Exercise 1:

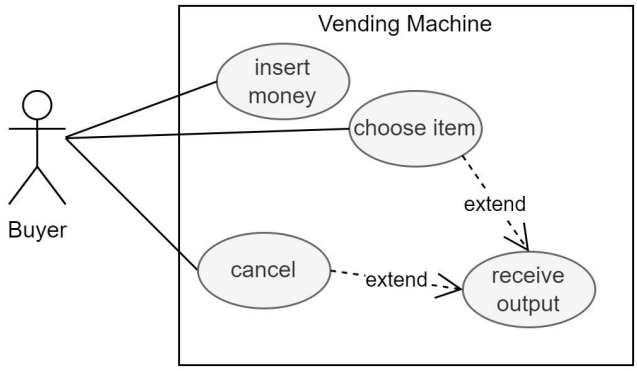
Extensions to the ER model are necessary for handling complex data requirements. These extensions address specialized relationship types, inheritance and generalization, aggregation and composition relationships, constraints and validation, semantic modeling, and temporal and historical data. By incorporating these extensions, the ER model becomes more expressive and capable of representing complex structures, enforcing data integrity, capturing semantics, and handling temporal aspects of the data.

Exercise 2:

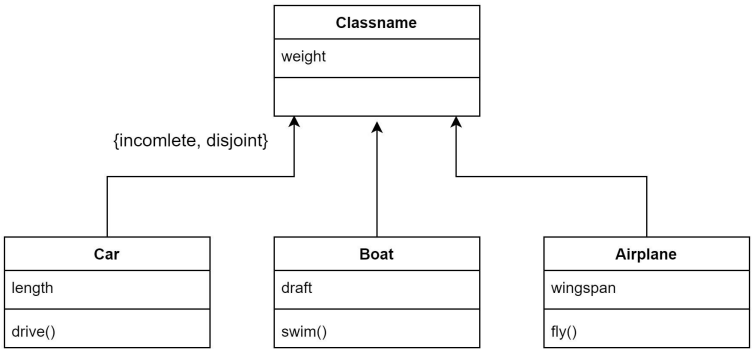
Object oriented data modeling is able to represent highly structured real world entities and concepts more directly through use of its classes with their attributes and methods, therefore leading to easier and more interpretable models. Letting data objects have their own type specific operations allows them to easily model more complex structures. Object oriented data models can also be changed and adapted more easily because individual objects can be changed without affecting the entire model. Furthermore OOMs benefit from object oriented concepts like Polymorphisms which creates more flexible models. Finally OOMs integrate nicely with object oriented programming languages which lessens the step from design to implementation.

Exercise 3:

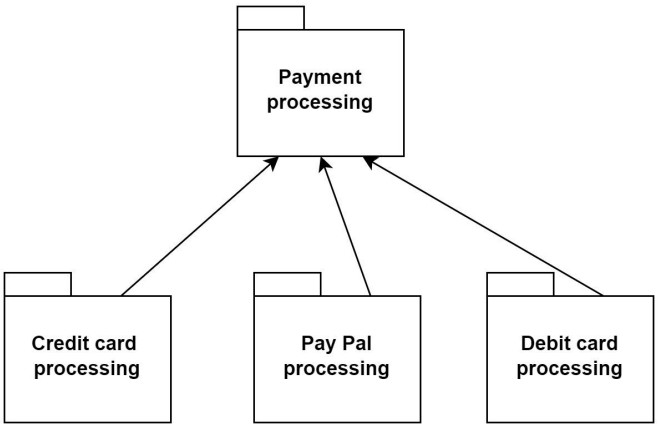
UML use case diagrams allow the modeling of a systems functionality from the user perspective or in other words user-system interaction. Modeling how users interact with the system helps to identify necessary functionality and scope as well as system components and dependencies. Use case diagrams also help to validate a system by identifying missing or faulty functionality.



UML class diagrams allow the modeling of a system's static structure like classes, methods, relationships etc. They enable the identification of a system's necessary components with their methods and attributes, aiding the development process. Class diagrams also serve as visual documentation of a systems structure.

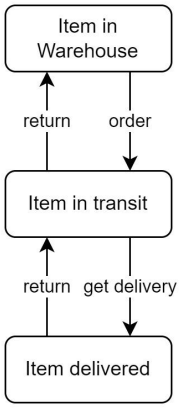


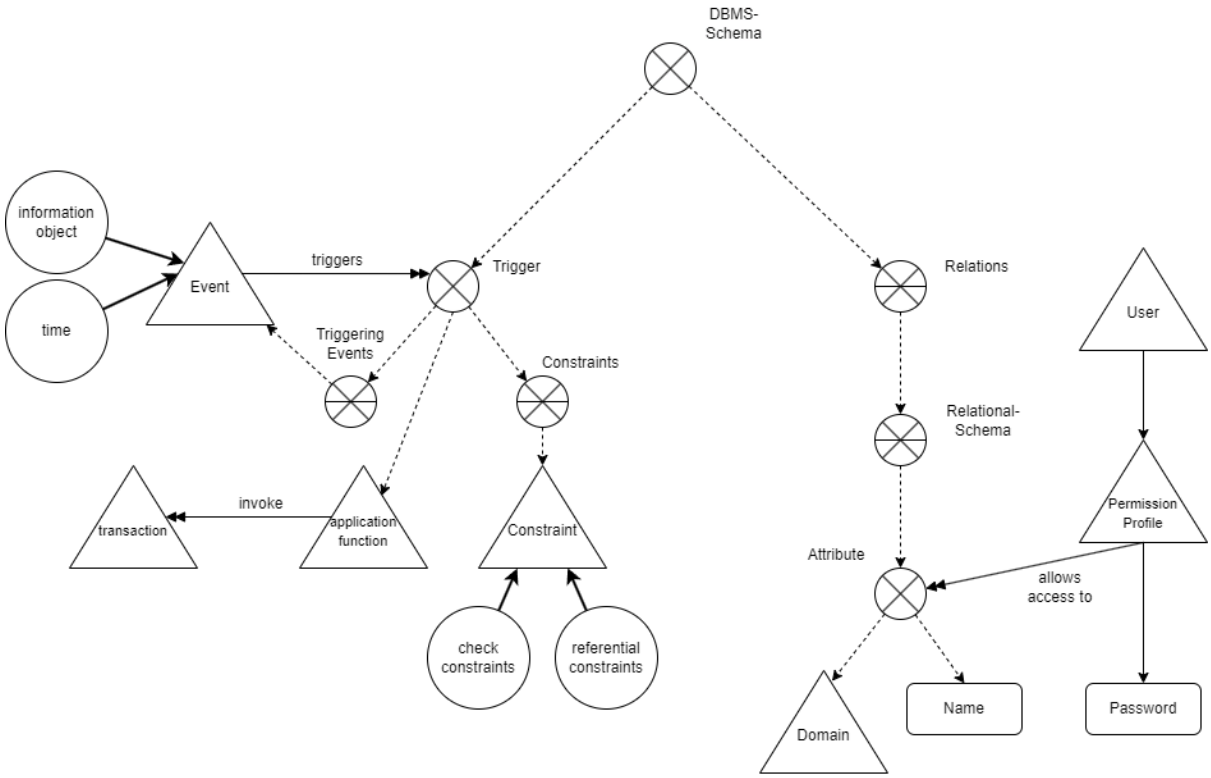
UML package diagrams provide a high level view of the components of a system. They help to properly manage and cluster related components into packages. This allows for easier organization and dependency management as well as documentation and understanding.



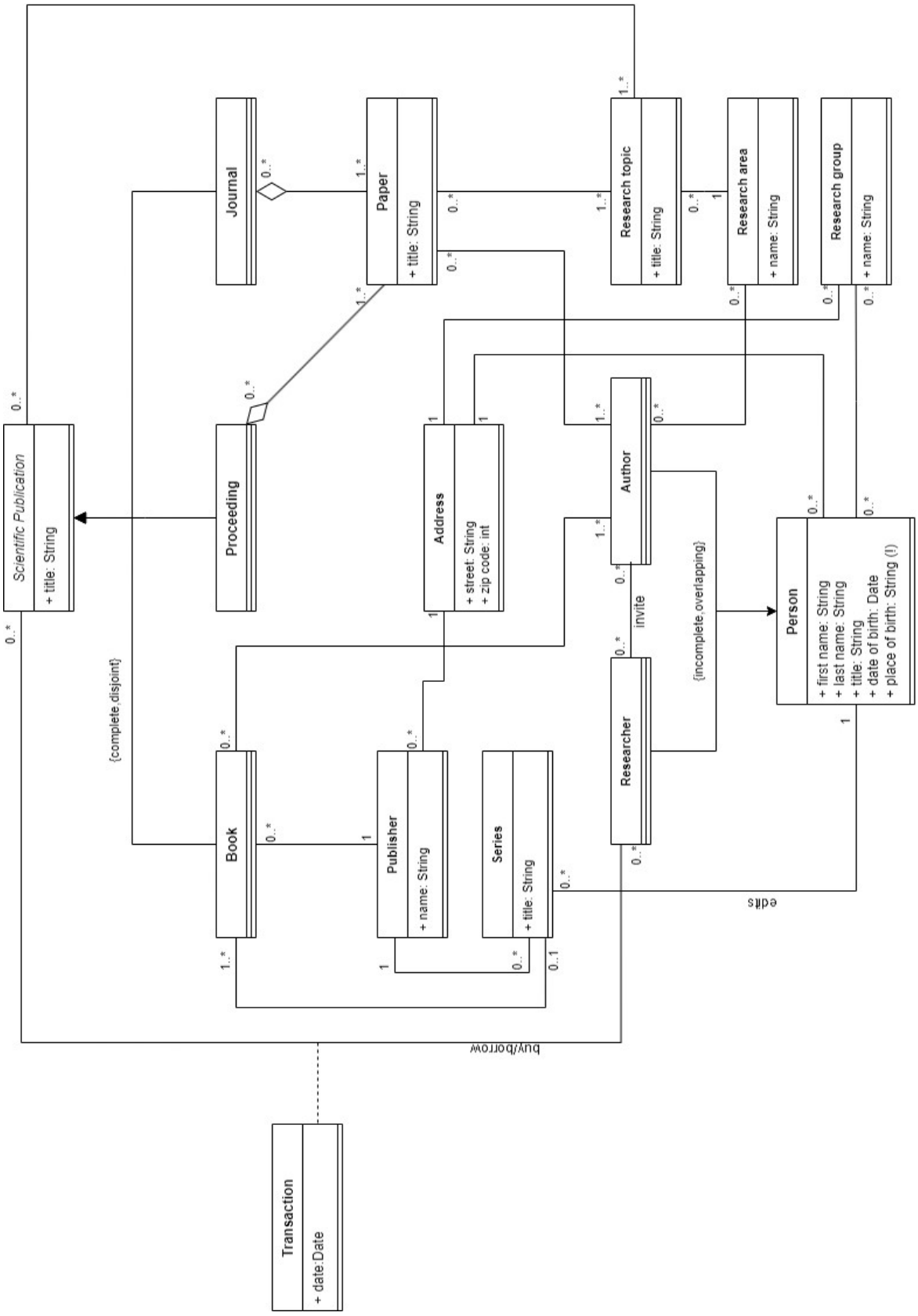
UML state diagrams model a systems temporal behavior by depicting its different states and their transitions. They help to analyse and efficiently design a systems operational flow.

Exercise 4:

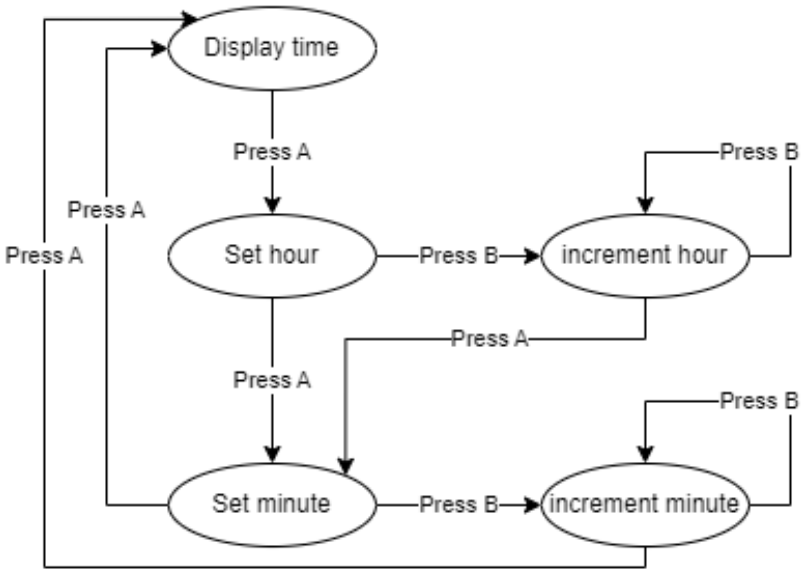



DBMS-
Schema
information
object
time
Event
triggers
Trigger
Triggering
Events
Constraints
Relations
User
Relational-
Schema
transaction
invoke
application
function
Constraint
Permission
Profile
Attribute
allows
access to
check
constraints
referential
constraints
Domain
Name
Password

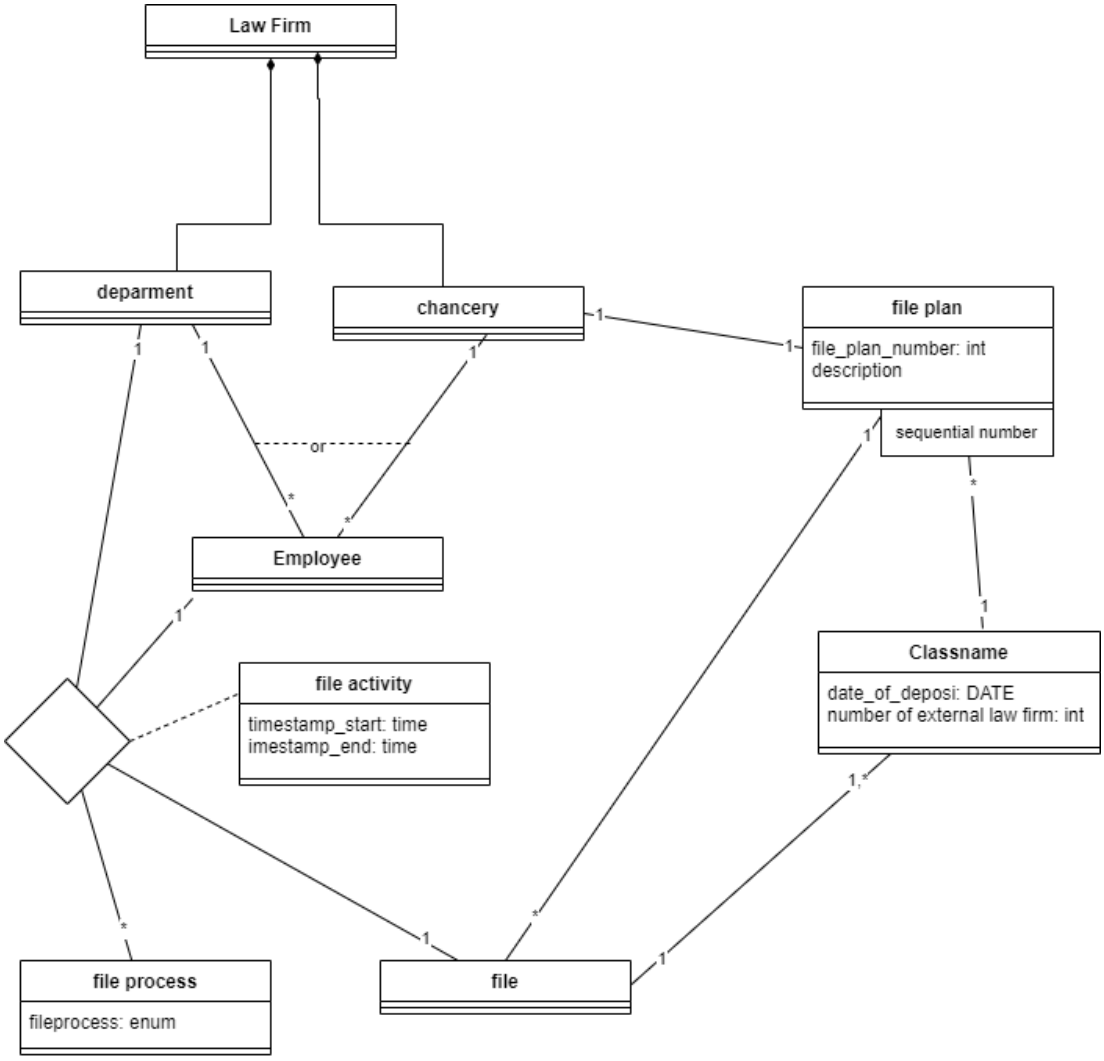
Exercise 5:


Scientific Publication
+ title: String
Journal
Paper
+ title: String
Research topic
+ title: String
Research area
+ name: String
Research group
+ name: String
Proceeding
Address
+ street: String
+ zip code: int
Author
Person
+ first name: String
+ last name: String
+ title: String
+ date of birth: Date
+ place of birth: String (!)
Book
Publisher
+ name: String
Series
+ title: String
Researcher
invite
{incomplete,overlapping}
{complete,disjoint}
edits
buy/borrow
Transaction
+ date:Date

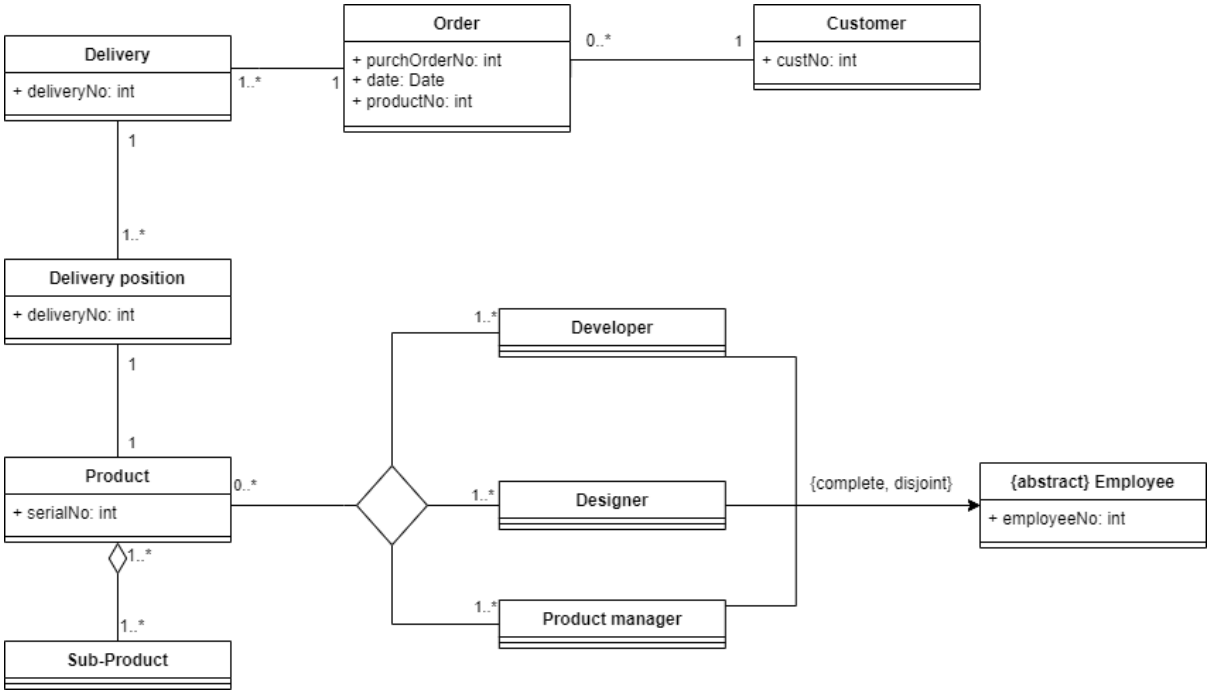
Exercise 6:



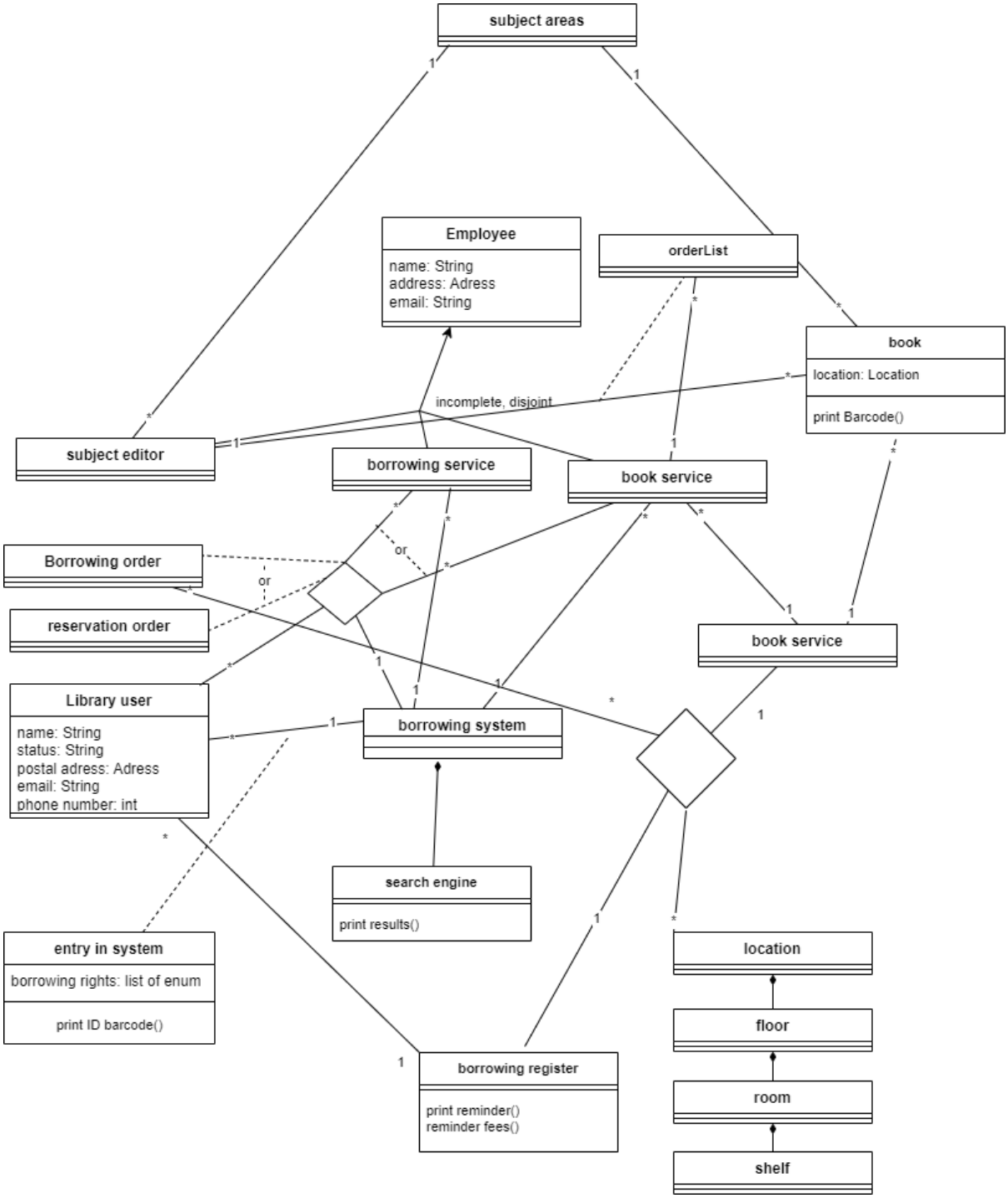
Exercise 7:



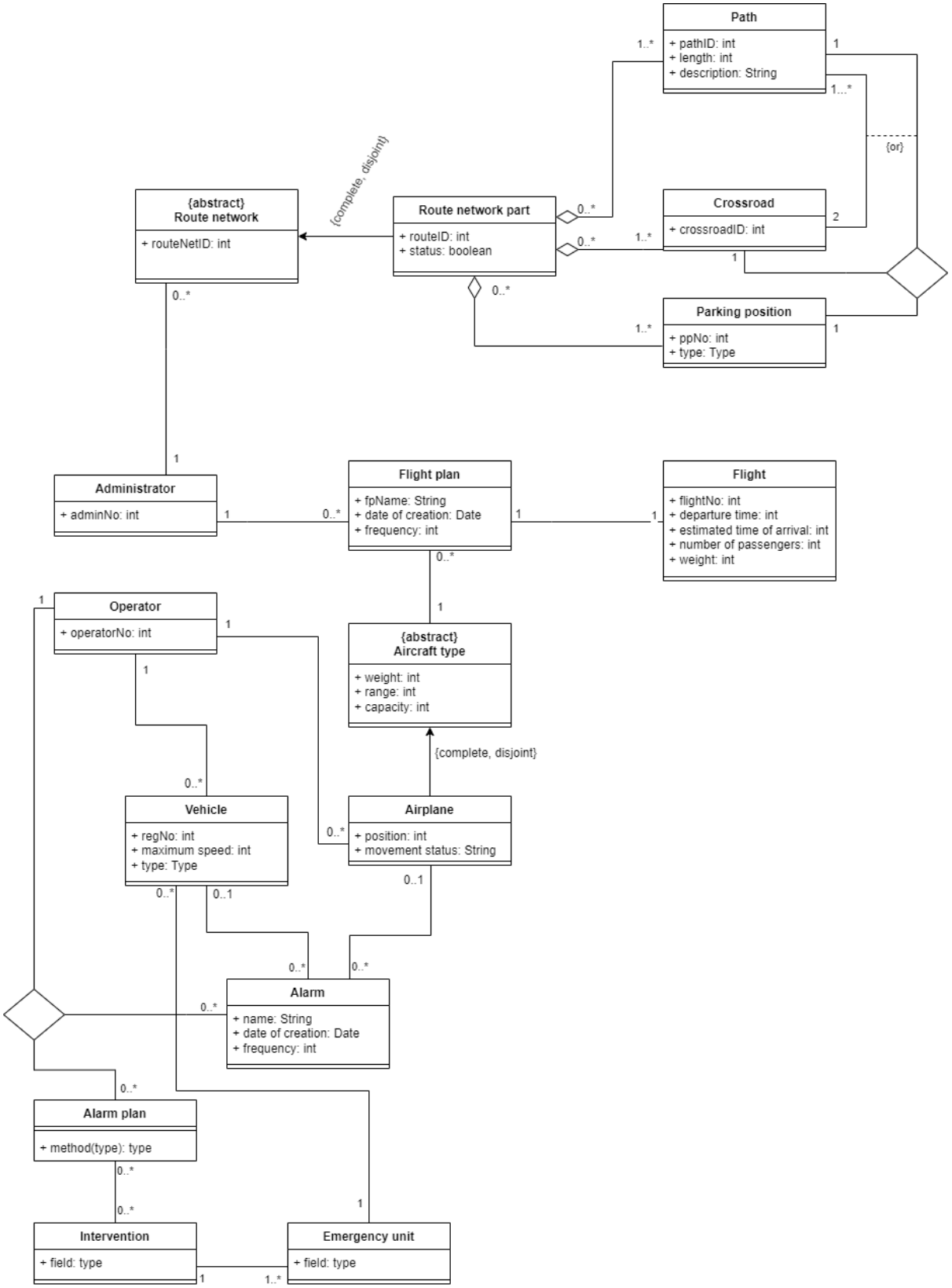
Exercise 8:


Delivery
+ deliveryNo: int
Order
+ purchOrderNo: int
+ date: Date
+ productNo: int
Customer
+ custNo: int
1..*
1
0..*
1
1
1..*
Delivery position
+ deliveryNo: int
1
1
Product
+ serialNo: int
0..*
1..*
1..*
Sub-Product
1..*
Developer
1..*
Designer
1..*
Product manager
{complete, disjoint}
{abstract} Employee
+ employeeNo: int

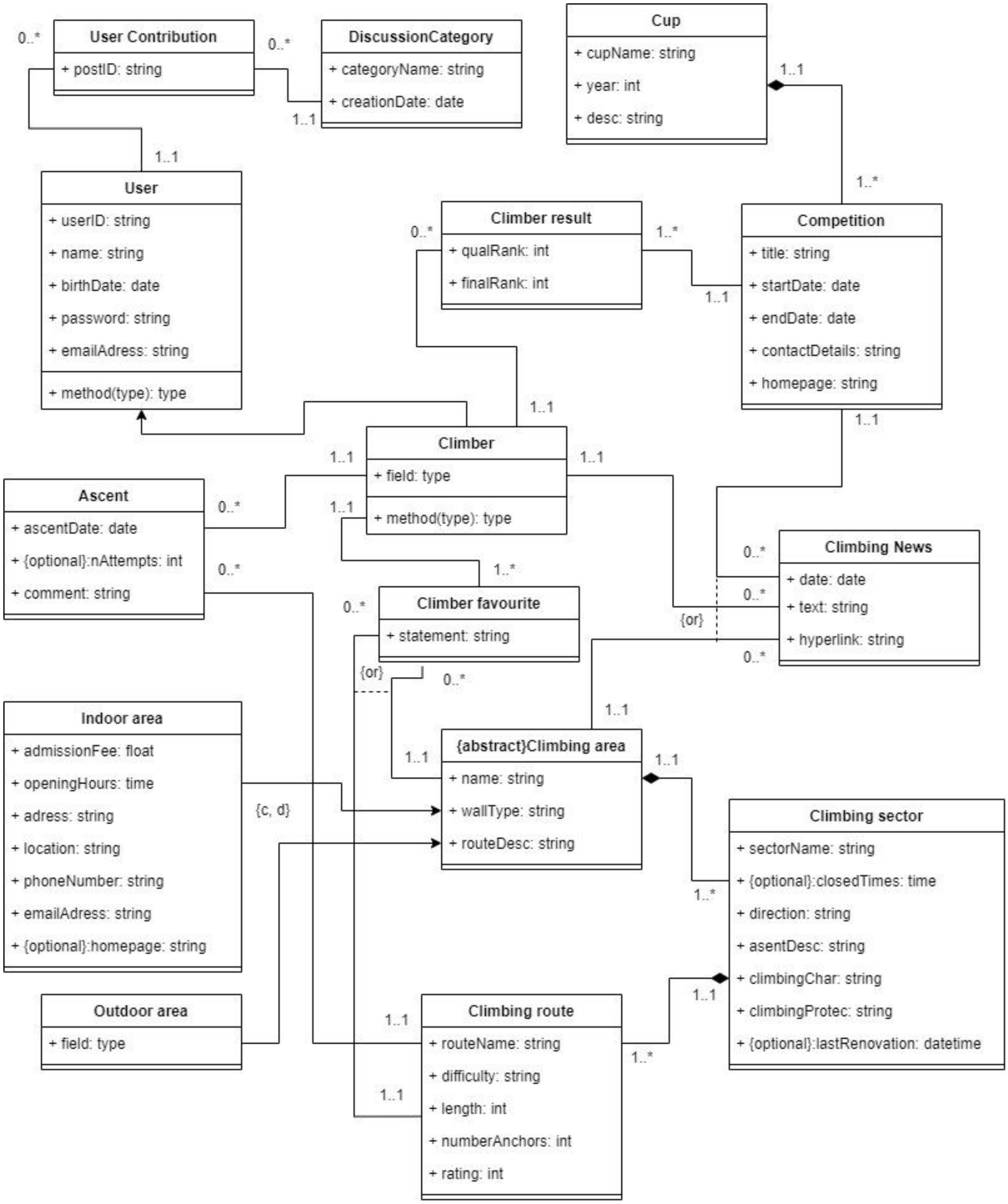
Exercise 9:


subject areas
Employee
name: String
address: Adress
email: String
orderList
book
location: Location
print Barcode()
incomplete, disjoint
subject editor
borrowing service
book service
Borrowing order
or
reservation order
book service
Library user
name: String
status: String
postal adress: Adress
email: String
phone number: int
borrowing system
search engine
print results()
entry in system
borrowing rights: list of enum
print ID barcode()
location
floor
room
shelf
borrowing register
print reminder()
reminder fees()

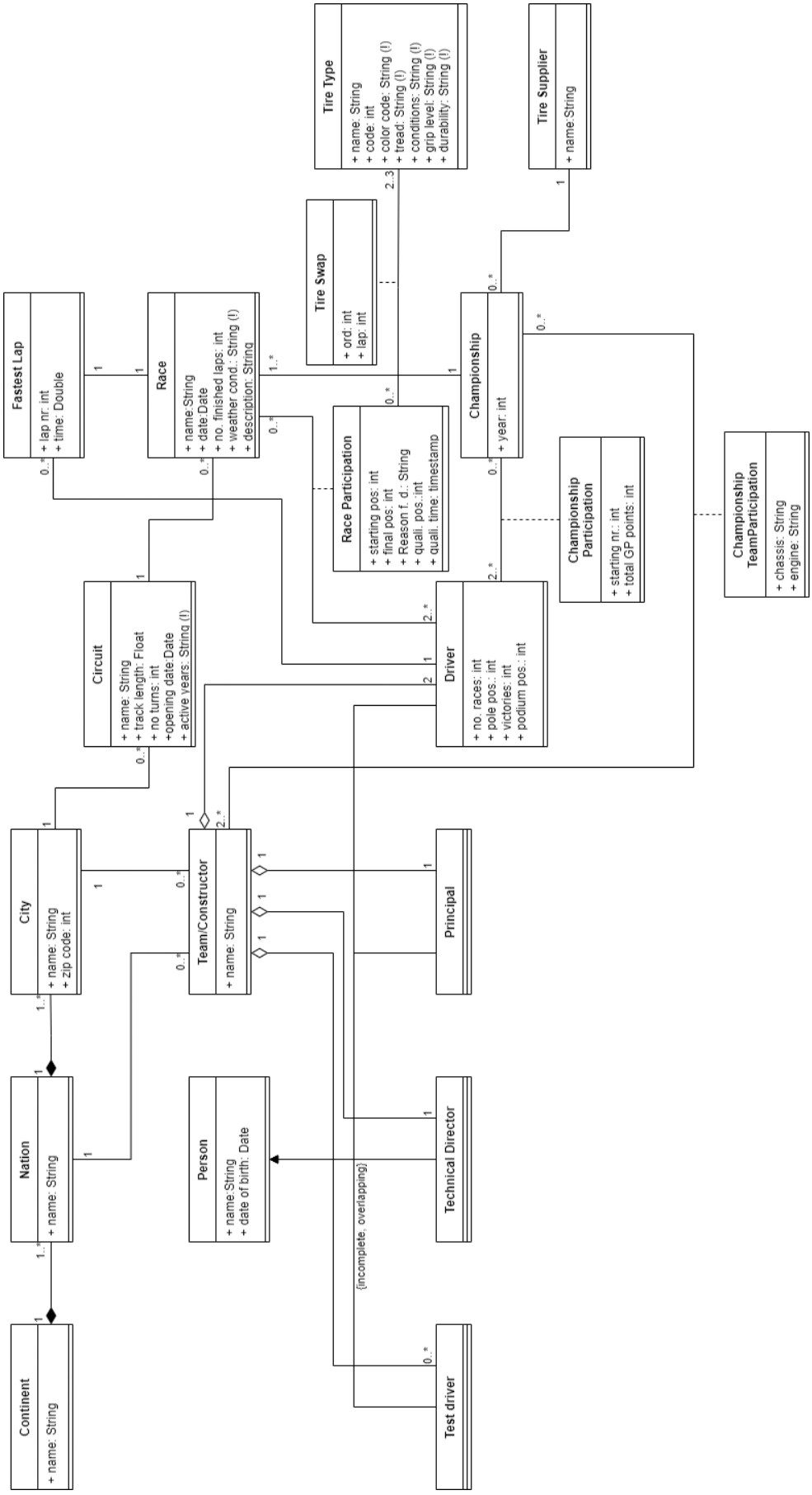
Exercise 10:



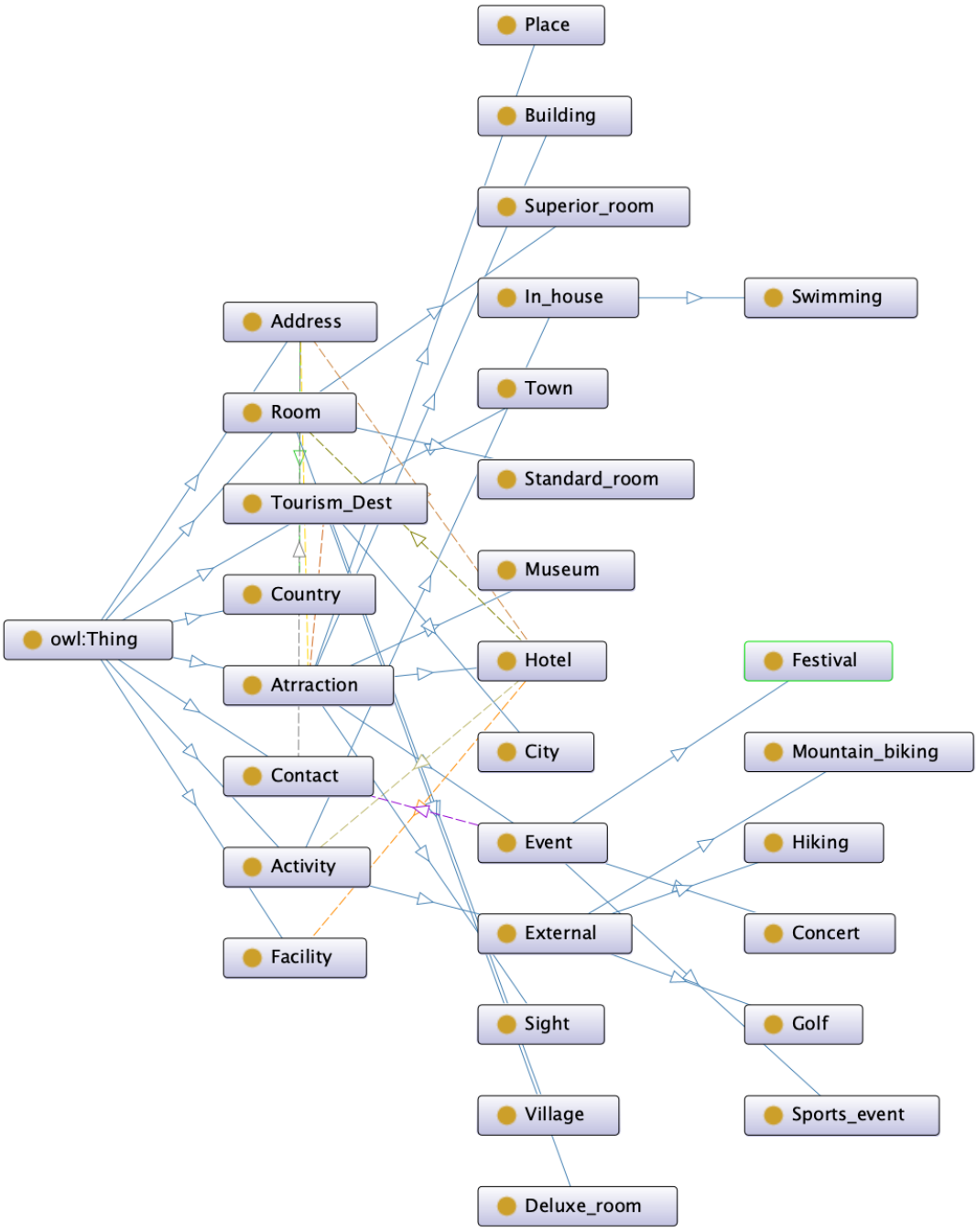
Exercise 11:


User Contribution
+ postID: string
DiscussionCategory
+ categoryName: string
+ creationDate: date
Cup
+ cupName: string
+ year: int
+ desc: string
User
+ userID: string
+ name: string
+ birthDate: date
+ password: string
+ emailAdress: string
+ method(type): type
Climber result
+ qualRank: int
+ finalRank: int
Competition
+ title: string
+ startDate: date
+ endDate: date
+ contactDetails: string
+ homepage: string
Climber
+ field: type
+ method(type): type
Ascent
+ ascentDate: date
+ {optional}:nAttempts: int
+ comment: string
Climber favourite
+ statement: string
Climbing News
+ date: date
+ text: string
+ hyperlink: string
{or}
Indoor area
+ admissionFee: float
+ openingHours: time
+ adress: string
+ location: string
+ phoneNumber: string
+ emailAdress: string
+ {optional}:homepage: string
{c, d}
{abstract}Climbing area
+ name: string
+ wallType: string
+ routeDesc: string
Climbing sector
+ sectorName: string
+ {optional}:closedTimes: time
+ direction: string
+ asentDesc: string
+ climbingChar: string
+ climbingProtec: string
+ {optional}:lastRenovation: datetime
Outdoor area
+ field: type
Climbing route
+ routeName: string
+ difficulty: string
+ length: int
+ numberAnchors: int
+ rating: int

Exercise 12:


Tire Type
+ name: String
+ code: int
+ color code: String (!)
+ tread: String (!)
+ conditions: String (!)
+ grip level: String (!)
+ durability: String (!)
Tire Supplier
+ name:String
Tire Swap
+ ord: int
+ lap: int
Fastest Lap
+ lap nr: int
+ time: Double
Race
+ name:String
+ date:Date
+ no. finished laps: int
+ weather cond.: String (!)
+ description: String
Championship
+ year: int
Race Participation
+ starting pos: int
+ final pos: int
+ Reason f. d.: String
+ quali. pos.: int
+ quali. time: timestamp
Championship Participation
+ starting nr.: int
+ total GP points: int
Championship TeamParticipation
+ chassis: String
+ engine: String
Circuit
+ name: String
+ track length: Float
+ no turns: int
+opening date:Date
+ active years: String (!)
Driver
+ no. races: int
+ pole pos.: int
+ victories: int
+ podium pos.: int
City
+ name: String
+ zip code: int
Team/Constructor
+ name: String
Principal
Nation
+ name: String
Person
+ name:String
+ date of birth: Date
Technical Director
Continent
+ name: String
Test driver
{incomplete, overlapping}

Exercise 13


Place
Building
Superior_room
In_house
Swimming
Address
Town
Room
Standard_room
Tourism_Dest
Museum
Country
owl:Thing
Hotel
Festival
Atrraction
City
Mountain_biking
Contact
Event
Hiking
Activity
External
Concert
Facility
Sight
Golf
Village
Sports_event
Deluxe_room