10.1 UML

a)

- **Borrow**
  - <<uses>>
  - Reservation
    - <<uses>>
    - Searching
      - <<uses>>
      - Notification
        - <<uses>>
        - Tracking
          - <<uses>>
          - Librarian
            - Library User
              - Assistant
                - Professor
              - Book-keeping System
Use Case: **Borrow**

**Actors:** Library User, Librarian, Library-keeping System

**Description:**
A library user wants to borrow a book. If the library user is a research assistant or professor the book can be borrowed without restrictions. If the borrow procedure is successful and the book is not reserved by another user (**Reservation**), the borrow period can be extended on request.

... Error Situations:
- Book is reserved by another user
- User wants to borrow a journal
- User not being a research assistant or a professor wants to borrow a book for longer than the book assigned time period
- ...

<<uses>>: **Reservation**
Use Case: Tracking

Actors: Library User, Librarian, Library-keeping System

Description:

The current position of a book is tracked. In case a library user has not brought back a book on time, he is sent an email and the librarian is notified (Notification).

... 

Error Situations:
  • User has not brought back a book on time
  • ... 

<<uses>>: Notification
Use Case: Searching

Actors: Library User, Librarian, Library-keeping System

Description:
All library users can search for a book by title, author, publication year. If a retrieved book is not present at the library, a reservation is possible (Reservation). The system automatically notifies the registered users by email (Notification) when a book returns to the library.

Error Situations:
• Retrieved book not present in the library
• ...

<<uses>>: Reservation, Notification
Separated into detailed class description and structure diagram for better presentation. Normally, all information is in one diagram.
Additional descriptions (e.g. role names, association names, constraints) can be attached to associations.
10.2 XML

a)
<employees>
  <employee>
    <FirstName>Jim</FirstName>
    <MiddleName>John</MiddleName>
    <LastName>Smith</LastName>
    <Salary>2000</Salary>
    <Position>Employee</Position>
  </employee>
  <employee>
    <FirstName>Luis</FirstName>
    <MiddleName>Giulliver</MiddleName>
    <LastName>Black</LastName>
    <Salary>2500</Salary>
    <Position>Employee</Position>
  </employee>
  <employee>
    <FirstName>Isabel</FirstName>
    <LastName>Siegel</LastName>
    <Salary>8000</Salary>
    <Position>CEO</Position>
    <Room>R101</Room>
  </employee>
  <employee>
    <LastName>Maibaum</LastName>
    <Salary>4500</Salary>
    <Position>Manager</Position>
    <Room>R201</Room>
  </employee>
</employees>
10.3 Questions on XML

a) Two syntactically different XML documents can have the same semantics. An XML Information Set defines in a clear way the syntax and the content of an XML document by information items. Therefore, it is more appropriate for the notion of equivalency between two XML documents. (e.g. how would you compare two source codes?)

b) Let’s suppose that we have a relational database containing multiple tables and multiple rows. \{T_1, T_2, \ldots, T_m\}
A randomly chosen table: T(A_1, A_2, \ldots, A_n) and a tuple in it t= \{v_1, v_2, \ldots, v_n\}

```xml
<Database>
  ...
  <T>
  ....
  <t>
    <A1>v_1</A1>
    <A2>v_2</A2>
    ...
    <A_n>v_n</A_n>
  </t>
  ....
  </T>
  ...
</Database>
```

```xml
<Database>
  ...
  <T>
  ....
  <t A_1=v_1 A_2= v_2 ... A_n= v_n />
  ....
  </T>
  ...
</Database>
```

c)  
- References are just syntax, no way to preserve referential integrity.
- No type information for the referenced elements.