Name: LUCY TURIHABWE StudentID: 613439

SCI: 4
Total: 95.5

Theory Section

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

3

Transient Detached
Managed Removed

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

3

Surrogate keys are preferred over natural keys because they're consistent - they never change as opposed to natural keys. They also don't have any meanings to the business domain.

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

3

A sequence is a separate database object that provides next values.
It can be used as identity source by multiple tables, ensuring unique ID columns with unique values even when these tables are combined into single view.
Database such as Oracle, Postgresql use a sequence.

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

3

2 unidirectional associations. unidirection manyToOne uses foreign key and unidirectional @OneToMany uses JoinTable. It happens when you don't put mappedBy on @ManyToOne annotation therefore 2 different associations are created
* bi-directional association is what is wanted and its achieved by adding mappedBy on @OneToMany and @JoinTable on @ManyToOne

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

3

entityManager.flush() ~~does~~ ~~the~~ explicitly pushes changes to the database
It only works on managed entities.

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

3

@LazyCollection is useful for big collections. By default, the entire collection is retrieved for operations such as .size(), .isEmpty(), .contains(). But instead of using the database for such operations, extra lazy fixes that.

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

3

Adding a version column aids in tracking updates and ensures that the first update wins. the last update fails hence solving the Lost Update Problem.

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

3

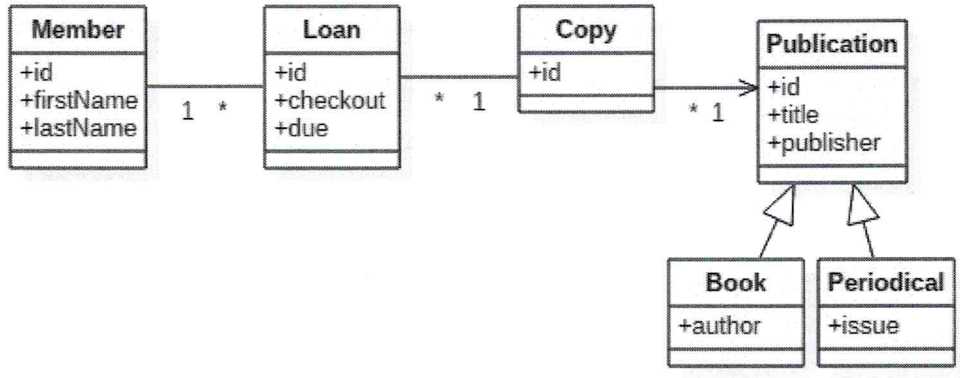
A global transaction is one that spans multiple transaction resources.
It's done by a transaction manager. It's enabled by a two-phase commit and transaction resources become dependent on each other.

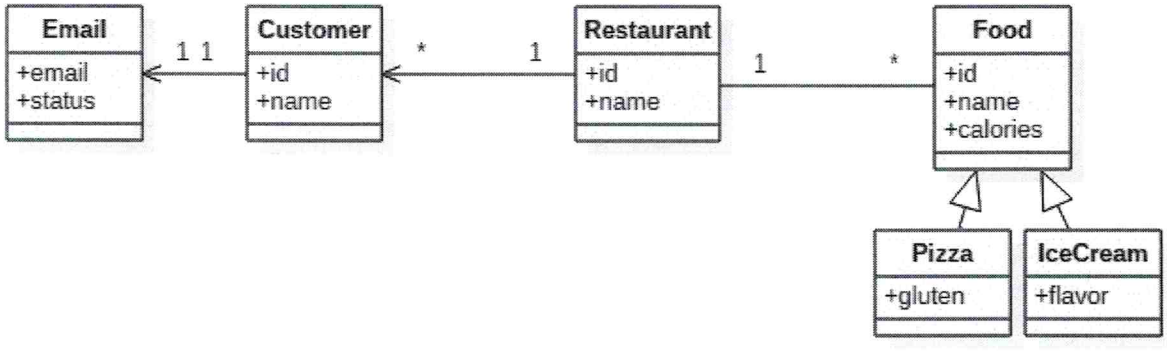
Name: LUCY TURIHABWE StudentID: 613439

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'





Name: Lucy Turihabwe StudentID: 613439

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;
}
```

Table name: Member

Column name	Type	Key	Extra
id	int(11)	PRI	auto_increment
given	varchar		
family	varchar		

Table name: Loan

Column name	Type	Key	Extra
id	bigint(20)	PRI	auto_increment
member_id	int(11)	MUL	
copy_id	bigint(20)	MUL	
checkout	date		
due	date		
returned	date		

Table name : Copy

Column name	Type	Key	Extra
id	bigint(20)	PRI	auto-increment
publication_id	bigint(20)	MUL	

Table name : Publication

Column name	Type	Key	Extra
id	bigint(20)	PRI	auto-increment
title	varchar(255)		
publisher	varchar(255)		
text	lob		

Table name: Magazine

Column name	Type	Key	Extra
publication_id	bigint(20)	MUL	
issue	varchar(255)		

Table name : Book

Column name	Type	Key	Extra
publication_id	bigint(20)	MUL	
author	varchar(255)		

Name: LUCY TURIHABWE StudentID: 613439

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(strategy=GenerationType.SEQUENCE)
    private Long id;

    private String name;

    @Embedded
    @Column(name="email")
    private Email mail;
    @Embedded
    private String status;
}

@Embeddable
public class Email {

    private String email;

    private String status;
}

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

    private String name;

    @OneToMany
    private List<Customer> customers =
        new ArrayList<>();

    @OneToMany(mappedBy="restaurant")
    private List<Food> foods =
        new ArrayList<>();
}
```

```
@Entity
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS)
public abstract class Food {

    @Id
    private Long id;

    private String name;

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

@Entity
public class Pizza extends Food {

    private boolean gluten;
}

@Entity
public class IceCream extends Food {

    private String flavor;
}
```

22

Name: LUCY TURIHABWE StudentID: 613439

```
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    |       |
+----------+--------------+------+-----+---------+-------+
```

TABLE PER CLASS

Name: LUCY TURIHABWE StudentID: 613439

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 as l where l.due
  = '2022-01-23'"
```

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

4

```
"select c from Copy c where c.publication.title = 'Dune'"
```

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

3

```
"select distinct m from Member m join m.loans as l join l.copy as c
  join c.publication as p where p.title = 'Communications of the ACM'"
  and type(p) = Magazine
```

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

3

```
"select c from Customer c where c.mail like '%gmail.com'"
                                              .email
```

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

4

```
"select r.customers from Restaurant r where r.name = 'India Cafe'"
```

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

3.5

```
"select distinct r.customers from Restaurant r join r.foods as f where r.name
  = 'Revelations' and f.class = Pizza"
    and f.name = 'Californian'
```