

Chương 4: JPA - HIBERNATE

Phần 1: Object-relational mapping (ORM)

Resource: <https://hibernate.org/orm/releases/6.1/>

1. Overview

Overview

- JPA – Java Persistence API - có thể lưu trữ, thao tác dữ liệu giữa Java Objects và ngược lại.
- Hibernate là một implementation phổ biến của JPA, là một công cụ ORM mã nguồn mở và nhẹ được sử dụng để lưu trữ, thao tác và truy xuất dữ liệu từ cơ sở dữ liệu.
- ORM (Object/Relational mapping). Là chiến lược lập trình để ánh xạ các POJO để có thể lưu trữ trong database, nó đơn giản hóa việc tạo, thao tác và xử lý dữ liệu.

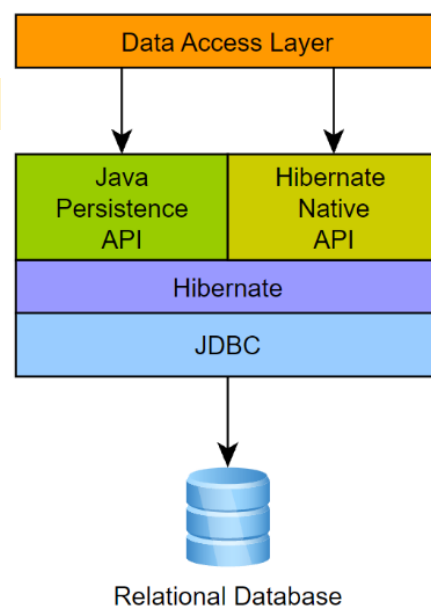
Deployment tools

- Hibernate 6.0 and later versions require at least Java 11 and JDBC 4.2
- JPA2.2
- Eclipse IDE for Java EE developers
- Hibernate release 6.1.1 final
- MS SQLSERVER

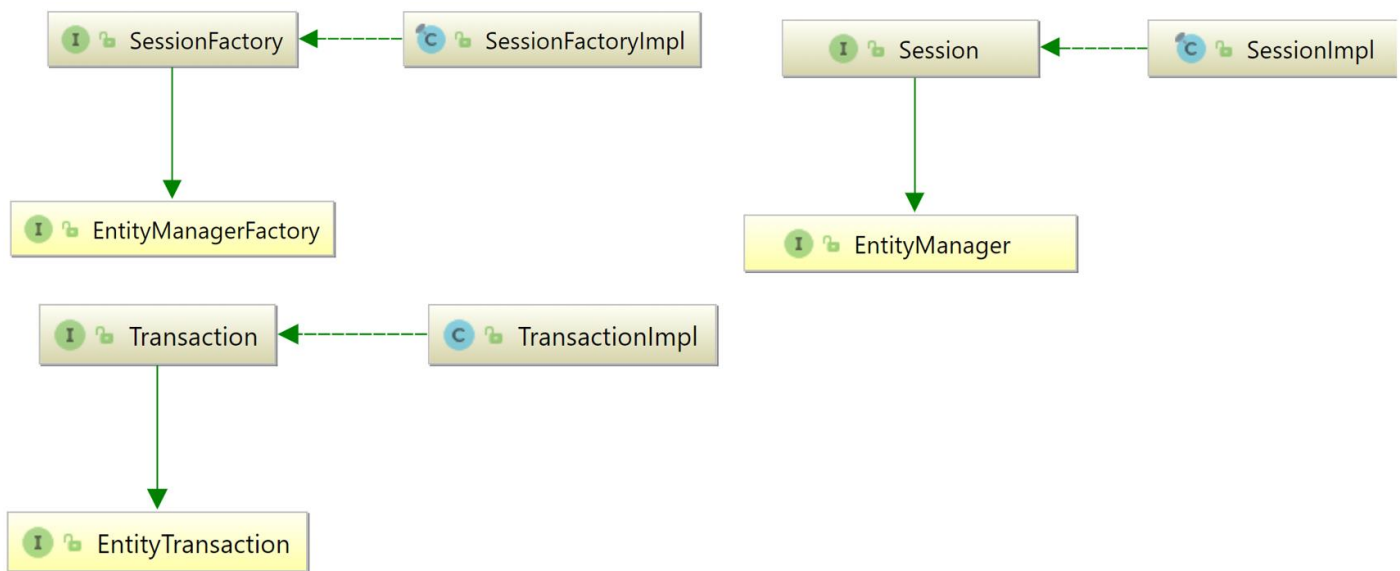
Installation

```
<dependencies>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.1.1.Final</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->
  <dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>9.4.0.jre16</version>
  </dependency>
</dependencies>
```

2. Architecture



- Ứng dụng Java sử dụng Hibernate APIs để nạp, lưu trữ, truy vấn, ... dữ liệu.
- Hibernate triển khai các đặc tả Java Persistence API; và sự kết hợp giữa các đặc tả JPA với các triển khai riêng của Hibernate có thể được hình dung trong sơ đồ sau:



SessionFactory (org.hibernate.SessionFactory)

- Implement từ Factory Design Pattern, mục đích dùng để tạo ra các đối tượng Session. Thông thường, một ứng dụng chỉ có duy nhất một đối tượng SessionFactory.
- Phương thức thường dùng:
 - + openSession() → tạo ra một đối tượng Session
 - + hoặc getCurrentSession() → tạo ra một đối tượng Session nếu chưa tồn tại.
 - + close() → Giải phóng tất cả tài nguyên của SessionFactory

Session (org.hibernate.Session)

- Là main runtime interface, duy trì kết nối vật lý với database. Nó cung cấp các phương thức cho các thao tác thêm, xóa, sửa... dữ liệu như save(), update(), delete(), load(), get() ...
- Vòng đời của một Session được giới hạn từ việc bắt đầu cho đến kết thúc một logical transaction (*begin transaction ... commit transaction*)

Transaction (org.hibernate.Transaction)

- Là một short-lived object, được dùng để nhóm các tác vụ thành một đơn vị atomic.
- Một transaction được gọi là thành công nếu nó được committed (*hoàn thành tất cả các tác vụ bên trong*)
- Nếu có bất kỳ tác vụ nào trong transaction không hoàn thành, quay lui về trạng thái bắt đầu của transaction là rollback

3. Bootstrap

Bootstrap –Thuật ngữ đề cập đến vấn đề khởi tạo và khởi động một software component .

JDBC Driver – Một software component cho phép Java Application có thể tương tác với database.

Dialect: - Dạng biến thể của ngôn ngữ (SQL), mỗi database đều có 2 phần liên quan đến SQL: một phần là native của database và một phần là của SQL chuẩn.

- Dialect cho phép Hibernate tạo ra các câu lệnh SQL tương thích với database mà đã chọn.

URL – Đường link cấu hình để kết nối đến database. Trong Hibernate ORM, URL chính là JDBC URL.

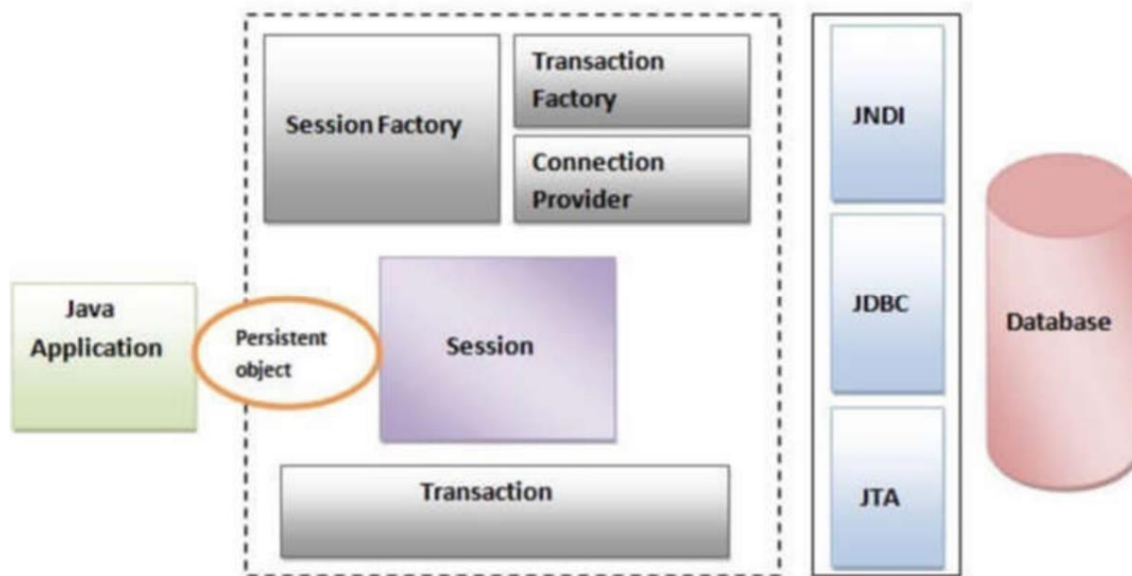
Show-sql - Hiển thị các câu DDL được tạo ra từ Hibernate.

Hbm2dll - Liên quan đến việc ta tạo, chỉnh sửa schema mà ta sử dụng trong database được ánh xạ bởi hibernate.

3.1 Native Bootstrapping

- Tạo ServiceRegistry chứa các dịch vụ mà Hibernate cần cho quá trình khởi động (*bootstrapping*) và quá trình thực thi (*run time*).

- Có 2 ServiceRegistries: `org.hibernate.boot.registry.BootstrapServiceRegistry` và `org.hibernate.boot.registry.StandardServiceRegistry`
- Tạo Metadata (`org.hibernate.boot.Metadata`) chứa các domain model mà sẽ ánh xạ sang database.
- Một số phương thức thường quan tâm của `org.hibernate.boot.MetadataSources`: `addAnnotatedClass`, `addAnnotatedClassName`, `addResource` ...
- Tạo SessionFactory – Dùng để tạo các đối tượng Session
- **Native Hibernate Architecture:**



- **Configuration file:** Tạo file `hibernate.properties` hoặc `hibernate.cfg.xml` hoặc viết java code để cấu hình. Thường dùng file XML để cấu hình, và file cấu hình đặt trong `java/main/resources`.

- **Hibernate.properties configuration file**

```
hibernate.connection.driver_class=com.microsoft.sqlserver.jdbc.SQLServerDriver
hibernate.connection.url=jdbc:sqlserver://localhost:1433;databaseName=test
hibernate.connection.username=sa
hibernate.connection.password=sapassword
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
hibernate.show_sql=true
hibernate.format_sql=true
hibernate.hbm2ddl.auto=update
hibernate.current_session_context_class=thread
```

- **Hibernate.cfg.xml configuration file**

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <property name="connection.driver_class">
```

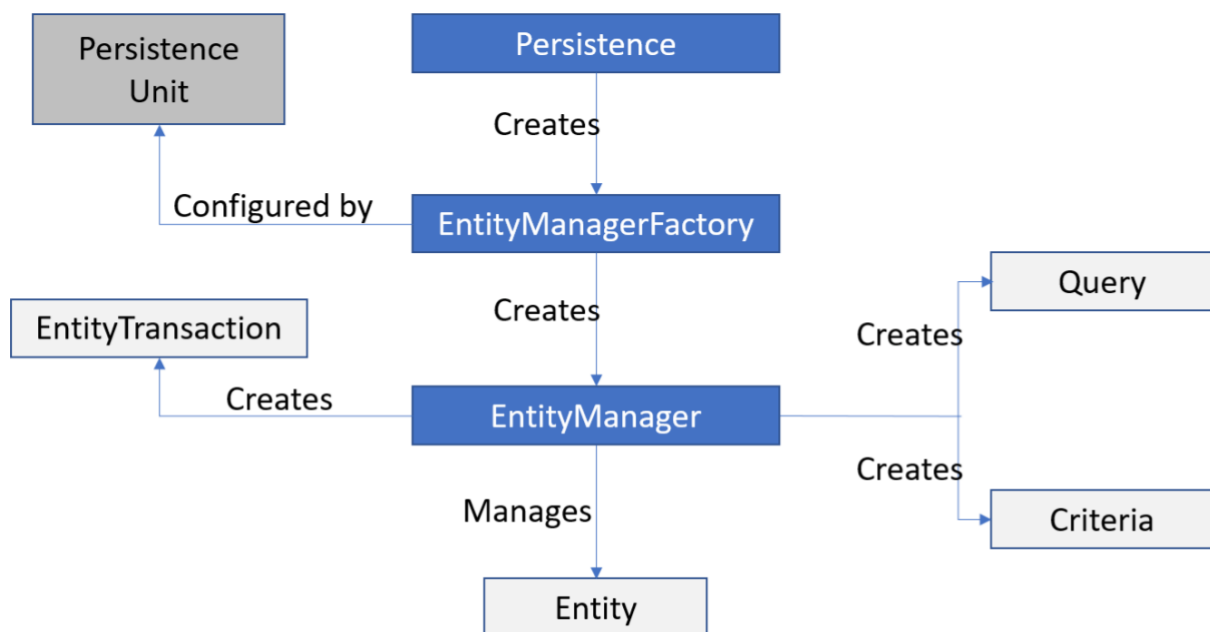
```

        com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
<property name="connection.url">
    jdbc:sqlserver://localhost:1433;databaseName=test</property>
<property name="connection.username">sa</property>
<property name="connection.password">sapassword</property>
<!-- SQL dialect -->
<property name="dialect">org.hibernate.dialect.SQLServer2012Dialect</property>
<!-- Enable Hibernate's automatic session context management -->
<property name="current_session_context_class">thread</property>
<!-- Echo all executed SQL to stdout -->
<property name="show_sql">true</property>
<!-- Drop and re-create the database schema on startup -->
<property name="hbm2ddl.auto">create-drop</property>
</session-factory>
</hibernate-configuration>

```

3.2 JPA Bootstrapping

- Đặc tả JPA có 2 cách tiếp cận chính để tạo đối tượng javax.persistence.EntityManager từ javax.persistence.EntityManagerFactory, đó là: Java EE và Java SE.
- Container-bootstrapping (EE) – Container sẽ tạo EntityManagerFactory cho mỗi persistent-unit được cấu hình trong META-INF/persistence.xml.
- Application-bootstrapping (SE) - Ứng dụng tạo javax.persistence.EntityManagerFactory thông qua phương thức createEntityManagerFactory của lớp bootstrap javax.persistence.Persistence.
- EntityManager quản lý các entity, cung cấp các phương thức cho các thao tác thêm, xóa, sửa và tìm kiếm dữ liệu như persist, merge, remove, find ...
- **JPA Architecture:**



- **META-INF/persistence.xml configuration file**

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"

```

```

xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
<persistence-unit name="JPA_Bootstrapping">
<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
<properties>
    <property name="javax.persistence.jdbc.driver"
        value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />

    <property name="javax.persistence.jdbc.dialect"
        value="org.hibernate.dialect.SQLServer2012Dialect" />

    <property name="javax.persistence.jdbc.url"
        value="jdbc:sqlserver://localhost:1433;databaseName=qlsv" />

    <property name="javax.persistence.jdbc.user" value="sa" />
    <property name="javax.persistence.jdbc.password" value="sapassword" />
    <property name="hibernate.show_sql" value="true" />
    <property name="hibernate.hbm2ddl.auto" value="update" />
</properties>

</persistence-unit>
</persistence>

```

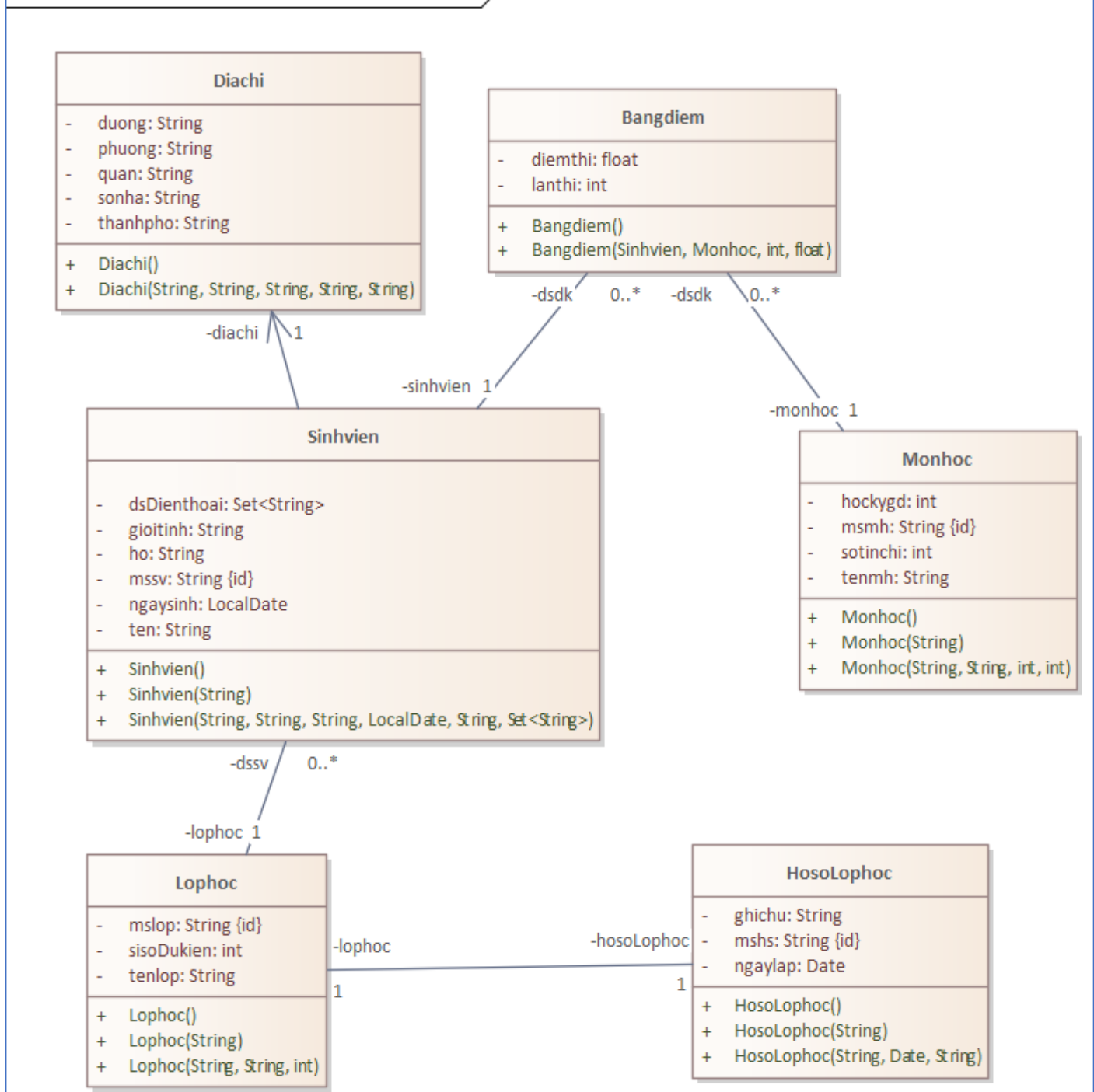
– **Cách tạo JPA project:**

Tạo một Maven Project → Click phải trên project → Properties → Project Facets → (*May be:* Convert to faceted from...) → check vào JPA → Further configuration available ... → JPA Implementation: Disable Library Configuration → OK → Apply and Close
Hoặc tạo JPA Project, sau đó Convert to Maven Project

4 Bài tập minh họa

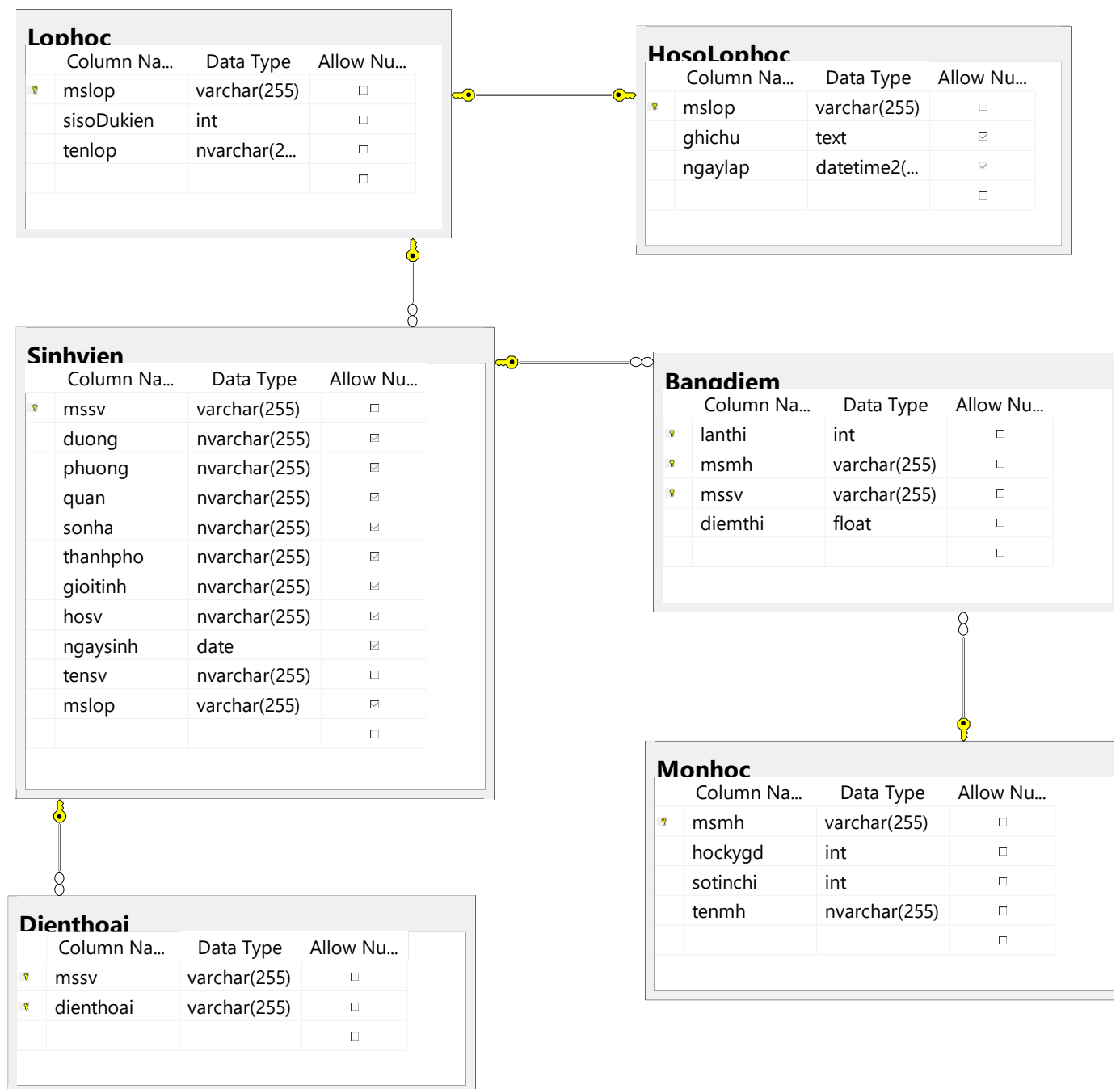
Cho sơ đồ lớp sau:

class Basic Class Diagram with Attributes and Operations



Yêu cầu:

- Ánh xạ mô hình lớp trên sang mô hình CSDL quan hệ tương ứng.



- (Java Programming) Viết các phương thức với các thao tác CRUD
 - a) Thực hiện các thao tác Create, Update, Delete, Find by Id trên từng đối tượng.
 - b) Tính số sinh viên theo từng lớp học.
 - + getSisoByLophoc() : Map<Lophoc, Integer>
 - c) Tính điểm trung bình của các môn học của các sinh viên
 - + listSinhvienDiemTB() : Map<Sinhvien, Float>
 - d) Những lớp học chưa có sinh viên
 - + listLophocNull() : List<Lophoc>
 - e) Những sinh viên học môn “Lập Trình Phân tán với Công Nghệ Java” có điểm cao nhất.
 - + listSinhvienGioiJava(): List<Sinhvien>

Hướng dẫn

Viết các lớp entity (chưa xét tới mối liên kết)

Các lớp entity phải:

- Có default constructor,
- Khai báo @Entity và @Id,
- May be Serializable

Lớp thực thể có chứa thuộc tính đa trị

- Cơ sở dữ liệu quan hệ sẽ tách ra thành một bảng mới, khóa chính của bảng mới là khóa chính của bảng ban đầu và thuộc tính đa trị.
- Khai báo @ElementCollection cho thuộc tính đa trị.
- Dùng @CollectionTable để mô tả thông tin cho bảng mới sẽ được tách ra.

Ví dụ:

@ElementCollection

@CollectionTable(

name = "Dienthoai",

joinColumns = @JoinColumn(name="mssv"),

uniqueConstraints = @UniqueConstraint(columnNames = {"mssv", "dienthoai"}))

)

@Column(name = "dienthoai", nullable = false)

private Set<String> dienthoais = new HashSet<String>();

Mapping entity associations/ relationships

fetch = FetchType.LAZY or fetch = FetchType.EAGER ????

By default:

OneToMany: LAZY

ManyToOne: EAGER

ManyToMany: LAZY

OneToOne: EAGER

4.1 Mối quan hệ 1 - 1 (One to One)

- Phải xác định được đối tượng chủ thể và đối tượng phụ thuộc, cả 2 bên đều khai báo @OneToOne
- Bên chủ thể thêm @PrimaryKeyJoinColumn
- Bên phụ thuộc thêm:
 - @MapsId
 - @JoinColumn với thuộc tính name

4.2 Mối quan hệ kế thừa (Inheritance relationship)

- Lớp cha:
 - @Entity
 - @Inheritance (strategy = InheritanceType.JOINED)
- Lớp con:
 - @Entity
 - @PrimaryKeyJoinColumn

4.3 Mỗi quan hệ 1 – n (*One to Many hoặc Many to One*)

- Phía One: Thêm @OneToMany với thuộc tính mappedBy.
- Phía Many thêm:
 - @ManyToOne
 - @JoinColumn với thuộc tính name

Note: Có thể thêm cascade = CascadeType.ALL (PERSIST, REMOVE, MERGE)

4.4 Mỗi quan hệ n – n (*Many to Many*)

Nguyên tắc đối với mỗi quan hệ Many to Many, cơ sở dữ liệu quan hệ sẽ tách thêm 1 thực thể mới (2 mỗi quan hệ One to Many). Khóa của thực thể mới chứa ít nhất 2 khóa của 2 thực thể tham gia vào mối liên kết này.

4.4.1 Dùng @ManyToMany

- Cả 2 bên điều khai báo @ManyToMany
- Bên chủ thể với thuộc tính mappedBy
- Bên còn lại thêm @JoinTable, với các thuộc tính:
 - name = "Table_Name",
 - joinColumns = @JoinColumn(name = "Column_name"),
 - inverseJoinColumns = @JoinColumn(name = "Column_name")

4.4.2 Tách thành 2 mỗi quan hệ @OneToMany (@IdClass)

- Lớp thực thể mới tách thêm:
 - Khai báo @Entity và @IdClass
 - Id của lớp là một Composite key.
 - Gồm ít nhất 2 field là khóa của 2 lớp thực thể tham gia vào mối liên kết.
 - Mỗi field khai báo @Id, @ManyToOne và @JoinColumn
 - Có thể có thêm các thuộc tính của mối liên kết
- Lớp mô tả composite key: các thuộc tính của lớp này là các thuộc tính tham gia vào khóa.
 - Khai báo @Embeddable
 - Phải có hashCode & equals,
 - Phải có Default constructor
 - Phải implement interface Serializable

4.4.3 Tách thành 2 mỗi quan hệ @OneToMany (@Embeddable)

- Lớp thực thể mới tách thêm:
 - Khai báo @Entity
 - Khai báo khóa Composite key: @Id hoặc @Embeddable
 - Có thể có thêm các thuộc tính của mối liên kết
- Lớp mô tả composite key: Như phần 4.4.2 trên
- Thuộc tính mappedBy của 2 mỗi quan hệ One to Many:
 - @OneToMany(mappedBy="pk.attributeKey")



Lophoc.java

@Entity

```
public class Lophoc {

    @Id
    private String mslop;
    @Column(name = "tenlop", nullable = false, columnDefinition = "nvarchar(255)")
    private String tenlop;

    private int sisoDukien;

    @OneToMany(mappedBy = "lophoc")
    private List<Sinhvien> dssv;

    @OneToOne
    @PrimaryKeyJoinColumn
    private Hosolophoc hosolophoc;

    public Lophoc() {
    }

    public Lophoc(String mslop, String tenlop, int sisoDukien) {
        super();
        this.mslop = mslop;
        this.tenlop = tenlop;
        this.sisoDukien = sisoDukien;
    }
}
```



Sinhvien.java

@Entity

```
public class Sinhvien {

    @Id
    private String mssv;
    @Column(name = "hosv", columnDefinition = "nvarchar(255)")
    private String ho;
    @Column(name = "tensv", nullable = false, columnDefinition = "nvarchar(255)")
    private String ten;
    private LocalDate ngaysinh;
    @Column(name = "gioitinh", columnDefinition = "nvarchar(255)")
    private String gioitinh;

    @ElementCollection
    @CollectionTable(
        name = "Dienthoai",
        joinColumns = {
            @JoinColumn(name="mssv")
        }
    )
    @Column(name="dienthoai", nullable = false)
    private Set<String> dsDienthoai;
}
```

```

@Embedded
private Diachi diachi;

@OneToMany(mappedBy = "sinhvien")
private List<Bangdiem> dsdk;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name="mslop")
private Lophoc lophoc;

public Sinhvien() {
}

public Sinhvien(String mssv, String ho, String ten, LocalDate ngaysinh, String gioitinh, Set<String>
dsDienthoai) {
    super();
    this.mssv = mssv;
    this.ho = ho;
    this.ten = ten;
    this.ngaysinh = ngaysinh;
    this.gioitinh = gioitinh;
    this.dsDienthoai = dsDienthoai;
}
}

```



Monhoc.java

```

@Entity
public class Monhoc {

    @Id
    private String msmh;
    @Column(name = "tenmh", nullable = false, columnDefinition = "nvarchar(255)")
    private String tenmh;
    private int hockygd;
    private int sotinchi;

    @OneToMany(mappedBy = "monhoc")
    private List<Bangdiem> dsdk;

    public Monhoc() {
    }

    public Monhoc(String msmh, String tenmh, int hockygd, int sotinchi) {
        super();
        this.msmh = msmh;
        this.tenmh = tenmh;
        this.hockygd = hockygd;
        this.sotinchi = sotinchi;
    }
}

```



HosoLophoc.java

@Entity

```

public class Hosolophoc {

    @Id
    private String mshs;
    private Date ngaylap;
    @Column(name = "ghichu", columnDefinition = "text")
    private String ghichu;

    @OneToOne
    @MapsId
    @JoinColumn(name="mslop")
    private Lophoc lophoc;

    public Hosolophoc() {
    }

    public Hosolophoc(String mshs, Date ngaylap, String ghichu) {
        super();
        this.mshs = mshs;
        this.ngaylap = ngaylap;
        this.ghichu = ghichu;
    }
}

```



Bangdiem.java

```

@Entity
@IdClass({BangDiemPK.class})
public class Bangdiem {

    @Id
    @ManyToOne
    @JoinColumn(name = "mssv")
    private Sinhvien sinhvien;
    @Id
    @ManyToOne
    @JoinColumn(name = "msmh")
    private Monhoc monhoc;
    @Id
    private int lanthi;
    private float diemthi;

    public Bangdiem() {
    }

    public Bangdiem(Sinhvien sinhvien, Monhoc monhoc, int lanthi, float diemthi) {
        super();
        this.sinhvien = sinhvien;
        this.monhoc = monhoc;
        this.lanthi = lanthi;
        this.diemthi = diemthi;
    }
}

```



BangDiemPK.java

@Embeddable

```
public class BangDiemPK implements Serializable{
    /**
     *
     */
    private static final long serialVersionUID = -9051618403046863807L;
    private String sinhvien;
    private String monhoc;
    private int lanthi;

    public BangDiemPK() {
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + lanthi;
        result = prime * result + ((monhoc == null) ? 0 : monhoc.hashCode());
        result = prime * result + ((sinhvien == null) ? 0 : sinhvien.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        BangDiemPK other = (BangDiemPK) obj;
        if (lanthi != other.lanthi)
            return false;
        if (monhoc == null) {
            if (other.monhoc != null)
                return false;
        } else if (!monhoc.equals(other.monhoc))
            return false;
        if (sinhvien == null) {
            if (other.sinhvien != null)
                return false;
        } else if (!sinhvien.equals(other.sinhvien))
            return false;
        return true;
    }
}
```



Diachi.java

@Embeddable

```
public class Diachi {
    @Column(columnDefinition = "nvarchar(255)")
    private String sonha;
    @Column(columnDefinition = "nvarchar(255)")
```

```

private String duong;
@Column(columnDefinition = "nvarchar(255)")
private String phuong;
@Column(columnDefinition = "nvarchar(255)")
private String quan;
@Column(columnDefinition = "nvarchar(255)")
private String thanhpho;

public Diachi() {
}

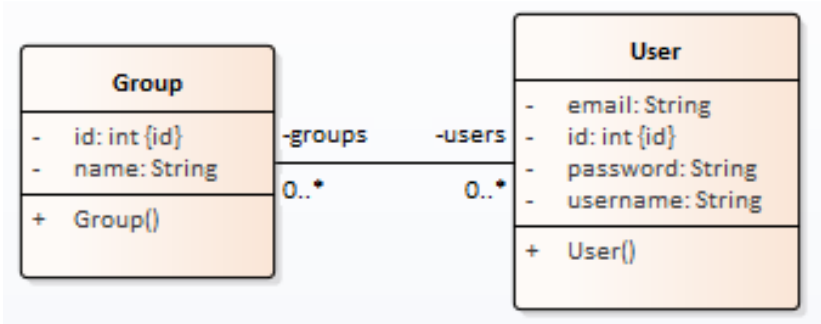
public Diachi(String sonha, String duong, String phuong, String quan, String thanhpho) {
    super();
    this.sonha = sonha;
    this.duong = duong;
    this.phuong = phuong;
    this.quan = quan;
    this.thanhpho = thanhpho;
}
}

```

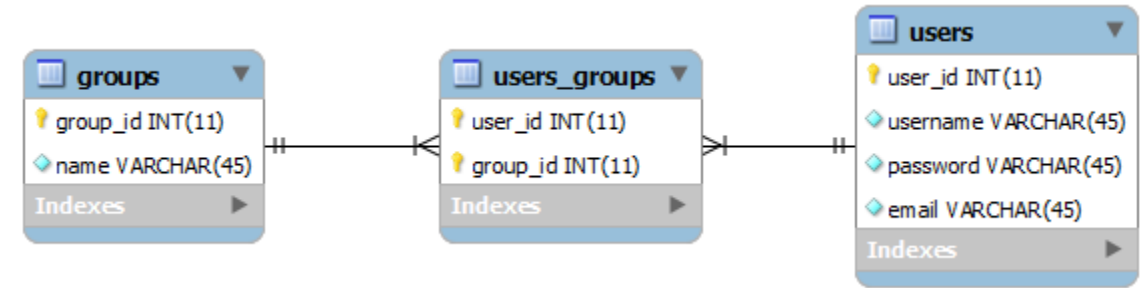
Bài tập

Bài 1:

Cho mô hình lớp sau:

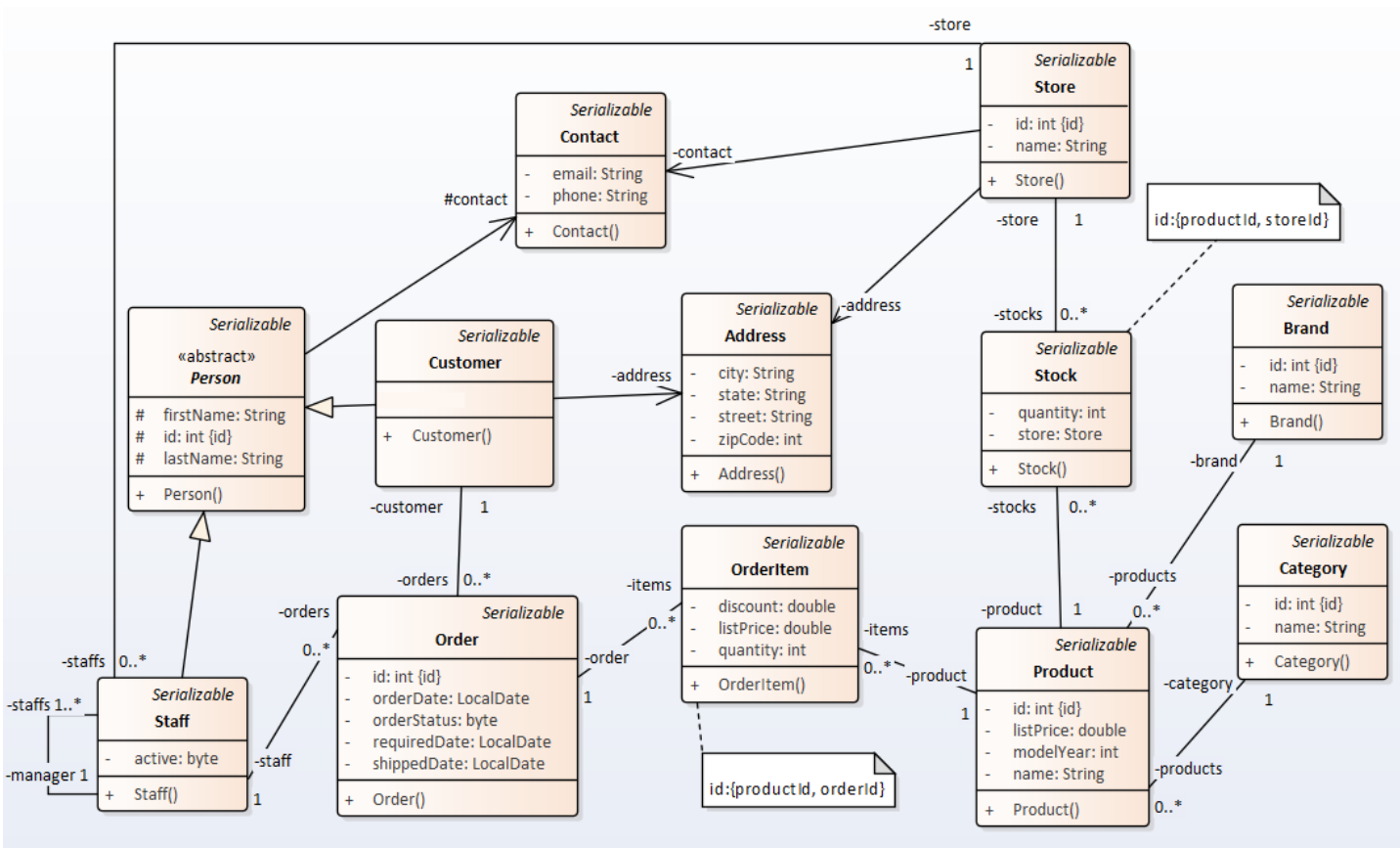


Viết các lớp persistence entity, sau cho ánh xạ ra được mô hình CSDL quan hệ sau:

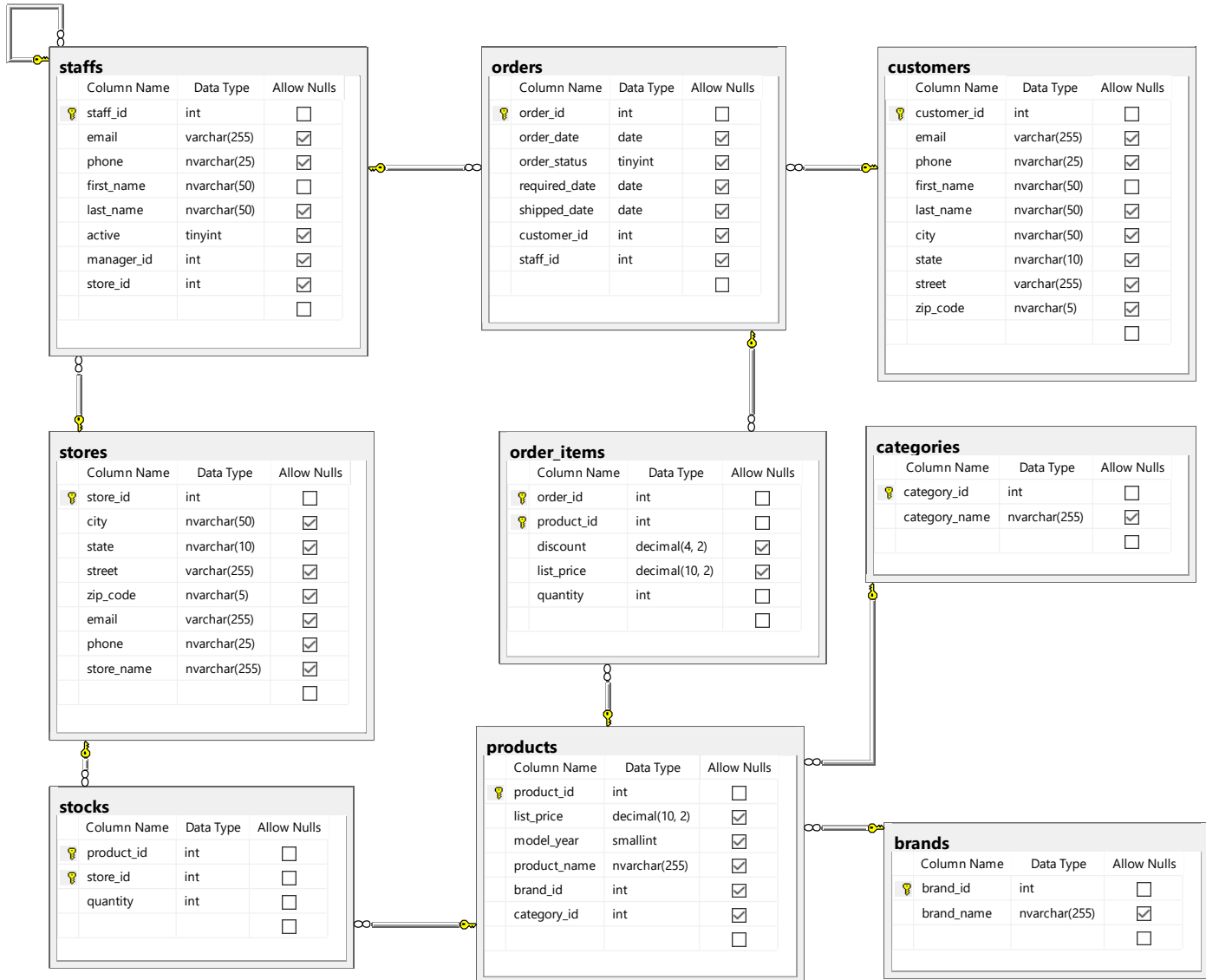


Bài 2:

A) Cho mô hình lớp sau:



Viết các lớp persistence entity, sau cho ánh xạ ra được mô hình CSDL quan hệ sau:



B) Load dữ liệu database (MS SQL Server)

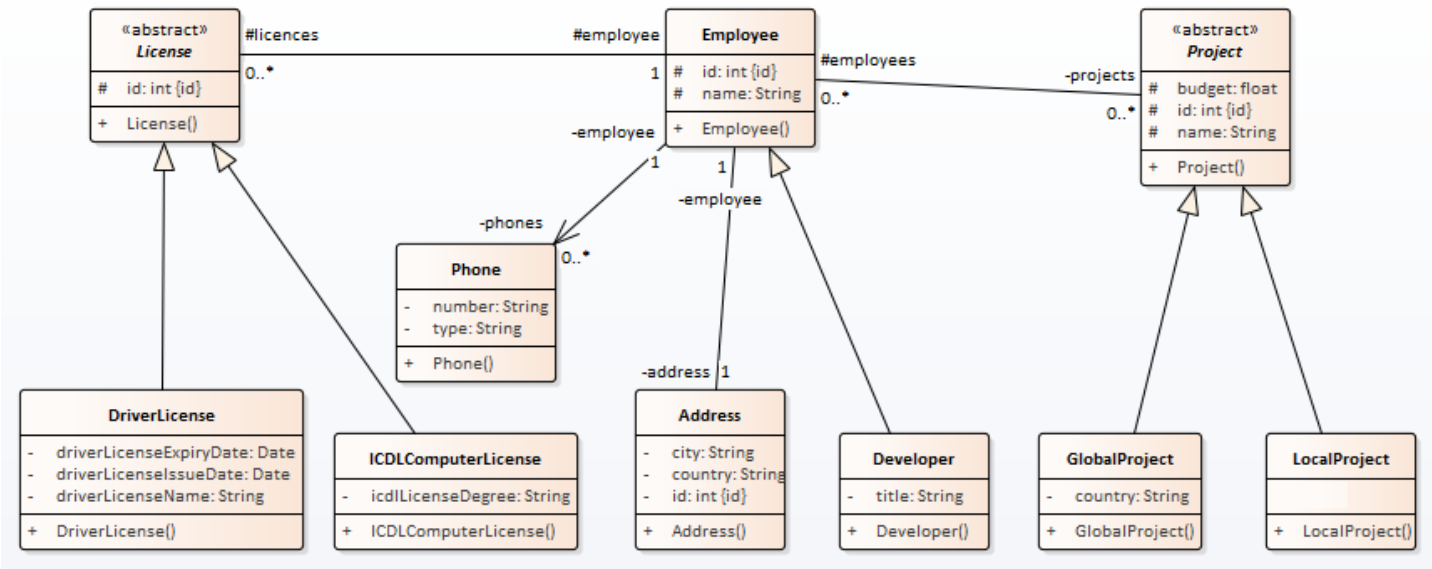
<https://www.mediafire.com/file/lqjpf4kvrwh4yt/BikeStores.rar>

C) (Java Programming) Viết các câu truy vấn sau:

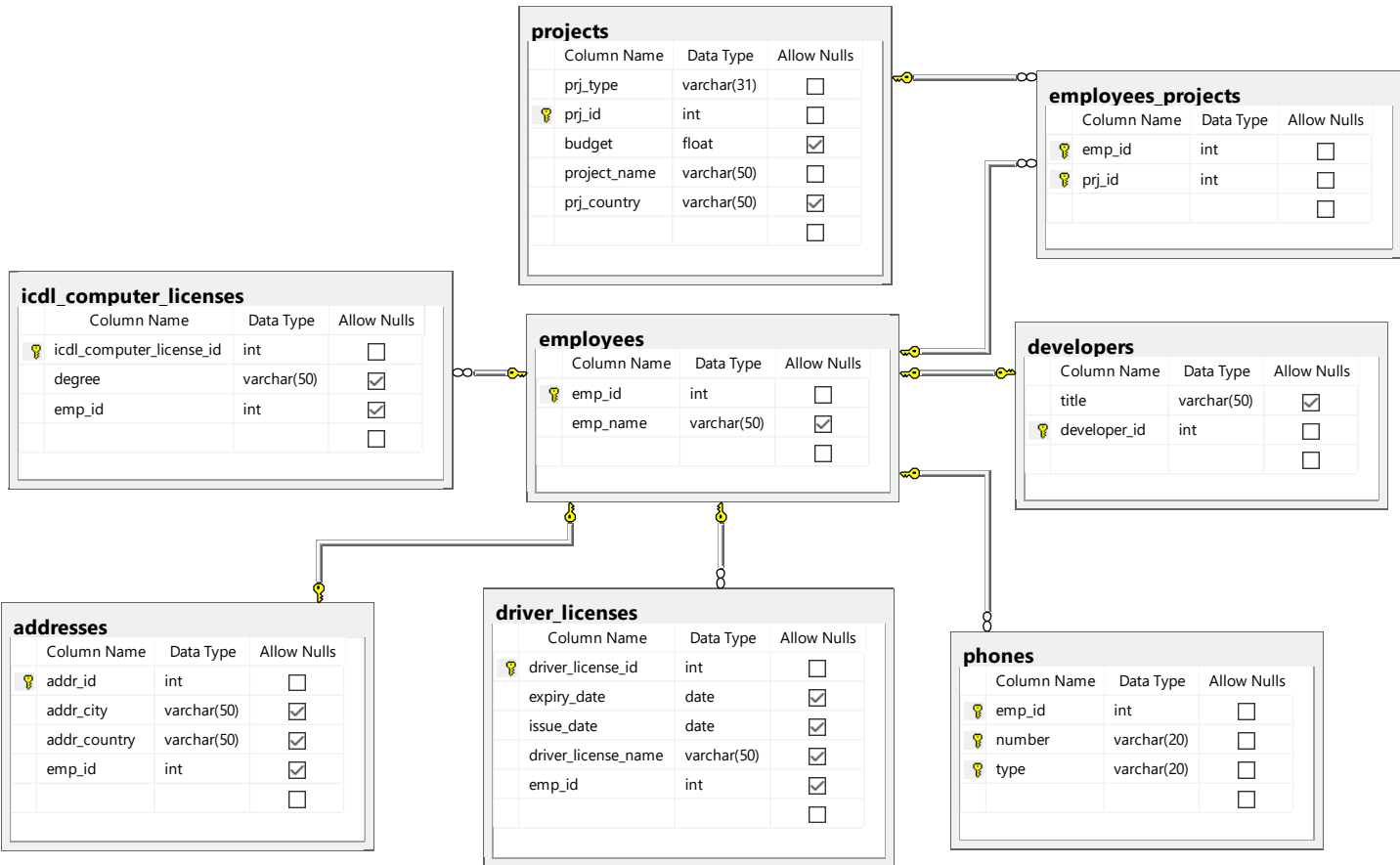
- Thực hiện các thao tác Create, Update, Delete, Find by Id trên từng đối tượng.
- Tìm danh sách sản phẩm có giá cao nhất.
- Tìm danh sách sản phẩm chưa bán được lần nào.
- Thống kê số khách hàng theo từng bang.
+ getNumberCustomerByState() : Map<String, Integer>
- Tính tổng tiền của đơn hàng khi biết mã số đơn hàng.
- Đếm số đơn hàng của từng khách hàng.
+ getOrdersByCustomers() : Map<Customer, Integer>
- Tính tổng số lượng của từng sản phẩm đã bán ra.
+ getTotalProduct(): Map<Product, Integer>
- Tính tổng tiền của tất cả các hóa đơn trong một ngày nào đó.
- Cập nhật giá của sản phẩm khi biết mã sản phẩm.
- Xóa tất cả các khách hàng chưa mua hàng.
- Danh sách các khách hàng có từ 2 số điện thoại trở lên
- Thống kê tổng tiền hóa đơn theo tháng / năm.
- Tìm khách hàng khi biết số điện thoại.

Bài 3:

Cho mô hình lớp sau:



Viết các lớp persistence entity, sau cho ánh xạ ra được mô hình CSDL quan hệ sau:



Phần 2: Object/Grid Mapper (OGM)

Download driver : <https://hibernate.org/ogm/releases/5.4/>

Hibernate Validator: https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/

Document: https://docs.jboss.org/hibernate/ogm/5.4/reference/en-US/html_single/

Compatibility

| | |
|------------------|------|
| Java | 8 |
| JPA | 2.2 |
| Hibernate ORM | 5.3 |
| Hibernate Search | 5.10 |

Installation

```
<dependencies>
  <dependency>
    <groupId>org.hibernate.ogm</groupId>
    <artifactId>hibernate-ogm-mongodb</artifactId>
    <version>5.4.1.Final</version>
  </dependency>
</dependencies>
```

Nếu dùng offline, các thư viện cần:

- /lib/core
- /lib/mongodb

Optional khác – may be /lib/provided

Configuration



hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>

    <!-- Database connection settings -->
    <property name="hibernate.ogm.datastore.provider">mongodb</property>
    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
    <property name="hibernate.ogm.datastore.host">localhost</property>
    <property name="hibernate.ogm.mongodb.port">27017</property>
    <property name="hibernate.ogm.datastore.database">DB_Name</property>
    <property name="hibernate.ogm.datastore.create_database">true</property>

  </session-factory>
</hibernate-configuration>
```

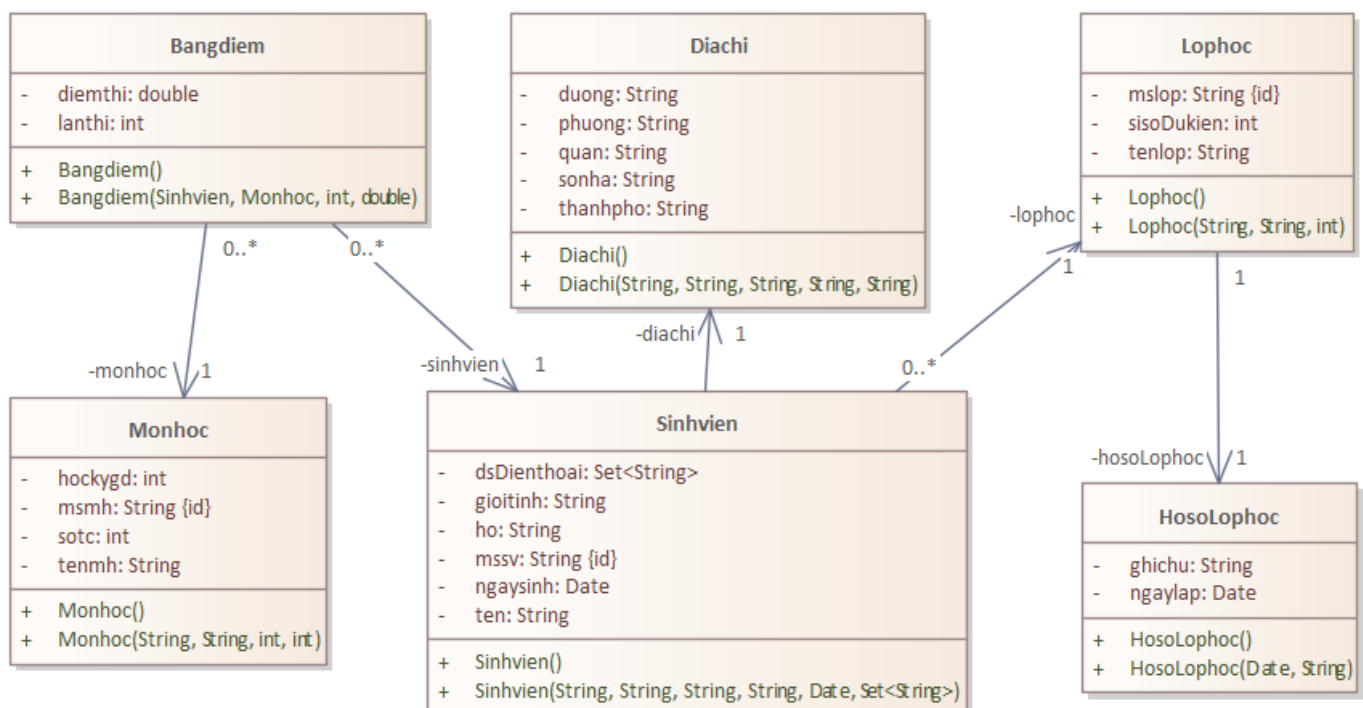
Note:

- 1/ Note that parameterized queries are not supported by MongoDB, so don't expect Query#setParameter() to work.
- 2/ You can limit the results of a query using the setMaxResults(...) method.
- 3/ Keys need to be given within quotes; the only relaxation of this is that single quotes may be used when specifying attribute names/values to facilitate embedding queries within Java strings.
- 4/ Note that results of projections are returned as retrieved from the MongoDB driver at the moment and are not (yet) converted using suitable Hibernate OGM type implementations.
- 5/ Support for the \$regex operator is limited to the string syntax. We do not support the /pattern/ syntax as it is not currently supported by the MongoDB Java driver.

Bài 1: Quản lý sinh viên

A) Ánh xạ mô hình lớp sang MongoDB schema tương ứng

class Basic Class Diagram with Attributes and Operations

**Lophoc Json Data**

```

{
  "_id": "DHKTPM15B",
  "tenlop": "Đại học kỹ thuật phần mềm 15B",
  "hosolophoc": {
    "ngaylap": {
      "$date": "2021-10-11T23:05:41.557Z"
    },
    "ghichu": "Hồ sơ lớp học: DHKTPM15B"
  },
  "sisoDukien": 80
}

```

Sinhvien Json Data

```

{

```

```

    "_id": "19234561",
    "diachi": {
        "phuong": "4",
        "sonha": "12",
        "thanhpho": "HCM",
        "duong": "Nguyễn Văn Bảo",
        "quan": "Gò Vấp"
    },
    "tensv": "Lan",
    "gioitinh": "Nữ",
    "ngaysinh": {
        "$date": "2001-01-09T17:00:00Z"
    },
    "hosv": "Nguyễn Hoàng",
    "dsDienthoai": [
        "0913444555",
        "0914444555"
    ]
}

```

Monhoc Json Data

```

{
    "_id": "LTPTJava",
    "hockygd": 5,
    "tenmh": "Lập trình phân tán Java",
    "sotc": 3
}

```

Bangdiem Json Data

```

{
    "_id": {
        "lanthi": 2,
        "msmh": "LTPTJava",
        "mssv": "19234561"
    },
    "diemthi": 8.5
}

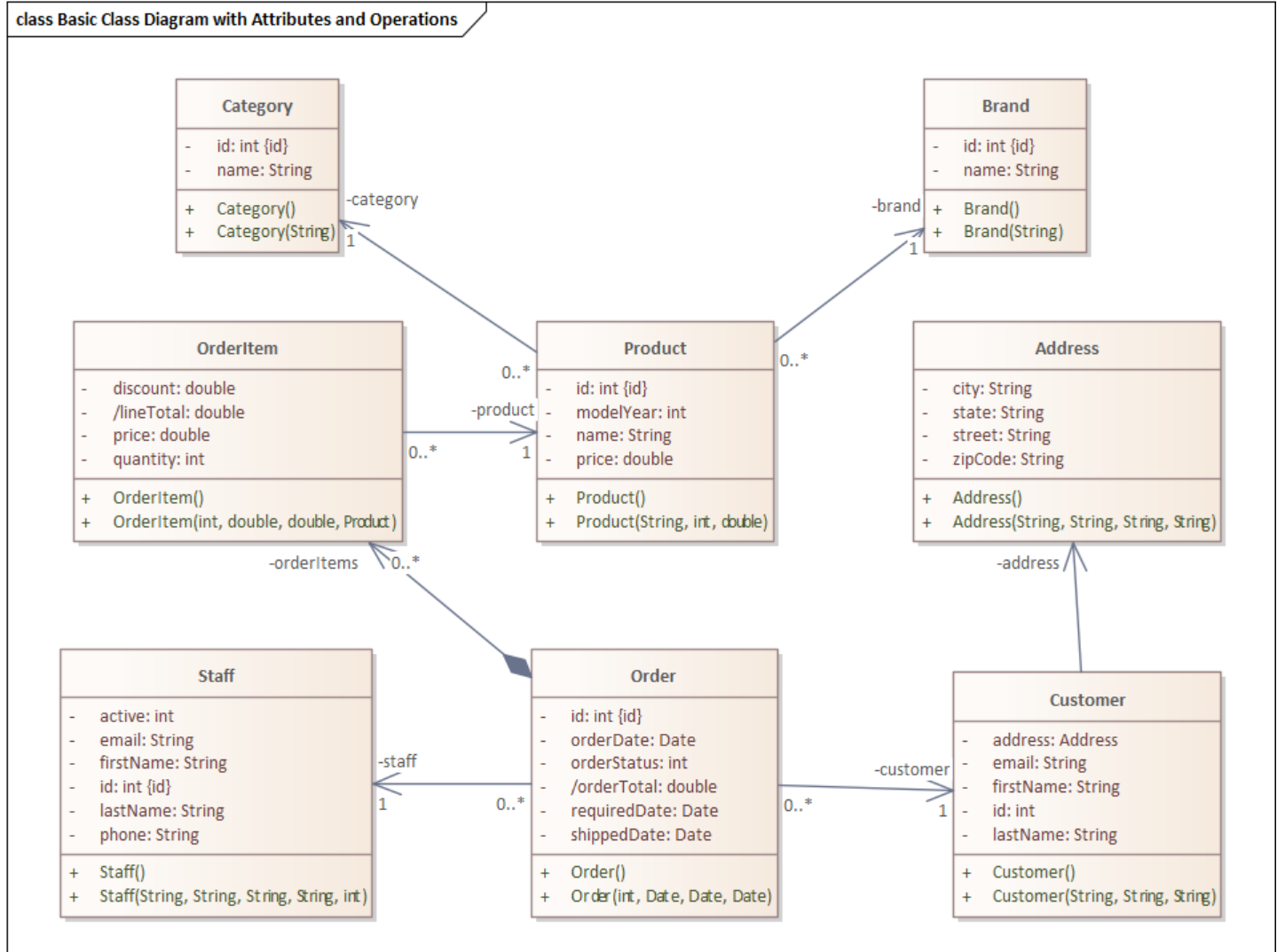
```

B) (Java Programming) Viết các phương thức với các thao tác CRUD


- Thực hiện các thao tác Create, Update, Delete, Find by Id, Get All trên từng đối tượng.
- Tính sĩ số sinh viên theo từng lớp học.
+ getSisoByLophoc() : Map<Lophoc, Integer>
- Tính điểm trung bình của các môn học của các sinh viên
+ listSinhvienDiemTB() : Map<Sinhvien, Float>
- Những lớp học chưa có sinh viên
+ listLophocNull() : List<Lophoc>
- Những sinh viên học môn “Lập Trình Phân tán với Công Nghệ Java” có điểm cao nhất.
+ listSinhvienGioiJava(): List<Sinhvien>

Bài 2: Quản lý bán hàng

A) Ánh xạ mô hình lớp sau sang MongoDB schema tương ứng



| | |
|--|---|
| Brand.java | Brand Json Data |
| <pre> public class Brand { private int id; private String name; public Brand() { } } </pre> | <pre> { "_id": 7, "brand_name": "Sun Bicycles" } </pre> |
| Category.java | Category Json Data |
| <pre> public class Category { private int id; private String name; public Category() { } } </pre> | <pre> { "_id": 4, "category_name": "Cyclocross Bicycles" } </pre> |
| Address.java | Customer Json Data |
| <pre> public class Address { </pre> | <pre> { </pre> |

| | |
|--|---|
| <pre>private String street; private String city; private String state; private String zipCode; public Address() { }</pre> | <pre>"_id": 1445, "first_name": "Ester", "last_name": "Acevedo", "email": "ester.acevedo@gmail.com", "address": { "city": "San Lorenzo", "state": "CA", "street": "671 Miles Court ", "zip_code": "94580" }</pre> |
|  Customer.java | |
| <pre>public class Customer { private int id; private String firstName; @Column(name="last_name") private String lastName; private String email; }</pre> | |
|  Product.java | Product Json Data |
| <pre>public class Product { private int id; private String name; private int modelYear; private double price; public Product() { } }</pre> | <pre>{ "_id": 16, "product_name": "Electra Townie Original 7D EQ - 2016", "model_year": 2016, "list_price": 599.99, "brand_id": 1, "category_id": 3 }</pre> |
|  Staff.java | Staff Json Data |
| <pre>public class Staff { private int id; private String firstName; private String lastName; private String email; private String phone; private int active; public Staff() { } }</pre> | <pre>{ "_id": 10, "first_name": "Bernardine", "last_name": "Houston", "email": "bernardine.houston@bikes.shop", "phone": "(972) 530-5557", "active": 1, "manager_id": 7 }</pre> |
|  OrderItem.java | Order Json Data |
| <pre>public class OrderItem { private int quantity; private double price; private double discount; private double lineTotal; }</pre> | <pre>{ "_id": 14, "order_status": 4, "order_date": { "\$date": "2016-02-08T17:00:00Z" }, }</pre> |

| | |
|--|--|
| <pre> public OrderItem() { } </pre> | <pre> "required_date": { "\$date": "2016-02-08T17:00:00Z" }, "shipped_date": { "\$date": "2016-02-08T17:00:00Z" }, "staff_id": 3, "customer_id": 258, "order_items": [{ "quantity": 1, "list_price": 469.99, "discount": 0.07, "product_id": 6 }] </pre> |
| <div>Order.java</div> <pre> public class Order { private int id; private int orderStatus; private Date orderDate; private Date requiredDate; private Date shippedDate; private double orderTotal; public Order() { } } </pre> | |

B) (Java Programming) Viết các phương thức với các thao tác CRUD

- Thực hiện các thao tác Create, Update, Delete, Find theo id, Find theo tên, Get & skip & limit trên từng đối tượng.
- Hiển thị các sản phẩm có model trong 1 năm bất kỳ.
- Hiển thị các sản phẩm trong một danh mục bất kỳ khi biết mã số danh mục.
- Hiển thị các sản phẩm trong một danh mục bất kỳ khi biết tên danh mục.
- Hiển thị các sản phẩm trong một nhãn hiệu bất kỳ khi biết mã số nhãn hiệu.
- Hiển thị các sản phẩm trong một nhãn hiệu bất kỳ khi biết tên nhãn hiệu.
- Liệt kê danh sách các hóa đơn có bán sản phẩm nào đó khi biết tên sản phẩm
- Liệt kê danh sách các hóa đơn có mua sản phẩm có chiết khấu (*discount*) từ 20% trở lên, sắp xếp theo ngày đơn hàng giảm dần. Danh sách gồm: Mã đơn hàng, ngày đơn hàng, tên sản phẩm được chiết khấu và tổng tiền.
- Tính tổng số sản phẩm theo từng danh mục, thông tin bao gồm: Mã danh mục, tên danh mục và số sản phẩm.
- Dùng text search để tìm kiếm trên 1 cột hoặc trên nhiều cột có giá trị kiểu chuỗi.
- Tìm danh sách sản phẩm có giá cao nhất.
- Tìm danh sách sản phẩm chưa bán được lần nào.
- Thống kê số khách hàng theo từng bang.
 - + `getNumberCustomerByState() : Map<String, Integer>`
- Tính tổng tiền của đơn hàng khi biết mã số đơn hàng.
- Đếm số đơn hàng của từng khách hàng.
 - + `getOrdersByCustomers() : Map<Customer, Integer>`
- Tính tổng số lượng của từng sản phẩm đã bán ra.
 - + `getTotalProduct(): Map<Product, Integer>`
- Tính tổng tiền của tất cả các hóa đơn trong một ngày nào đó.