ตัวอย่างการออกแบบระบบ Meeting Scheduler

ข้อมูลเบื้องต้น

ระบบนัดหมายประชุม เป็นระบบจัดการเพื่อกำหนดการประชุมสำหรับกลุ่มที่อยู่ในองค์กร โดยการจัดประชุมจะพิจารณาจากเวลาที่ว่างของสถานที่ประชุมว่าง ระบบจะให้สมาชิกสามารถจองการประชุม ยกเลิกการประชุม โดยสมาชิกที่ได้รับเชิญเข้าประชุมจะได้รับการแจ้งเตือนว่ามีการประชุม และแม้จะกำหนดการประชุมไปแล้ว ยังอนุญาตให้เพิ่มสมาชิกคนอื่นๆ ที่ว่างในเวลานั้นเข้าในการประชุมได้ด้วย

ฟังก์ชันที่ระบบสามารถทำได้

- การจองห้อง (Room assignment) ระบบนัดหมายการประชุมจะต้องรับผิดชอบในการจองห้องประชุม ดังนั้นระบบจะต้องตรวจสอบได้ว่าห้องใดว่างหรือไม่ว่าง และ แต่ละห้องมีความจุเท่าไร
- เวลาว่างของสมาชิก (Availability of attendees) เนื่องจากการประชุมประกอบด้วยสมาชิกหลายคน สมาชิกแต่ละคนก็มีภารกิจและกำหนดการต่างกัน ดังนั้นระบบจะต้องตรวจสอบได้ว่าสมาชิกว่างในเวลาที่ประชุมหรือไม่ และระบบจะต้องเก็บข้อมูลการประชุมของสมาชิกทั้งหมด

แนวทางการออกแบบ

แนวทางการออกแบบจะใช้วิธี bottom-up design โดยมีขั้นตอนดังนี้

- ระบุและออกแบบองค์ประกอบที่เล็กก่อน เช่น ระยะเวลา และ ห้องประชุม
- จากนั้นค่อยเพิ่มส่วนประกอบเพิ่มเติม เช่น การประชุม และ ปฏิทิน
- ทำซ้ำในขั้นตอนข้างต้น จนถึงส่วนประกอบหลักของระบบ

ข้อกำหนดของระบบนัดหมายประชุม

1. จะต้องกำหนดจำนวนห้องประชุมได้ เช่น สมมติให้มี n ห้อง
2. แต่ละห้องประชุมจะต้องระบุความจุ หรือ จำนวนผู้เข้าประชุมว่ารองรับได้กี่คน
3. ถ้าห้องประชุมนั้นยังไม่ได้ถูกจอง สามารถจะจองห้องประชุมนั้นได้ โดยการจองจะต้องกำหนดระยะเวลาประชุม วัน เวลาเริ่มต้น และเวลาสิ้นสุดของการประชุม
4. เมื่อกำหนดการประชุมแล้ว ระบบจะส่งข้อความเตือน (Notification) ไปยังสมาชิกที่ได้รับเชิญเข้าประชุม
5. ผู้ใช้ทุกคนจะได้รับเชิญประชุม (invite) ไม่ว่าผู้ใช้จะว่างในช่วงเวลานั้นหรือไม่ โดยผู้ใช้สามารถจะตอบรับ (accept) หรือ ปฏิเสธ (reject) การเชิญนั้นก็ได้
6. ผู้ใช้แต่ละคนจะมีปฏิทิน เพื่อเอาไว้ดูกิจกรรมในวันและเวลาต่างๆ เพื่อใช้ในการพิจารณาว่าจะเข้าประชุมหรือไม่เข้าประชุม
7. เฉพาะผู้ใช้บางคนเท่านั้นที่มีสิทธิ์สร้างการประชุม

Use Case Diagram สำหรับระบบ Meeting Scheduler

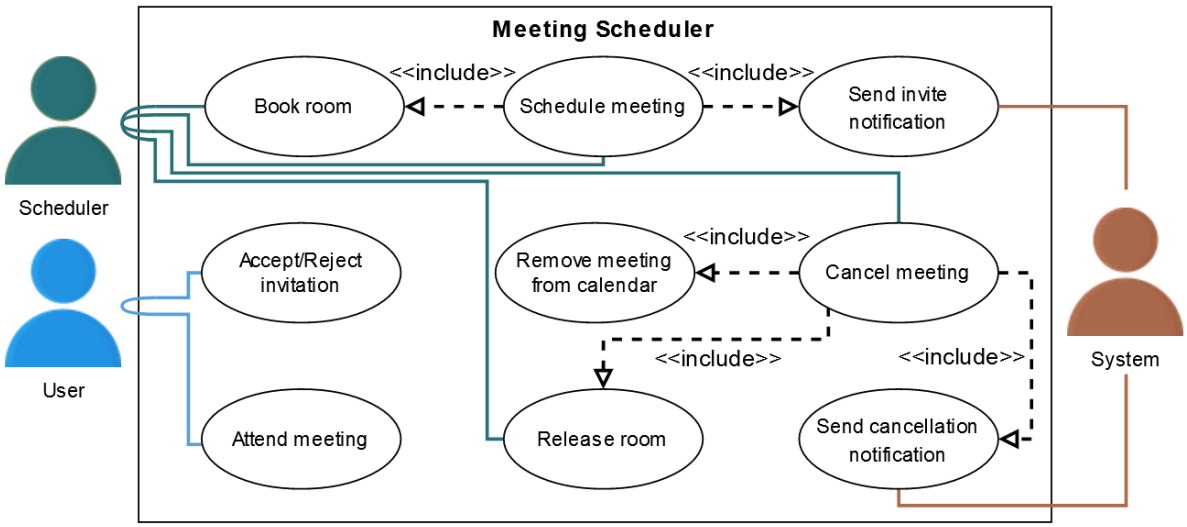
- **ระบบ** (system) จะเรียกระบบที่จะสร้าง use case ว่า "meeting scheduler"
- **Actor**
    - Actor หลัก (Primary Actor)
        - Scheduler หมายถึง Actor ที่มีสิทธิ์ในการสร้างการประชุม และ ยกเลิกการประชุม รวมถึงจองห้องประชุม และคืนห้องประชุม
        - User หมายถึง Actor ที่ตอบรับหรือปฏิเสธการเชิญประชุม และตัดสินใจว่าจะเข้าประชุมหรือไม่
    - Actor รอง (Secondary Actor)
        - System เป็น Actor ที่รับผิดชอบในการส่ง Notification เกี่ยวกับการประชุมและการยกเลิกการประชุม
- Use Cases จะใช้วิธีระบุกิจกรรมที่กระทำโดยแต่ละ Actor
    - Scheduler
        - Schedule/Cancel meeting : เป็นกิจกรรมที่สร้างกำหนดการประชุม หรือ ยกเลิกการประชุมเดิม (หากมองเป็น UI คือ ทั้งสร้างและยกเลิกเป็นกิจกรรมในหน้าเดียวกัน จึงรวมไว้ใน Use Case เดียวกัน)
        - Book/Release Room : เป็นกิจกรรมของการจองห้องประชุม และ คืนห้องประชุม
    - User
        - Attend meeting : เป็นกิจกรรมสำหรับการเลือกเข้าประชุม
        - Accept/Reject meeting: เป็นกิจกรรมในการตอบรับหรือปฏิเสธจะเข้าร่วมประชุม
    - System
        - Send invite notification : เป็นกิจกรรมในการส่ง notification ของการเชิญประชุม
        - Send cancelation notification : เป็นกิจกรรมในการส่ง notification ของการยกเลิกการประชุม
- ความสัมพันธ์ระหว่าง Actor กับ Use case สามารถเขียนได้ตามตารางนี้

Scheduler	User	System
Schedule/Cancel meeting	Attend meeting	Send invite notification
Book/Release room	Accept/Reject meeting	Send cancelation notification

- Include
    - กิจกรรมใน Use case "Schedule meeting" จะ include Use case "Book room" เนื่องจากการจัดประชุม จำเป็นต้องใช้ห้อง
    - กิจกรรมใน Use case "Schedule meeting" จะ include Use case "Send invite notification" เนื่องจากหลังจากที่สร้างการประชุมแล้ว จะต้องส่งการเชิญประชุมตามมา

- กิจกรรมใน Use case " Cancel meeting" จะ include Use case " Release room" เนื่องจากเมื่อยกเลิกการจัดประชุม ห้องก็ไม่ต้องใช้แล้วก็คืนห้องไป
- กิจกรรมใน Use case " Cancel meeting" จะ include Use case "Send cancelation notification" เนื่องจากเมื่อยกเลิกการจัดประชุม ก็จะต้องแจ้งให้ผู้ที่ถูกเชิญทราบ
- กิจกรรมใน Use case " Cancel meeting" จะ include Use case "Remove meeting from calendar" (Use case นี้ จะไม่มีความสัมพันธ์กับผู้ใช้ เนื่องจากการการใส่การประชุมในปฏิทิน จะรวมอยู่ในการตอบรับเข้าประชุมแล้ว แต่เมื่อยกเลิกการประชุม ไม่ได้มีการตอบรับ ดังนั้นจึงต้องมีกิจกรรมในการลบกำหนดการของการประชุมนั้นในปฏิทินของทุกคนออก

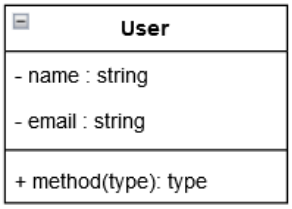
- เมื่อนำส่วนต่างๆ ข้างต้นมารวมกัน สามารถจะเขียนเป็น Use Case Diagram ได้ดังนี้



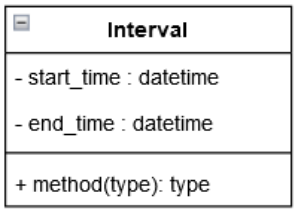
Class Diagram สำหรับระบบ Meeting Scheduler

ในการออกแบบ class diagram จะใช้วิธี bottom-up โดยจะออกแบบคลาสพื้นฐานก่อน

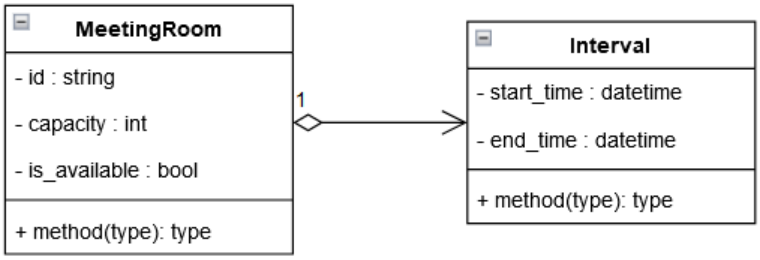
- **คลาส User** คลาสนี้จะรับผิดชอบในการเก็บข้อมูลส่วนบุคคล ซึ่งในระบบ จะเก็บข้อมูลผู้ใช้เพียง 2 ข้อมูล คือ ชื่อ สำหรับอ้างอิง และ email สำหรับใช้สื่อสาร เช่น invite คลาส User จะแสดง ดังนี้



- **คลาส Interval** โดยคลาสนี้ จะเก็บ วัน/เวลาเริ่มประชุม และวัน/เวลา สิ้นสุดการประชุม เป็นคลาสที่เป็นตัวแทนของการประชุมแต่ละครั้ง เหตุที่ควรสร้างเป็นคลาส เนื่องจากจะมีการใช้ Interval ใน 2 ที่ คือ ระบุช่วงเวลาในการประชุม และ ใช้ในการบอกว่าห้องประชุม มีการประชุมในช่วงเวลาใดบ้าง ดังนั้นการสร้างเป็นคลาสจะดีกว่า เพราะเป็นการเก็บข้อมูลไว้ที่เดียว ไม่เกิดข้อมูลซ้อน คลาส Interval จะแสดง ดังนี้

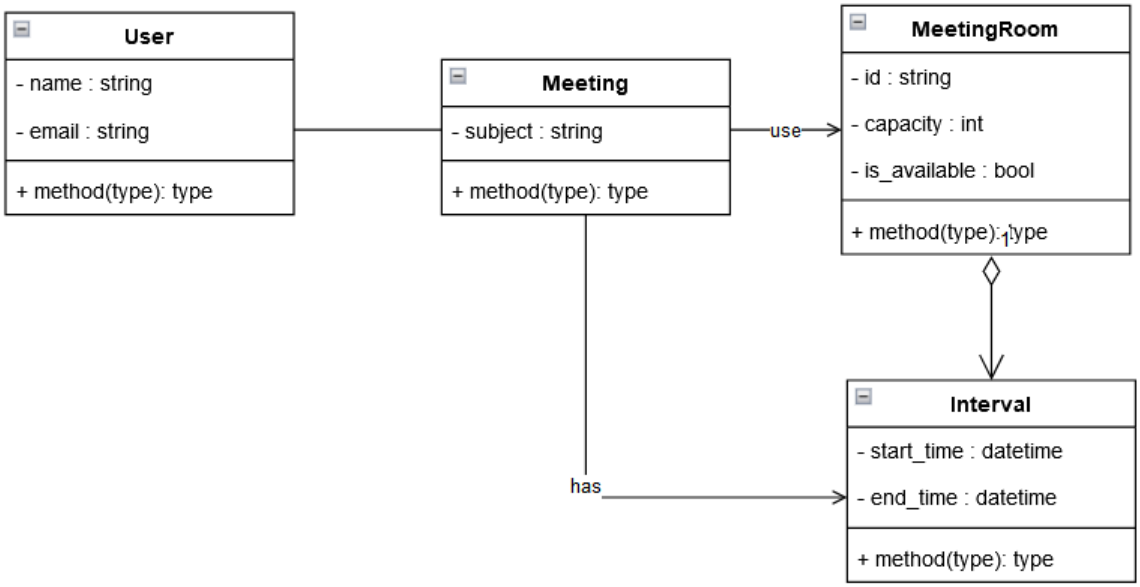


- **คลาส MeetingRoom** จะเก็บข้อมูลห้องประชุม ประกอบด้วย หมายเลขห้อง ความจุ และ สถานะ (ที่ต้องมีสถานะเนื่องจากห้องอาจไม่พร้อมใช้) คลาสห้องประชุม จะมีความสัมพันธ์แบบ Aggregation กับคลาส Interval เนื่องจากห้องประชุมจะต้องบอกช่วงเวลาที่มีการใช้งานและที่ว่าง นอกจากนั้นข้อมูลทุกคลาสจะต้องมีตำแหน่งที่เก็บ Instance ของคลาสนั้น ซึ่งคลาสที่เหมาะจะเก็บ Interval ที่สุด คือ คลาส MeetingRoom ซึ่งแสดงความสัมพันธ์ระหว่าง MeetingRoom กับ Interval ดังนี้ (อาจกล่าวได้ว่า Meeting Room คือ **ที่รวม**ของ Interval)

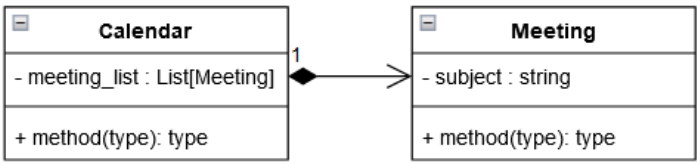


- **คลาส Meeting** จะเก็บข้อมูลการประชุม ได้แก่ หัวข้อประชุม คลาส Meeting จะมีความสัมพันธ์แบบ Association ดังนี้
    - ใน “การประชุม” มี “ผู้เข้าร่วมประชุม” และ “ผู้เข้าร่วมประชุม” เข้าร่วม “การประชุม” จึงเป็นความสัมพันธ์แบบสองทาง
    - ใน “การประชุม” ใช้ “ห้องประชุม” เป็นความสัมพันธ์แบบทางเดียว
    - ใน “การประชุม” มีการกำหนด “ช่วงเวลา เป็นความสัมพันธ์แบบทางเดียว

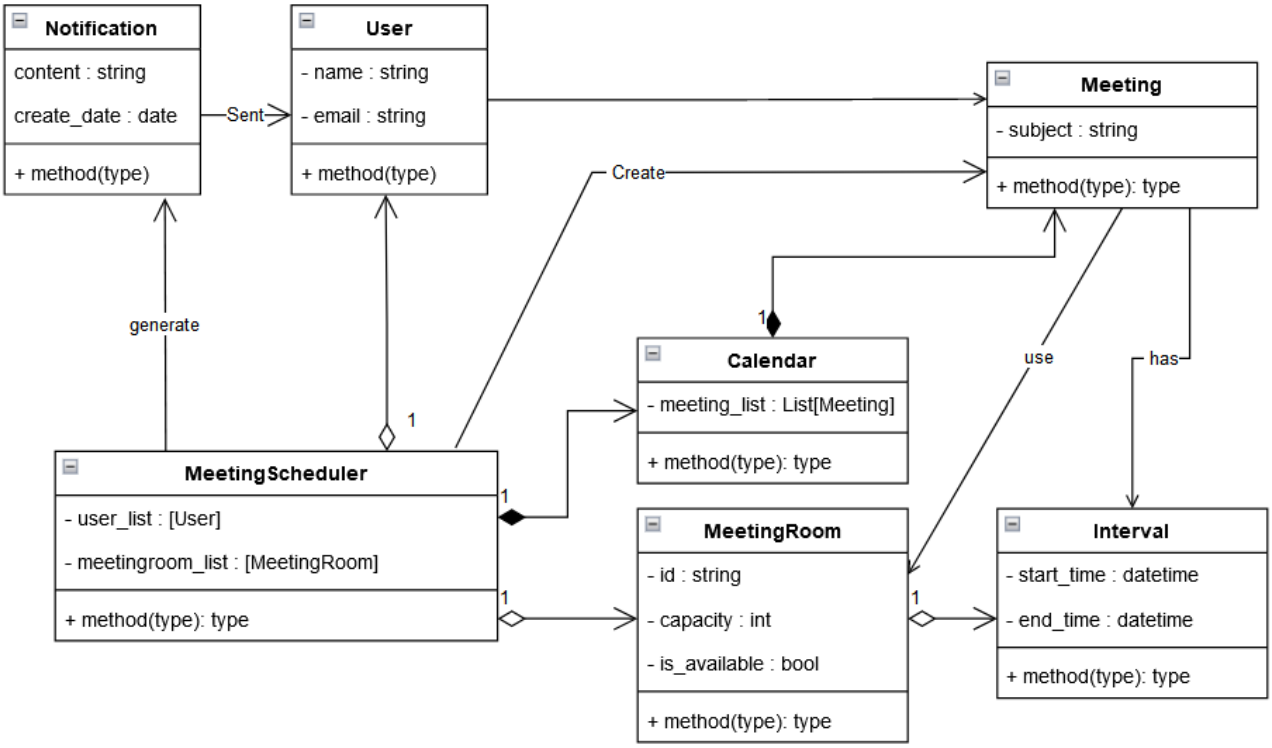
สามารถเขียน Class ที่แสดงความสัมพันธ์ระหว่าง **Meeting, MeetingRoom, Interval และ User**



- **คลาส Calendar** คลาสนี้ไม่ใช่คลาสปฏิทินปกติ แต่เป็นปฏิทินการประชุม สาเหตุที่ต้องสร้างคลาส Calendar ขึ้นมา เนื่องจาก Instance ของ Meeting จะต้องมีที่เก็บ และเพื่อใช้ในการแสดงว่าการประชุมมีในวันเวลาใดบ้าง ดังนั้นจึงสร้างคลาสขึ้นมา คลาสนี้ทำหน้าที่รวบรวม Meeting จึงสามารถจะใช้ความสัมพันธ์แบบ Aggregation หรือหากจะกล่าวว่า Meeting เป็น “ส่วนประกอบ” ของ Calendar ก็สามารถจะใช้ความสัมพันธ์แบบ Composition ได้ ในที่นี้จะใช้เป็น Composition



- **คลาส MeetingScheduler** คลาสนี้เป็นตัวแทนของระบบ ทำหน้าที่เก็บข้อมูลของระบบ ได้แก่ ข้อมูลผู้ใช้ (คลาส User) ข้อมูลห้อง (คลาส Room) และทำหน้าที่ในการสร้าง Notification สามารถเขียนความสัมพันธ์กับคลาสอื่นๆ ได้ดังนี้



จะเห็นว่าคลาส MeetingScheduler จะเก็บข้อมูลของระบบ ได้แก่ List ของผู้ใช้ และ List ของห้องประชุม โดยได้ใส่ความสัมพันธ์แบบ Association กับคลาส Meeting ซึ่งจุดนี้อยู่ที่มุมมองของผู้ออกแบบระบบ โดยอาจมองว่าผู้ที่สร้าง “การประชุม” คือ ผู้ใช้ ซึ่งจะทำให้คลาส User มี 2 ความสัมพันธ์กับคลาส Meeting คือ สร้างการประชุม และ เข้าร่วมประชุม ผู้ออกแบบจึงออกแบบให้คลาส MeetingScheduler ทำหน้าที่เป็นผู้สร้างการประชุมแทน เนื่องจากใน MeetingScheduler เป็นที่รวบรวมผู้ใช้อยู่แล้ว จึงสามารถออกแบบตามนี้ได้ อีกจุดหนึ่ง คือ ออกแบบให้ MeetingScheduler มีความสัมพันธ์กับ Calendar ในแบบ Composition เนื่องจากมองว่า Calendar เป็นส่วนประกอบของ MeetingScheduler ซึ่งจะมีผลให้มี Instance ของ Calendar เก็บไว้ใน MeetingScheduler ด้วย

- **คลาส Notification** เป็นคลาสที่เก็บข้อมูลของ Notification ที่ส่งไปประกอบด้วย ข้อความที่ส่ง และวันที่สร้าง คลาสนี้จะมีความสัมพันธ์แบบ Association กับ MeetingScheduler โดย Notification จะ generate โดย MeetingScheduler และส่งไปยัง User

Sequence Diagram สำหรับระบบ Meeting Scheduler

Sequence Diagram เป็น diagram สำหรับช่วยทำความเข้าใจการโต้ตอบระหว่าง object ต่างๆ ในระบบ แต่ในที่นี้จะยกตัวอย่างเพียง 2 sequence diagram คือ

- Schedule a meeting
- Cancel meeting:

Sequence Diagram สำหรับ Schedule a meeting

เป็น sequence diagram ในการสร้าง meeting

ขั้นที่ 1 หา Actor และ Object

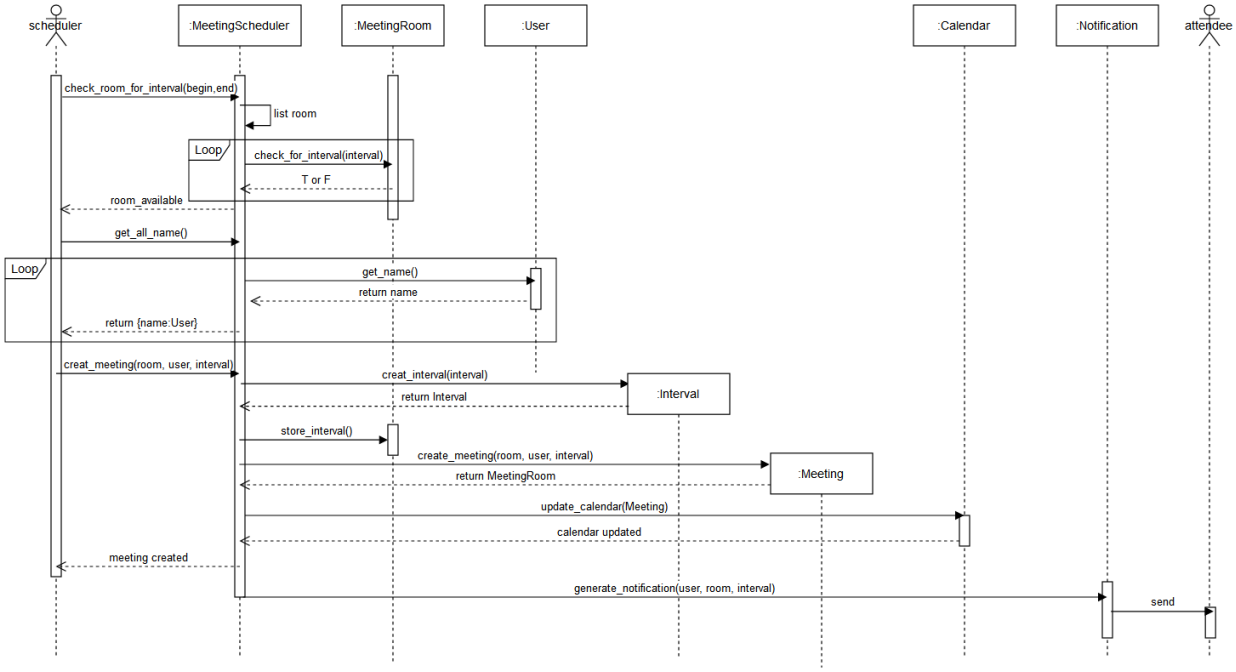
Actor : หากดูจาก use case diagram จะพบว่ามี Actor เดียว คือ Scheduler

Object : อาจเขียนเป็นภาษาพูดก่อน เช่น Scheduler สร้าง Meeting ซึ่งต้องระบุห้อง สร้าง Interval จากนั้นนำ Meeting ไปเก็บใน Calendar โดยให้หาคลาสที่เกี่ยวข้องมาเขียนลงในประโยคให้หมด ก็จะสามารถหา Object ได้ ซึ่งกรณีนี้ คือ Scheduler, Calendar, MeetingRoom, Interval, และ Meeting

ขั้นที่ 2 เขียนขั้นตอนทำงาน ซึ่งหากใน use case description เขียนได้ละเอียดแล้วก็สามารถนำมาใช้ได้เลย โดยขั้นตอนในการทำงาน Schedule a meeting มีดังนี้

1. เมื่อ Scheduler ต้องการสร้าง Meeting ใน Interval ที่กำหนด ต้องค้นหาห้องที่ว่างในช่วงเวลานั้น
2. จากนั้นจะค้นหาและเพิ่มชื่อผู้ที่จะเชิญเข้าร่วมประชุม
3. เมื่อได้ทั้งช่วงเวลา ผู้เข้าร่วมประชุม และ ห้องประชุมก็จะทำการจองห้อง
    a. จะสร้าง Interval
    b. สร้าง Meeting และนำ Interval ที่สร้างขึ้นไปเก็บใน Meeting
4. นำ Meeting ที่สร้างไป update ใน calendar
5. แจ้งผู้สร้างว่าการสร้าง Meeting สำเร็จแล้ว
6. ส่ง Invite ไปยังผู้ได้รับเชิญให้เข้าประชุม

จากขั้นตอนข้างต้น สามารถเขียน sequence diagram ได้ดังนี้



หมายเหตุ การเขียนเส้น message ไปยัง object Interval และ Meeting หมายถึงการเรียก Constructor ของ Object เพื่อสร้าง Instance ไม่ใช่เป็นการเรียก method อื่นๆ ในคลาส

Code for the Meeting Scheduler

ในส่วนนี้จะแสดงผลของความสัมพันธ์ ที่มีผลต่อ Attribute ของคลาส

คลาส User เก็บข้อมูลผู้ใช้ จะไม่มีการเก็บข้อมูลความสัมพันธ์อื่นๆ แต่อาจเก็บข้อมูล Notification ก็ได้ หากโปรแกรมมีฟังก์ชันในการเรียกดู Notification ย้อนหลัง และอาจเก็บข้อมูล Meeting ถ้าโปรแกรมมีฟังก์ชันเรียกดูการประชุม แต่ในที่นี้เป็นโปรแกรมอย่างง่ายจึงไม่เก็บ

```
class User:
  def __init__(self, name, email):
    self.__name = name
    self.__email = email
```

คลาส Interval เก็บข้อมูลวัน/เวลาเริ่มต้นและสิ้นสุดของการประชุม

```
class Interval:
  def __init__(self, start_time, end_time):
    self.__start_time = start_time
    self.__end_time = end_time
```

คลาส MeetingRoom เป็นตัวแทนของห้องประชุม โดยมีข้อมูลของตนเอง คือ capacity และความพร้อมใช้ (is_available) และมีข้อมูลที่เพิ่มมาจากความสัมพันธ์ คือ Interval คือ ช่วงเวลาที่มีการใช้งานห้องนั้นในการประชุม

```
class MeetingRoom:
  def __init__(self, id, capacity, is_available):
    self.__id = id
    self.__capacity = capacity
    self.__is_available = is_available
    self.__booked_intervals = []
```

คลาส Meeting เป็นตัวแทนของการประชุมแต่ละครั้ง เนื่องจากการสร้างการประชุมตาม Sequence Diagram จะทราบ Interval, Meeting และ Attendance List แล้ว จึงสามารถใส่ใน Constructor ตอนสร้างได้เลย สำหรับ self.__participants ในโปรแกรมจะกำหนดให้เป็น dictionary เพื่อให้สามารถเก็บ status ของแต่ละคนได้ด้วยว่า ตอบรับหรือยัง หรือ ปฏิเสธ ซึ่งจะต้องมี Code ส่วนที่แปลงจาก list ที่ส่งมาเป็น dictionary แต่ขอละเอาไว้

```
class Meeting:
    def __init__(self, id, participants, interval, room, subject):
        self.__id = id
        self.__participants = {}
        self.__interval = interval
        self.__room = room
        self.__subject = subject

  def add_participants(self, participants):
      pass
```

คลาส Calendar เป็นที่เก็บของ Meeting

```
class Calendar:
  def __init__(self):
    self.__meetings = []
```

คลาส MeetingScheduler เป็นคลาสหลักของระบบ เป็นตัวแทนของระบบ คลาสนี้ในระบบจะมีแค่ Instance เดียว (เรียกว่า Singleton) โดยจะมี argument คือ organizer ซึ่งเป็นบุคคลหรือกลุ่มบุคคลที่สามารถสร้างการประชุมได้ และยกเลิกการประชุมได้ ซึ่งรวมถึงการจองห้องและคืนห้อง โดยจะมี method ที่ใช้ตรวจสอบว่าห้องใดว่าง สำหรับการประชุมบ้าง และ ต้องมี method สำหรับสร้างการประชุม

```
class MeetingScheduler(organizer, calendar):
  def __init__(self, organizer, calendar):
    self.__organizer = organizer
    self.__calendar = calendar
    self.__rooms = []

  def schedule_meeting(self, users, interval):
    pass
  def cancel_meeting(self, users, interval):
    pass
  def book_room(self, room, number_of_persons, interval):
    pass
  def release_room(self, room, interval):
    pass
  def check_rooms_availability(self, number_of_persons, interval):
    pass
```

คลาส Notification รับผิดชอบในการส่งข้อความไปถึงผู้ใช้

```
class Notification:
  def __init__(self, notification_id, content, creation_date):
    self.__notification_id = notification_id
    self.__content = content
    self.__creation_date = creation_date
  def send_notification(self, user):
    pass
  def cancel_notification(self, user):
    pass
```