Name: Abderarore Bekeui StudentID: 613482.

SCI: 3.8
Total: 97.6

Theory Section

A. [3 pts] What are the 4 states of the entity life cycle?

3

- managed - removed
- Transient - detached.

B. [3 pts] Why are surrogate keys preferred over natural keys?

3

b/c they don't have any semantic meaning in our entity class so we can make sure that they are unique, constant, but if they are surrogate key it is not easy to manage them b/c they have meaning in our entity class so uniqueness is in question.

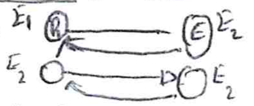
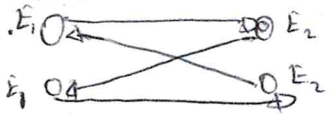
C. [3 pts] Explain what a sequence is in a database:

3

a sequence is a separate database object that provide next value, so we can use this sequence as one of generation strategy in our database.

D. [3 pts] Explain the difference between a bi-directional association and two uni-directional associations.

3

bi-directional:- ~~[illegible]~~ a given object is point to another object, & that obj is point back to this obj only. E_1 E_2  , two uni-directional:- E_1 ~~one~~ obj is point to another E_2 obj, that E_2 ~~[illegible]~~ ob[j] can point to any other obj of E_1. 

E. [3 pts] What does entitManager.flush() do?

3

- it will load all the changed ~~that~~ are made ~~[illegible]~~ to a given managed object in a persist[ence] context in to the database (it do implicit update)

F. [3 pts] What does the 'Extra' @LazyCollection do in terms of Hibernate optimization?

3

=> this Hibernate annotation will make a given collection ~~[illegible]~~ not to the collection loading ~~[illegible]~~ when we make a method like (collection.size(), the DB will itself to count and return that.

G. [3 pts] Explain how a version column can fix the Lost Update problem

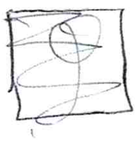
3

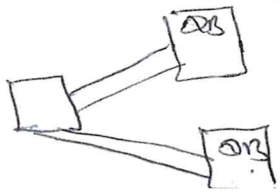
- when ever TX want to make update a given data it need to specify the version before and also update that version column after finish that. so first it compare version and if that match to what it specifies the update only made other wise it will be discarded.

H. [3 pts]Explain what a (XA) global transaction is:

3

~~extended XA~~ is a type of transaction with involves multiple transaction resources like database, message bus. so they need a transaction manager.



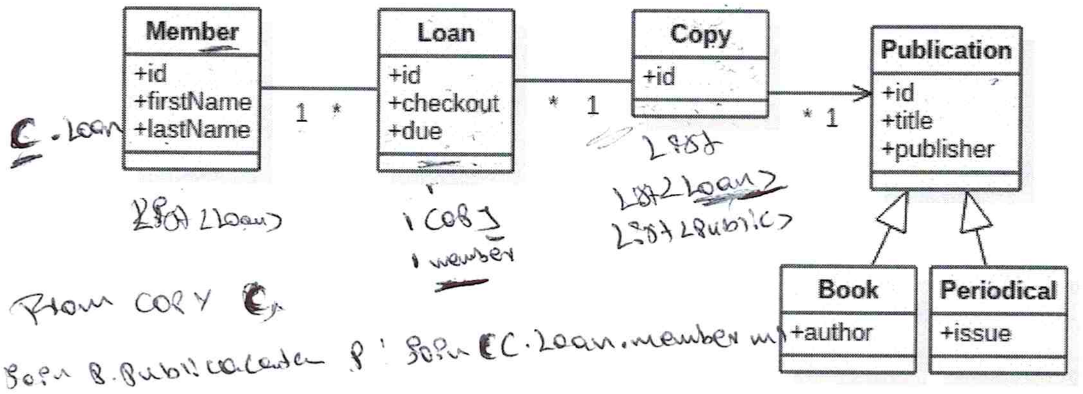


Name: Abderraore Bedemi StudentID: 613482.

These are UML diagrams of the domains used for the code exercises. I don't recommend using them for the mapping exercises (I may have forgotten to add or rename properties). They are meant for use with the JPQL queries to get an idea of how the different classes relate to each other.

The first domain is a Library domain, the second is a Restaurant domain

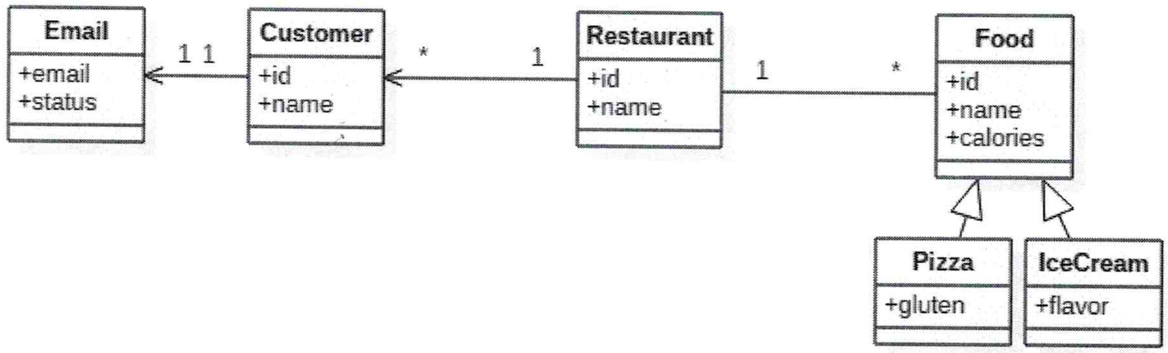
Hint about queries with dates: use the date directly in the string. For instance to get all loans that are due on 2022-01-23 write: from Loan l where l.due = '2022-01-23'



From Copy C,

Join p.publication p' Join C.Loan.member m

where p type(p) = — and p.title = " ").



" select distinct m from member m'

' Join m.loans L ~~copy~~' Join L.copys.publication p' ~~Join L.copy.publication~~

' where. C ~~publication~~

p.title = "Communications of the ACM" and type(p) = periodical

1) ' select L.member from ~~Loan~~ copy C '

+ ' ~~Join L.copy~~ C '

' Join C.publication p' where

p.title = " .... "

and type(p) = 'periodical').

select (C) from food F' where type(F) = pizza

and f.name = 'californian' and F.Restaurant

.name = 'revelesh

~~sele~~

from customer C where

C.Email.email like '% gmail.com

F.Restaurant.

Join F.Restaurant.customers as C

Name: Abedenelore Dedeman StudentID: 613482

Exercises:

1. [24 pts] Based on the following classes with annotations write what the tables names, column names, and data types will be (also include if a column is auto_increment).

```
@Entity
public class Member {
    @Id
    @GeneratedValue
    private Integer id;
    @Column(name="given")
    private String firstName;
    @Column(name="family")
    private String lastName;
    @OneToMany(mappedBy="member")
    private List<Loan> loans
        = new ArrayList<>();
}
@Entity
public class Loan {
    @Id
    @GeneratedValue
    private Long id;
    @ManyToOne
    private Member member;
    @ManyToOne
    private Copy copy;
    @Temporal(TemporalType.DATE)
    private Date checkout;
    @Temporal(TemporalType.DATE)
    private Date due;
    @Temporal(TemporalType.DATE)
    private Date returned;
}
@Entity
public class Copy {
    @Id
    @GeneratedValue
    private Long id;
    @OneToMany(mappedBy = "copy")
    private List<Loan> loans
        = new ArrayList<>();
    @ManyToOne
    private Publication publication;
}
```

```
@Entity
@Inheritance(strategy =
        InheritanceType.JOINED)
public abstract class Publication {
    @Id
    @GeneratedValue
    private Long id;
    private String title;
    private String publisher;
    @Lob
    private String text;
}
@Entity
public class Book extends Publication {
    private String author;
}
@Entity(name = "Magazine")
public class Periodical extends Publication {
    private String issue;
}
```

Member

id - int - autoincrement
given - varchar(255)
family - varchar(255)

Copy

id - bigint(20) - auto
publication_id - bigint

Loan

id - bigint - auto
member_id - int
copy_id - bigint
due - DATE
checkout - DATE
returned - DATE

Book

id - bigint
author - varchar

Magazine

id - bigint
issue - varchar

Publication

id - bigint - auto
title - varchar
publisher - varchar
text - clob

24

Name: Abedeperzore Bedemi StudentID: 613482

2. [24 pts] Add annotations to the following classes to map to the tables shown on the next page.

```
@Entity

public class Customer {

    @ID
    @GeneratedValue
    private Long id;                                  ✓


    private String name;

    @Embeded
    private Email mail;
}

@Embeddable
public class Email {

                                                      ✓
    private String email;


    private String status;
}

@Entity  ✗
public class Restaurant {
    @Id
    @GeneratedValue
    private Integer id;


    private String name;
    @OneToMany ~~Em~~
    @
                                                      ✓
    private List<Customer> customers =
        new ArrayList<>();
    @OneToMany(mappedBy="restaurant")
    @~~JoinColumn(name="restaurant_id")~~
    private List<Food> foods =
        new ArrayList<>();
}
```

```
@Entity
@Inheritance(strategy=InheritanceType
                                        .TABLE_PER
public abstract class Food {             _CONCRETE
    @Id                                  TABLE
    @GeneratedValue(GenerationType.TABLE)
    private Long id;
                                                  ✓

    private String name;

    @Column(name="cals")
    private int calories;
    @~~JoinColumn(name="diner_id")~~
    @~~ManyToOne~~
    private Restaurant restaurant;
}

@Entity
public class Pizza extends Food {

                                                  ✓
    private boolean gluten;
}

@Entity
public class IceCream extends Food {

                                                  ✓
    private String flavor;
}
```

23

member.

id - int - auto
given - varchar
family - varchar

Loan.

id - Long int - auto.
member_id - ~~[illegible]~~ int
Copy_id - Long int
~~Date - DATE~~
due - DATE
checkout - DATE
returned - DATE

Book

id - Long int.
author - varchar.

Copy

id - Long - Auto
Publication_id - ~~Long~~ int.

Magazine

id - Long int
issue - varchar.

Publication

- ~~Long~~ id - bigint auto
title - varchar
Publisher - varchar
text - CLOB.

Name:____ StudentID:____________

```
describe Customer;
+--------+--------------+------+-----+---------+----------------+
| Field  | Type         | Null | Key | Default | Extra          |
+--------+--------------+------+-----+---------+----------------+
| id     | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| email  | varchar(255) | YES  |     | NULL    |                |
| status | varchar(255) | YES  |     | NULL    |                |
| name   | varchar(255) | YES  |     | NULL    |                |
+--------+--------------+------+-----+---------+----------------+
describe Restaurant_Customer;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| Restaurant_id | int(11)    | NO   | MUL | NULL    |       |
| customers_id  | bigint(20) | NO   | PRI | NULL    |       |
+---------------+------------+------+-----+---------+-------+
describe Restaurant;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
describe hibernate_sequences;
+-----------------------+--------------+------+-----+---------+-------+
| Field                 | Type         | Null | Key | Default | Extra |
+-----------------------+--------------+------+-----+---------+-------+
| sequence_name         | varchar(255) | NO   | PRI | NULL    |       |
| sequence_next_hi_value | bigint(20)  | YES  |     | NULL    |       |
+-----------------------+--------------+------+-----+---------+-------+
describe Pizza;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | bigint(20)   | NO   | PRI | NULL    |       |
| cals     | int(11)      | YES  |     | NULL    |       |
| name     | varchar(255) | YES  |     | NULL    |       |
| diner_id | int(11)      | YES  | MUL | NULL    |       |
| gluten   | bit(1)       | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
describe IceCream;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | bigint(20)   | NO   | PRI | NULL    |       |
| cals     | int(11)      | YES  |     | NULL    |       |
| name     | varchar(255) | YES  |     | NULL    |       |
| diner_id | int(11)      | YES  | MUL | NULL    |       |
| flavor   | varchar(255) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

Name: Abebe [illegible] Ketema StudentID: 613482

3. [12 pts] Based on the library domain write the following queries.
   a. All members who have a loan that is due on the 23rd of January 2022

4

```
' Select distinct m from member m'
+ ' Join m.Loans L where L.due = '2022-01-23'
```

   b. All copies of the book with title "Dune"

3

```
~~'from copy c where c.publication.title = "Dune"'~~
' Select distinct c from copy c'
+ ' Join c.publications p where p.title = "Dune" }   and type(p) = Book
```

   c. All members who checked out the periodical titled "Communications of the ACM"

3.9

```
' Select distinct m from copy c'
' Join c.publications p' , ' Join c.loan.member m'
' where type(p) = ~~Periodical~~ Magazine and p.title = "Communication of the ACM"'
```

4. [12 pts] Based on the restaurant domain write the following queries.
   a. All Customers whose email address ends in 'gmail.com'

3.9

```
from customer c where c.~~email~~ mail.email like '% gmail.com'
```

   b. All Customers who visited the restaurant "India Cafe"

4

```
' ~~from~~ Select distinct ~~C~~ from Restaurant R '
' Join R.customers C where R.name = "India cafe '
```

   c. All Customers who ate the pizza with name'Californian' at the restaurant 'Revelations'

4

```
' Select c from Food f'
+ ' Join f.Restaurant.customers c where type(f) = pizza and
' f.name = 'californian' + and ' f.Restaurant.name = 'Revelations'
```