Determine feasibility of implementation designs

Task 1. Play Out an Interaction

- Conduct this activity in groups of 6 students.
- Play out the interactions defined in the *Find a Meeting Script* document on the course site.
- Allocate each of the 6 objects to a different student.

Find a Meeting Scenario

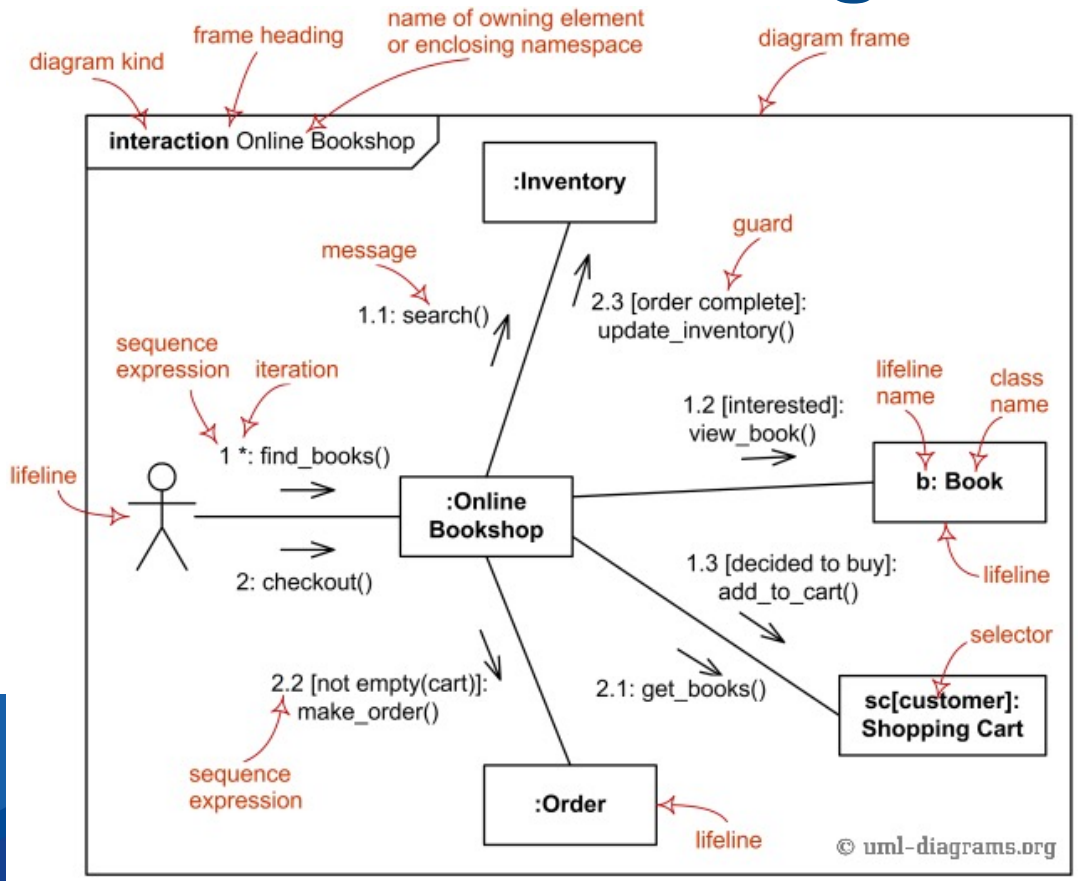
- There are the following objects:
  - 1 Controller
  - 1 Calendar
  - 2 Meetings
  - 2 Contact
- The calendar collects all meetings, which are scheduled at a day/time and include contacts.
- The objective is to find when the next meeting is scheduled that includes a given contact

Task 2. Draw Communication Diagram

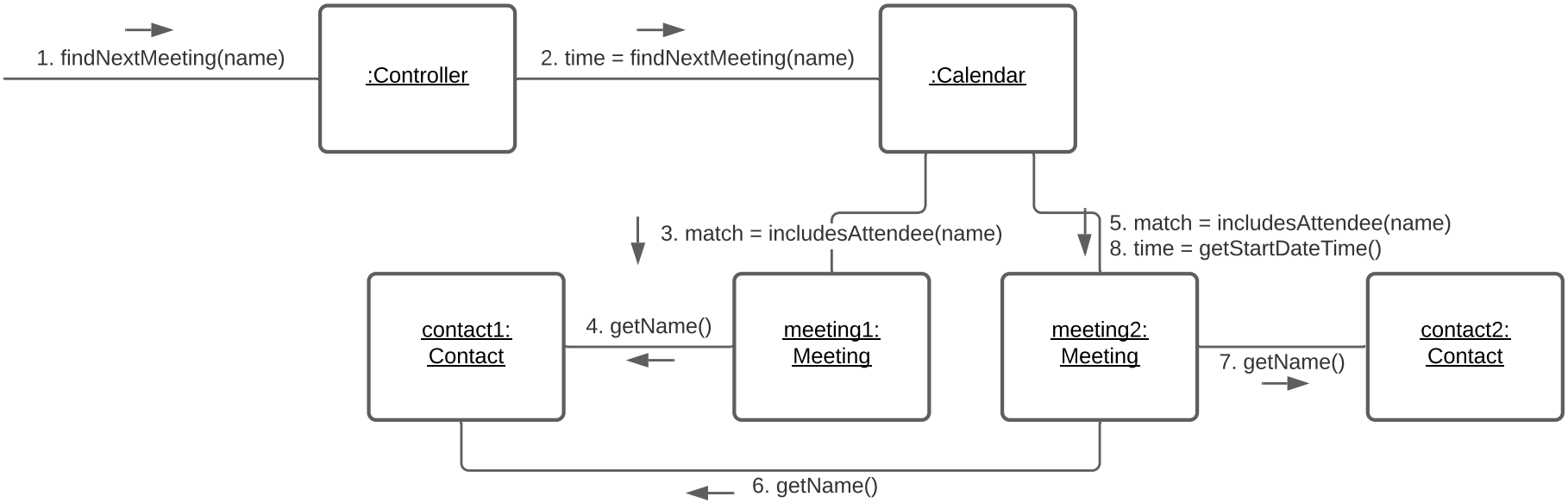
- The UML defines two diagrams for showing interactions
  - Sequence Diagram (see previous workshop)
  - Communication Diagram

- Draw a *UML Communication Diagram* for the interaction in Task 1

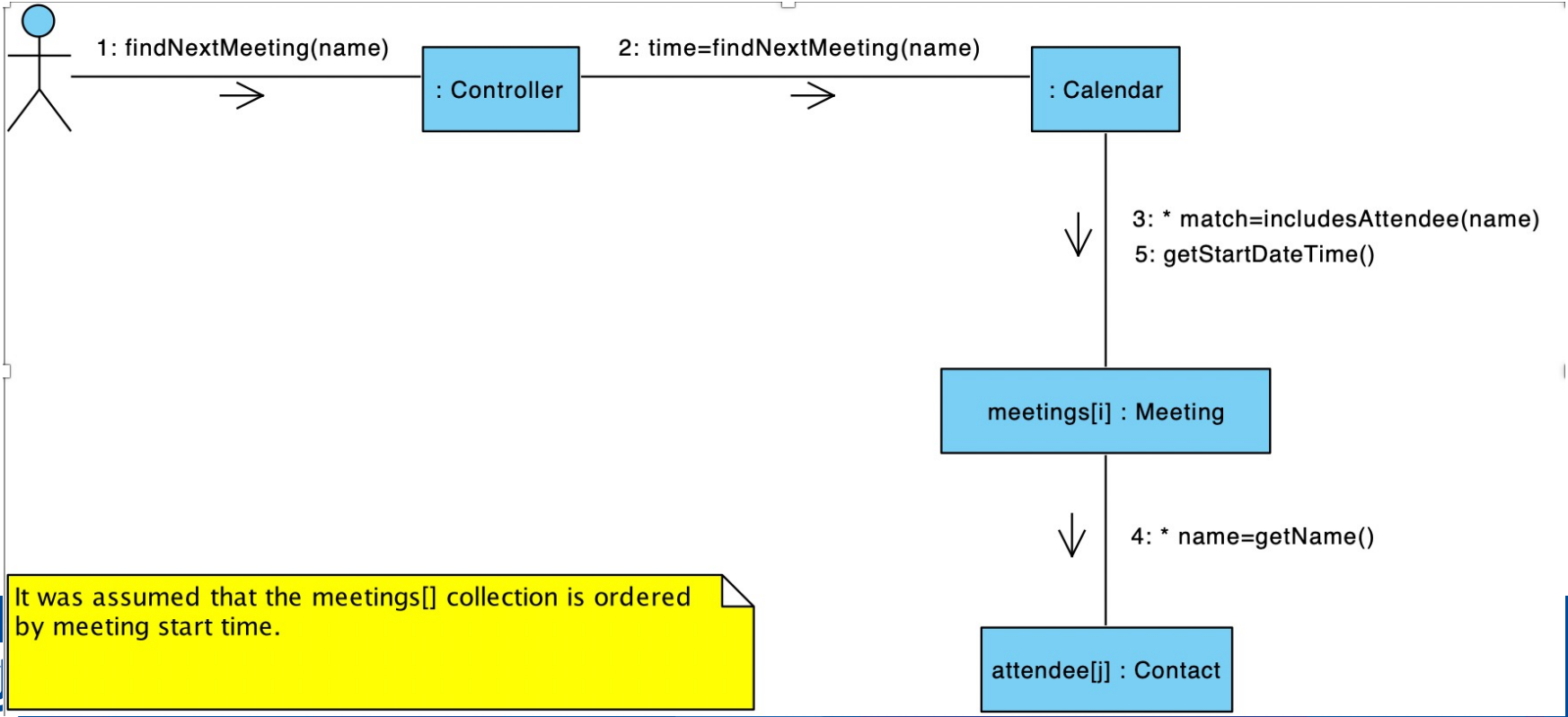
UML Communication Diagram Syntax



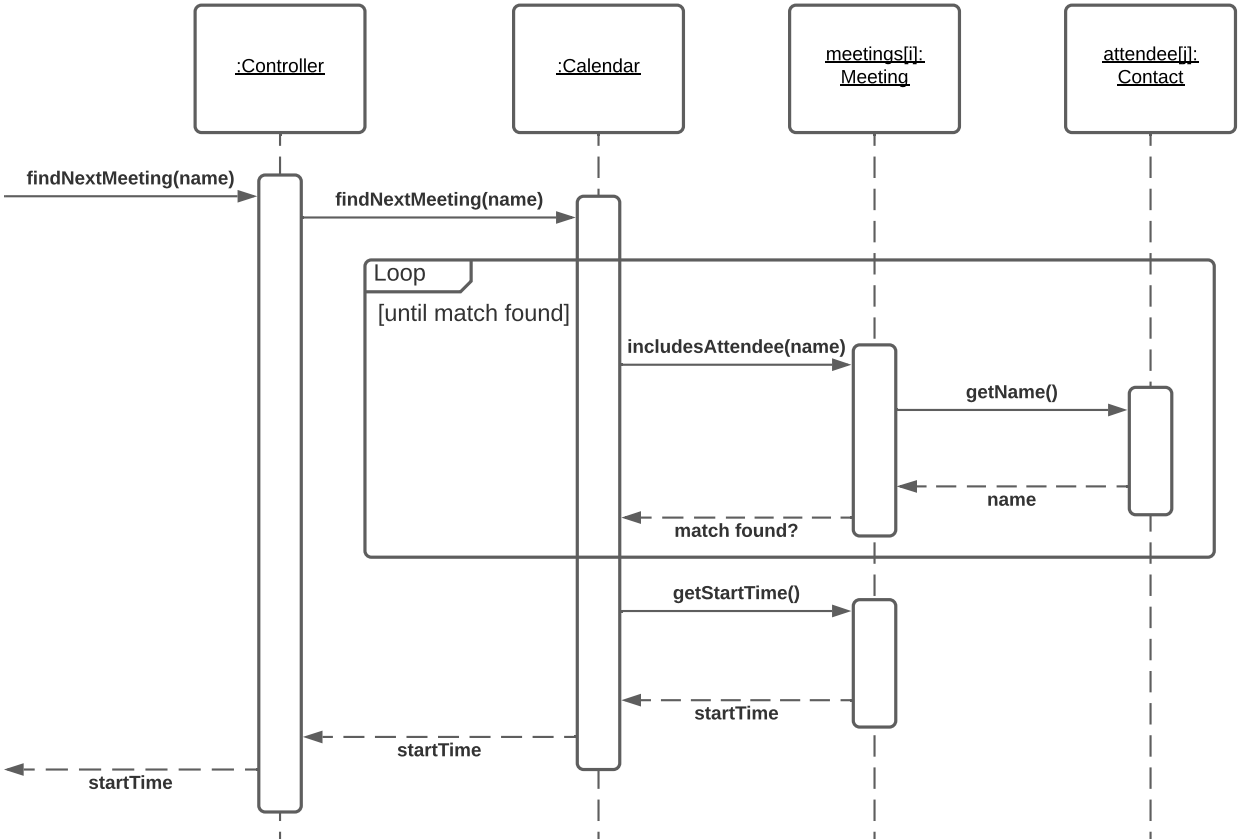
Find a Meeting Comm. Diagram v1



Find a Meeting Comm. Diagram v2



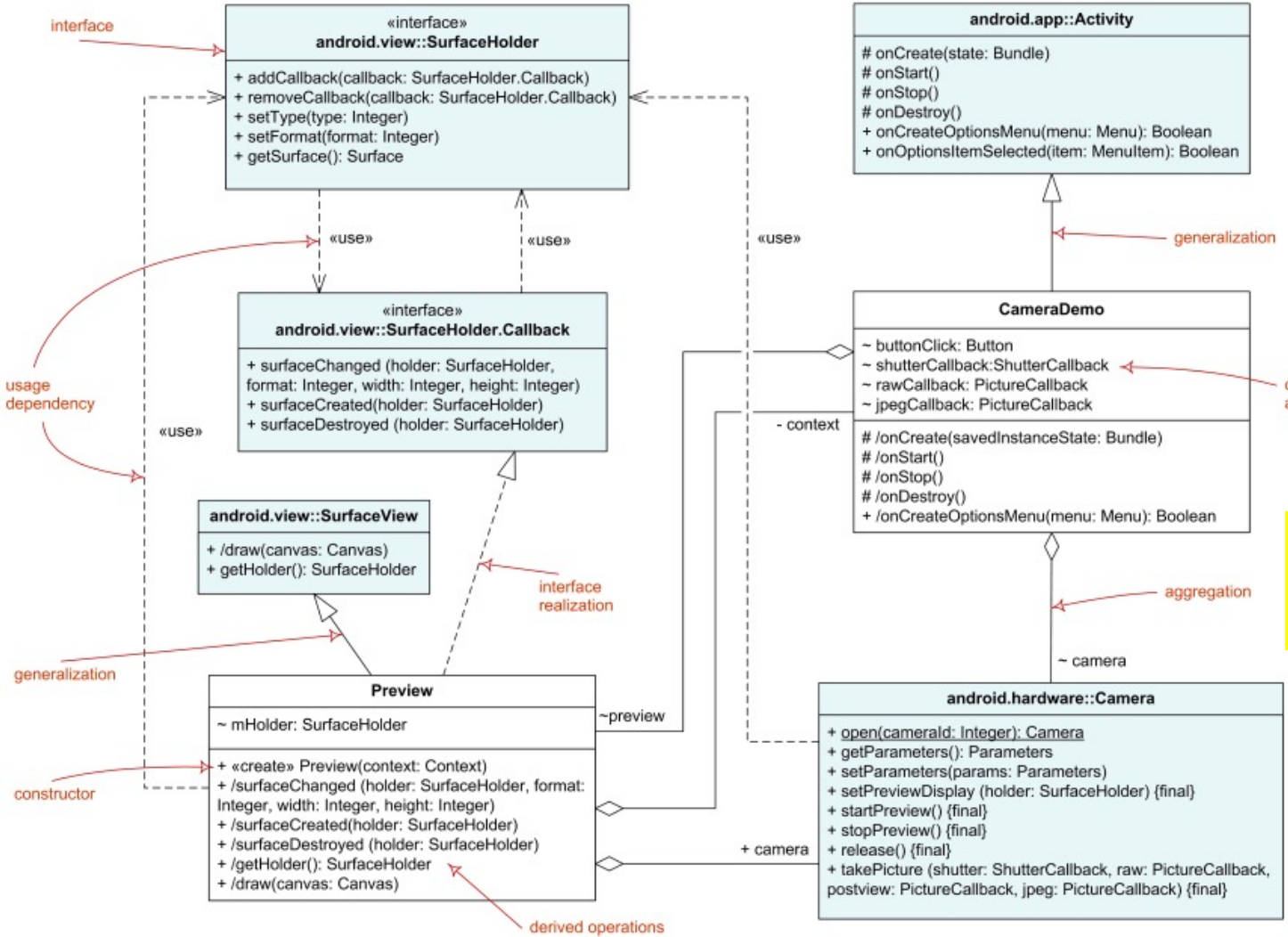
Find a Meeting Sequence Diagram



University of South Australia

Task 3. UML Class Diagram

- Draw a UML Class Diagram showing the classes participating in the interaction defined in Task 1
- Ensure that the diagram is consistent with the interaction diagram drawn in Tasks 2.

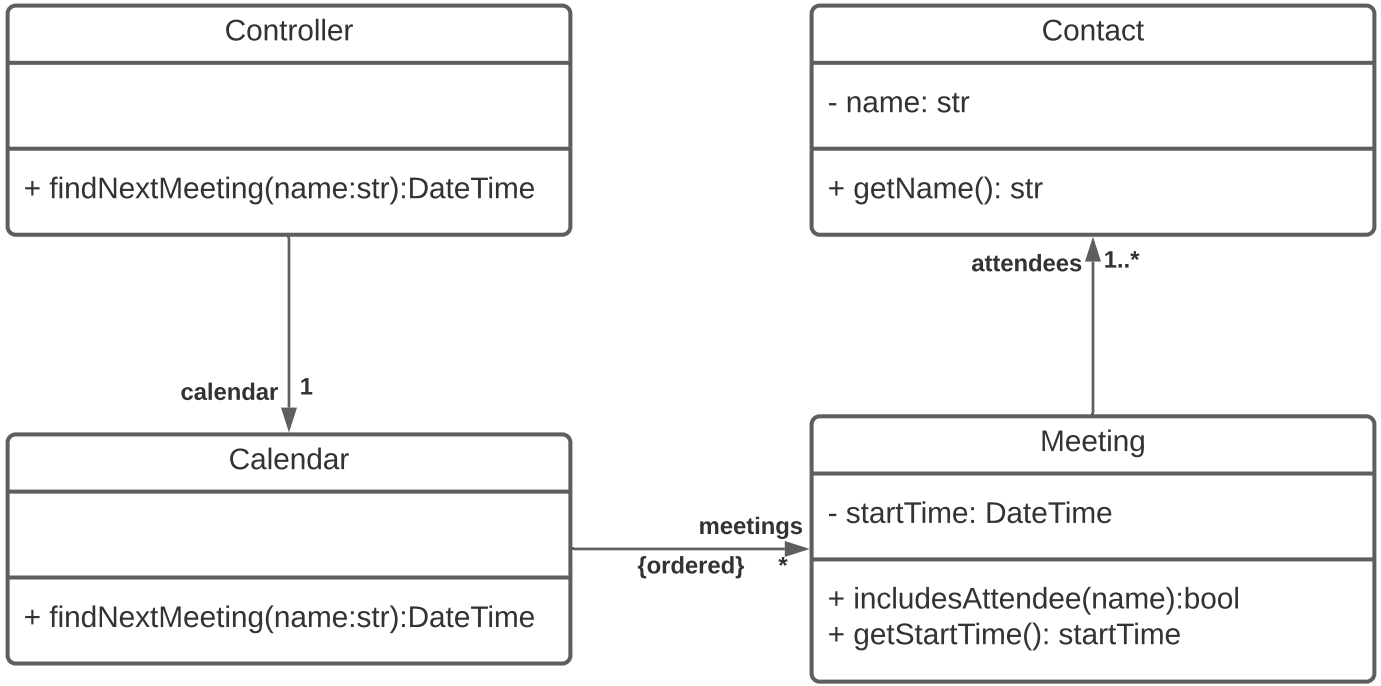


The associations should be directed, not use aggregation

Diagram Consistency

- Operations received by objects in the interaction diagrams occur in the corresponding class in the Class Diagram
- The classes have attributes for their encapsulated data
  - No attributes for transient data elements should be introduced
- Relationships between classes are defined
  - Directed, labelled with role and multiplicity

Find a Meeting Class Diagram



Find a Meeting in Python (1/2)

```
class Controller:
    def findNextMeeting(self,name):
        return self.calendar.findNextMeeting(name)


class Calendar:
    def findNextMeeting(self, name):
        for m in self.meetings:
            if m.includesAttendee(name):
                dt = m.getStartDateTime()
                return dt
        return None
```

Find a Meeting in Python (2/2)

```
class Meeting:
    def includesAttendee(self, name):
        for c in self.contacts:
            if c.getName() == name:
                return True
        return False

    def getStartDateTime(self):
        return self.startDateTime
```

```
class Contact:

   def getName(self):
       return self.name
```

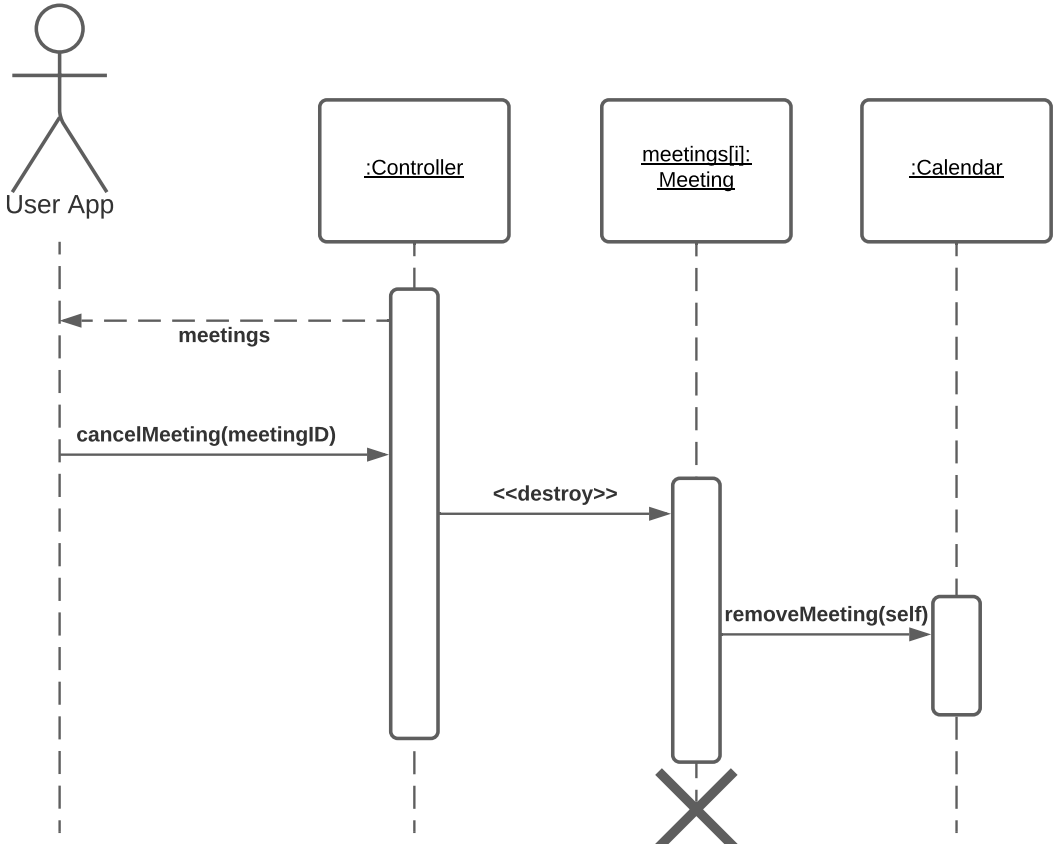
Task 4. Assess a Design

- Assess the interaction design for the use case *Cancel Meeting* given on the subsequent slide
- Is the interaction feasible?
- Does the interaction achieve all desired effects?
- Does the interaction satisfy design principles?

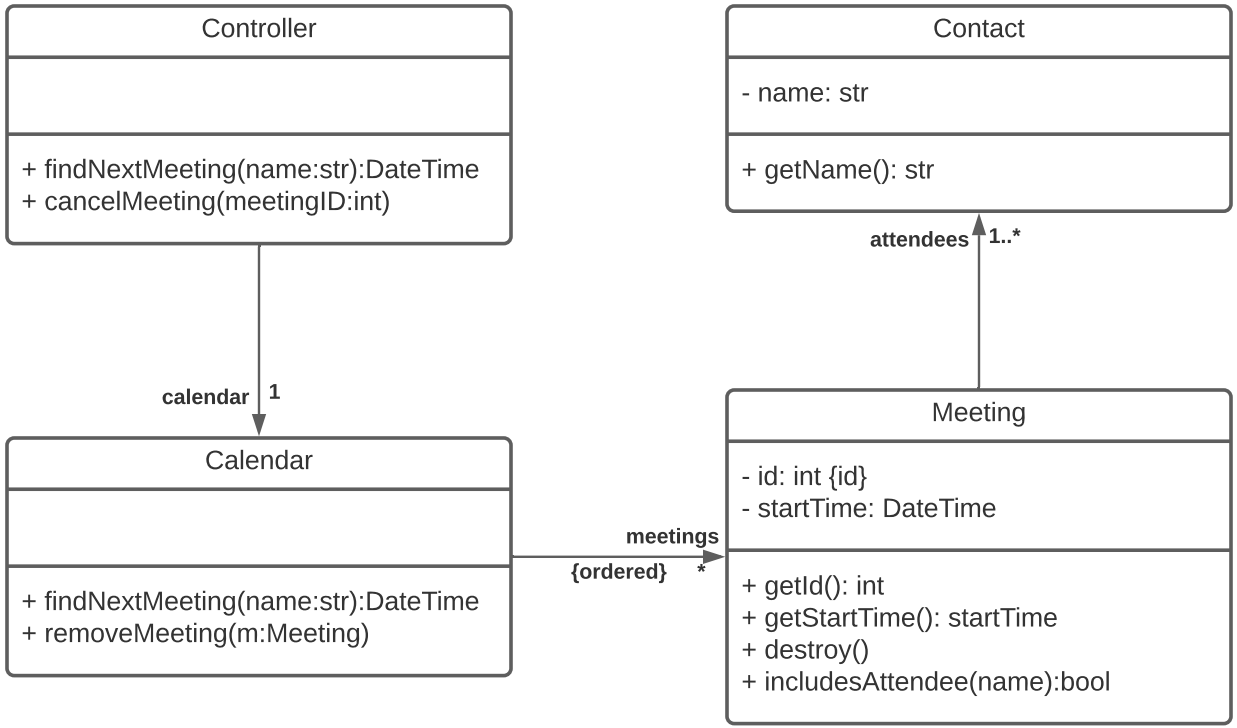
UC99 Cancel Meeting

1. User selects a meeting and requests deletion of the meeting
2. System deletes the meeting from calendar

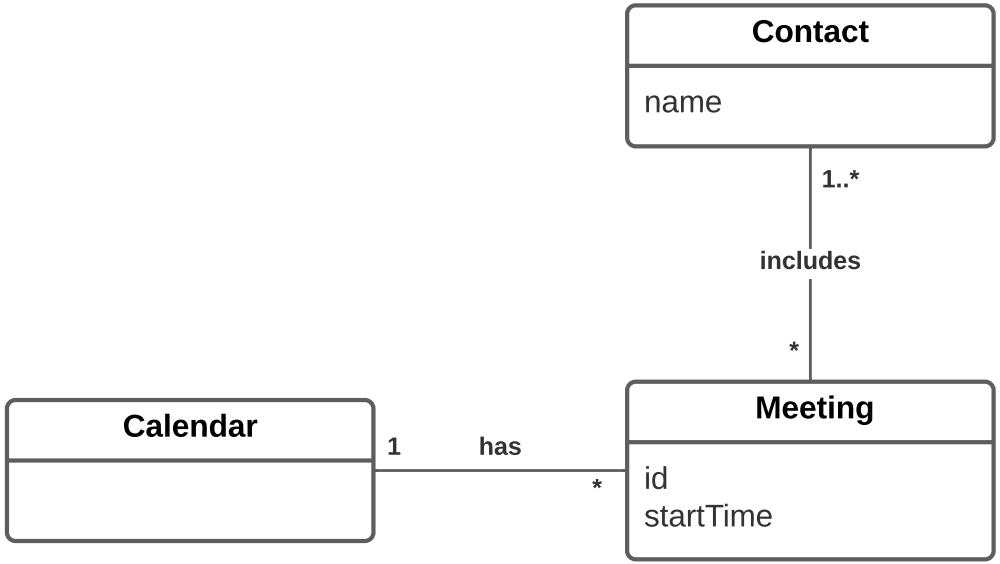
Proposed Interaction Design



Class Diagram for Previous Slide



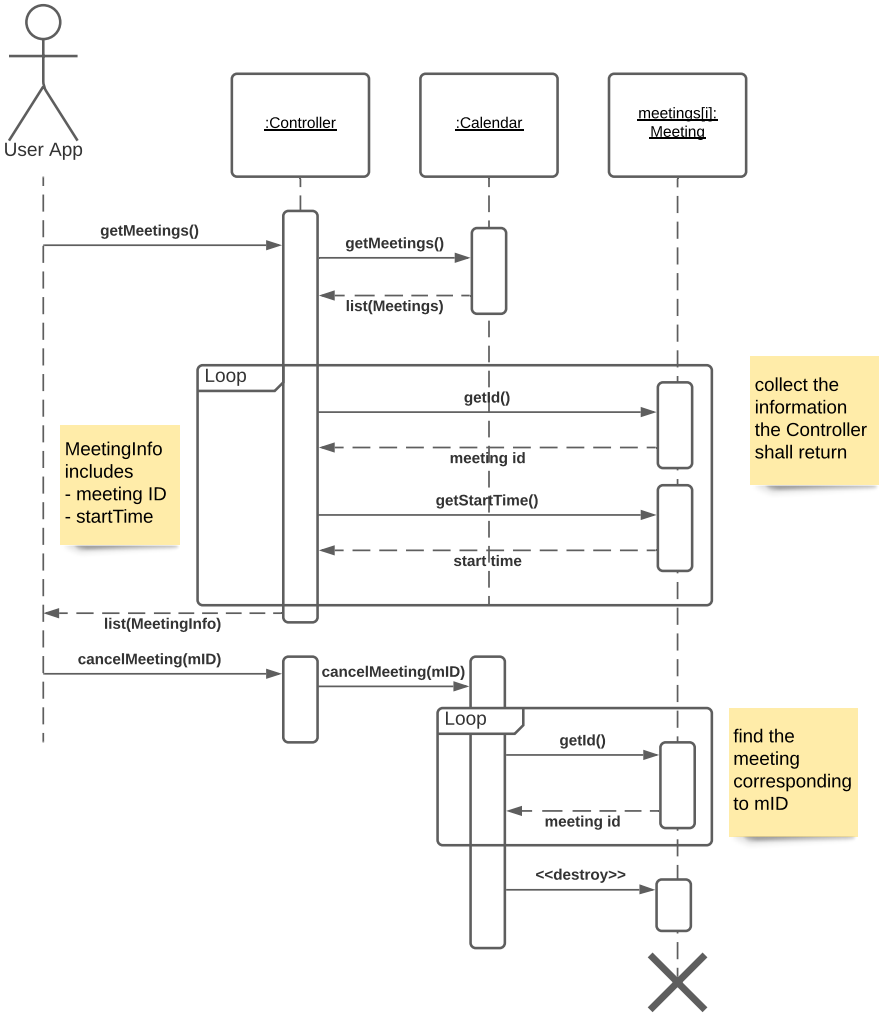
Domain Model for Calendar



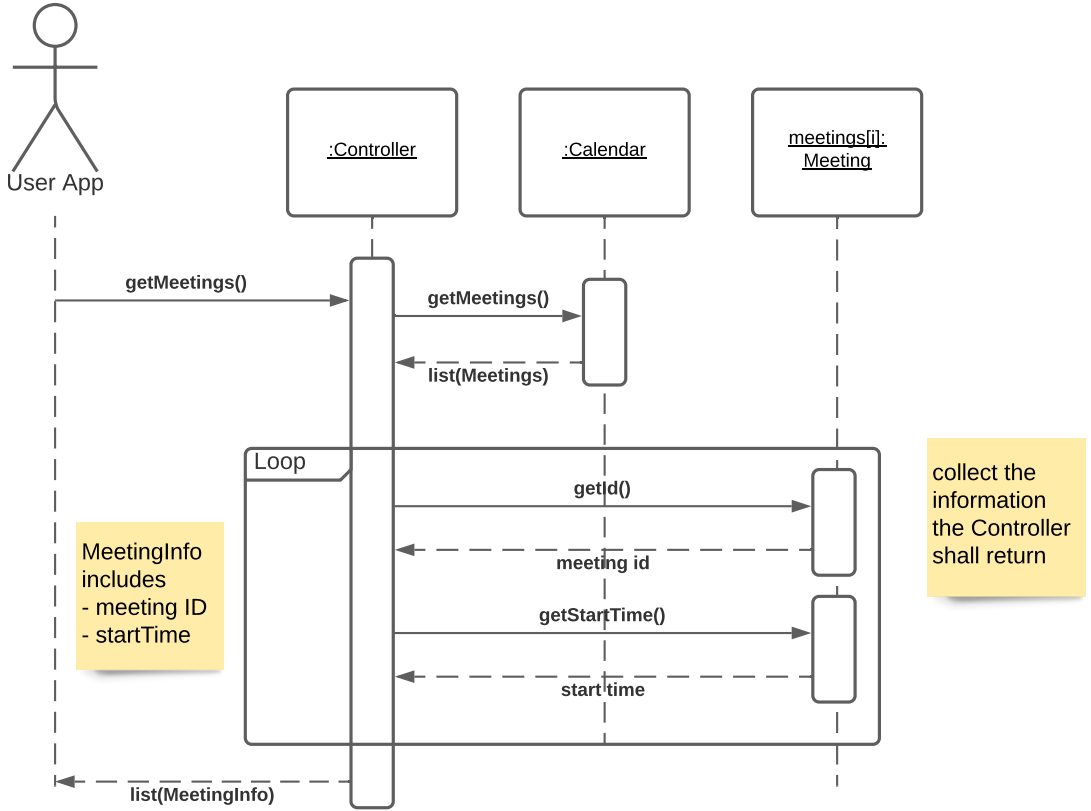
Issues with this design

- Controller does not know when the use case starts
  - When should it return the meetings to the User App?
- Controller does not have the meeting information
  - Calendar has that information. Controller needs to ask for it.
- Controller does not know which Meeting object corresponds to the given meetingID in cancelMeeting()
- Controller does not have visibility of the Meeting objects directly
- Meeting objects do not have visibility of the Calendar, hence, cannot invoke an operation in the Calendar

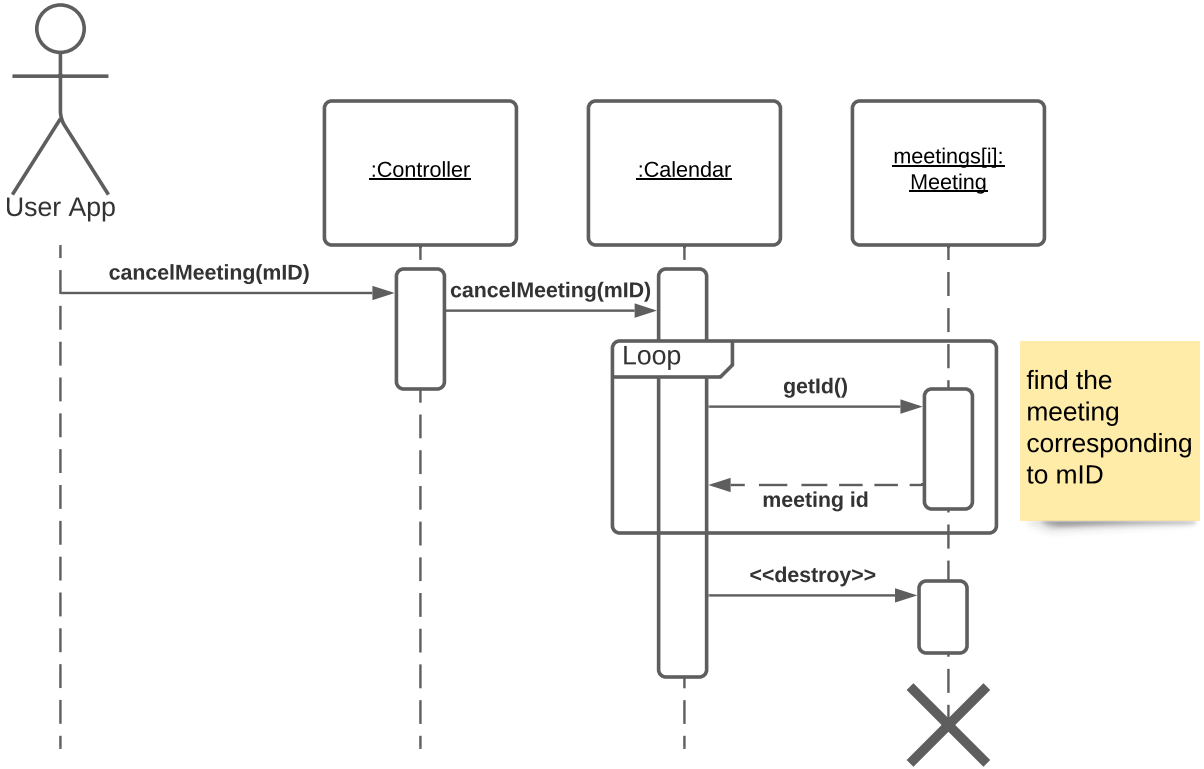
Revised Interaction Design (1/3)



Revised Interaction Design (2/2)

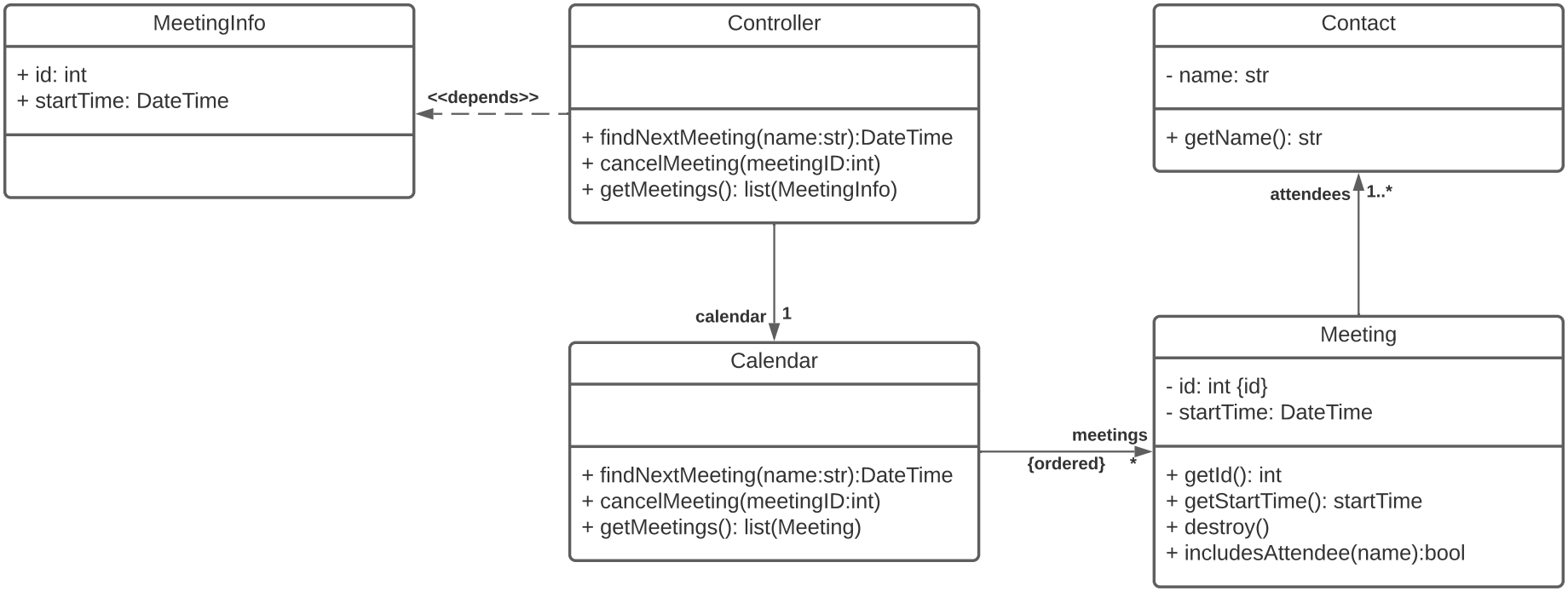


Revised Interaction Design (3/3)



University of South Au

Class Diagram for Revised Design



Calendar System in Python (1/3)

```
class Controller:
    def findNextMeeting(self, name):
        return self.calendar.findNextMeeting(name)

    def getMeetings(self):
        meetings = self.calendar.getMeetings()
        mInfo = [(m.getId(), m.getStartDateTime()) for m in meetings]
        return mInfo

    def cancelMeeting(self, mID):
        self.calendar.cancelMeeting(mID)
```

Calendar System in Python (2/3)

```
class Calendar:
    def findNextMeeting(self, name):
        for m in self.meetings:
            if m.includesAttendee(name):
                dt = m.getStartDateTime()
                return dt
        return None

    def cancelMeeting(self, mID):
        for m in self.meetings:
            if m.getId() == mID:
                self.meetings.remove(m)
                m.destroy()
                return
```

Calendar System in Python (3/3)

```
class Meeting:
    def includesAttendee(self, name):
        for c in self.contacts:
            if c.getName() == name:
                return True
        return False

    def getId(self):
        return self.id

    def getStartDateTime(self):
        return self.getStartDateTime
```

```
class Contact:

    def getName(self):
        return self.name
```

You Should Know

- How objects interact at runtime
- Document program structure and interactions using UML Diagrams
- Identify deficiencies in an interaction design

Activities this Week

- Attend second Workshop session
- Complete Quiz 4
- Start working on Assignment 1

University of
South Australia