This assignment will use an SQL schema similar to the one in Assignment #3, except that some uniqueness constraints have been added.

create table Building (

id int primary key,

address varchar(500) not null unique

) engine=InnoDB;

create table Apartment (

id int primary key,

number varchar(31) not null,

building int not null references Building(id)

on update cascade on delete cascade,

unique(building, number)

) engine=InnoDB;

create table RoomType (

id int primary key,

description varchar(5000)

) engine=InnoDB;

create table RoomTypeName (

type int references RoomType(id)

on update cascade on delete cascade,

name varchar(255),

primary key(type, name)

) engine=InnoDB;

create table Room (

id int primary key,

area double not null,

type int not null references RoomType(id)

on update cascade,

apartment int not null references Apartment(id)

on update cascade on delete cascade

) engine=InnoDB;

create table ApartmentFloor (

apartment int references Apartment(id)

on update cascade on delete cascade,

floor int,

primary key(apartment, floor)

) engine=InnoDB;

create table Person (

id int primary key,

name varchar(5000) not null

) engine=InnoDB;

create table Owner (

person int references Person(id)

on update cascade on delete cascade,

apartment int references Apartment(id)

on update cascade on delete cascade

primary key(person, apartment)

) engine=InnoDB;

Design and develop persistable classes with JPA annotations that map to the schema above. Develop a main program that performs the following commands using the JPA query language, printing the results of queries to the console. Test your program using OpenJPA.

1. Change the owners of Apartment 2C in the building at 100 Main Street to George and Fred.
2. Find the addresses of all buildings that have a Living Room with area greater than 1000 s.f. owned by Fred in an apartment on the fifth floor or higher.
3. Find all apartments, showing the address and apartment number that have 3 or more rooms with the same type.
4. List the buildings in order by address together with the number of apartments in each building.
5. Task 1
6. em = factory.createEntityManager();
7. em.getTransaction().begin();
8. Set<Person> newOwners = new HashSet<Person>();
9. newOwners.add((Person) em.createQuery("select p from Person p where p.name='Fred'").getSingleResult());
10. newOwners.add((Person) em.createQuery("select p from Person p where p.name='George'").getSingleResult());
11. Query q = em.createQuery("select a from Apartment a where a.number = '2C' and a.building.address = '100 Main Street'");
12. for (Apartment a : (List<Apartment>) q.getResultList()) {
13. a.setOwners(newOwners);
14. }
15. em.getTransaction().commit();
16. em.close();
17. Task 2
18. em = factory.createEntityManager();
19. q = em.createQuery("select distinct r.apartment.building.address " +
20. "from Room r join r.type.names n join r.apartment.floors f join r.apartment.owners o " +
21. "where f >= 5 and n = 'Living Room' and o.name = 'Fred' and r.area > 1000");
22. for (String address : (List<String>) q.getResultList()) {
23. System.out.println(address);
24. }
25. em.close();
26. Task 3
27. em = factory.createEntityManager();
28. q = em.createQuery("select a.building.address, a.number from Apartment a where " +
29. "exists (select t from RoomType t where " +
30. "3 <= (select count(r) from Room r where r.type = t and r.apartment = a))");
31. for (Object[] l : (List<Object[]>) q.getResultList()) {
32. System.out.println(l[0] + " " + l[1]);
33. }
34. em.close();
35. Task 4
36. em = factory.createEntityManager();
37. q = em.createQuery("select a.building.address, count(a) from Apartment a group by a.building, a.building.address");
38. for (Object[] l : (List<Object[]>) q.getResultList()) {
39. System.out.println(l[0] + " " + l[1]);
40. }

em.close();