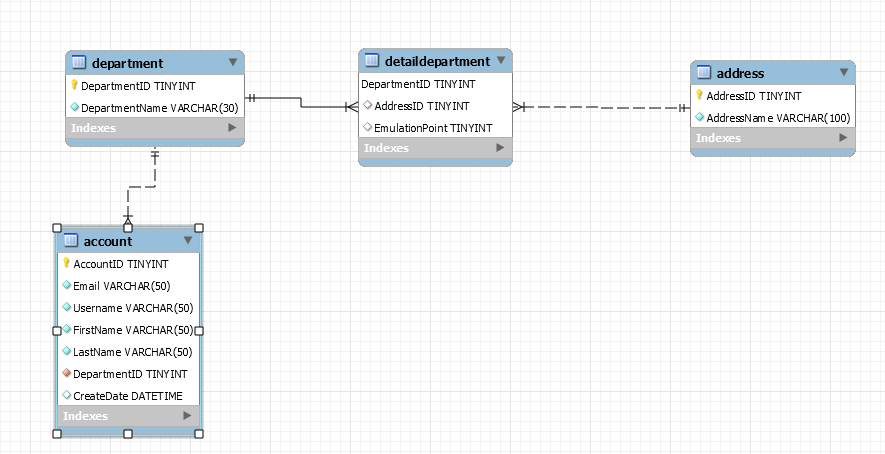
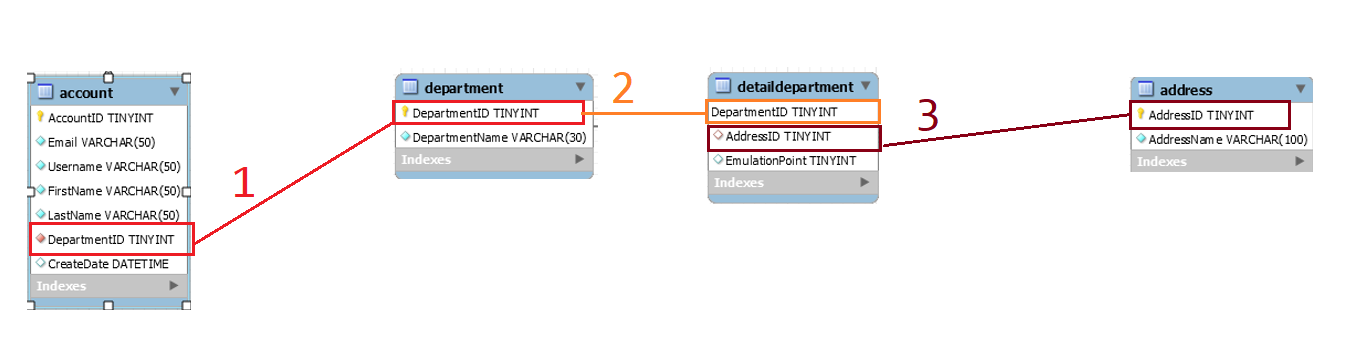
# **1. Hibernate One To One, OneToMany, ManyToOne, Inheritance.**





- Tạo DB TestingSystem để Demo chương trình. Gồm 4 bảng Account, Department, Address, DetailDepartment Quan hệ giữa các bảng này như sau:

- 1 Department có nhiều Account.

- 1 Department có 1 DetailDepartment. DetailDepartment cấu hình kế thừa từ Department.

- 1 DetailDepartment có 1 Address.

|  |
| --- |
| -- create table : address  **DROP** **TABLE** IF **EXISTS** Address;  **CREATE** **TABLE** Address(  AddressID TINYINT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,  AddressName NVARCHAR(100) **NOT** **NULL** **UNIQUE** **KEY**  );  -- create table: Department  **DROP** **TABLE** IF **EXISTS** Department;  **CREATE** **TABLE** Department(  DepartmentID TINYINT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,  DepartmentName NVARCHAR(30) **NOT** **NULL** **UNIQUE** **KEY**  );  -- create table : Detail Department  **DROP** **TABLE** IF **EXISTS** DetailDepartment;  **CREATE** **TABLE** DetailDepartment(  DepartmentID TINYINT UNSIGNED **PRIMARY** **KEY**,  AddressID TINYINT UNSIGNED, -- UNIQUE KEY, -- NOT NULL ,  EmulationPoint TINYINT UNSIGNED,  **FOREIGN** **KEY**(DepartmentID) **REFERENCES** Department(DepartmentID),  **FOREIGN** **KEY**(AddressID) **REFERENCES** Address(AddressID)  );  -- create table: Account  **DROP** **TABLE** IF **EXISTS** `Account`;  **CREATE** **TABLE** `Account`(  AccountID TINYINT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,  Email **VARCHAR**(50) **NOT** **NULL** **UNIQUE** **KEY**, -- Cannot update this field  Username **VARCHAR**(50) **NOT** **NULL** **UNIQUE** **KEY**, -- Cannot update this field  FirstName NVARCHAR(50) **NOT** **NULL**,  LastName NVARCHAR(50) **NOT** **NULL**, -- create field fullName in POJO  DepartmentID TINYINT UNSIGNED **NOT** **NULL**, -- Set default waiting  CreateDate DATETIME **DEFAULT** NOW(), -- Cannot update this field  **FOREIGN** **KEY**(DepartmentID) **REFERENCES** Department(DepartmentID)  ); |

**- 1 Department có nhiều Account.**

**Config Table Account:**

|  |
| --- |
| @ManyToOne  @JoinColumn(name = "DepartmentID", nullable = **false**)  **private** Department department; |

**Config Table Department:**

|  |
| --- |
| @OneToMany(mappedBy = "department", fetch = FetchType.***EAGER***)  **private** List<Account> account; |

**- 1 Department có 1 DetailDepartment. DetailDepartment cấu hình kế thừa từ Department.**

**Config Table Department:**

|  |
| --- |
| @Entity  @Table(name = "Department", catalog = "TestingSystem")  @Inheritance(strategy = InheritanceType.***JOINED***)  **public** **class** Department **implements** Serializable |

**Config Table DetailDepartment:**

|  |
| --- |
| @Entity  @Table(name = "DetailDepartment", catalog = "TestingSystem")  @PrimaryKeyJoinColumn(name = "DepartmentID")  **public** **class** DetailDepartment **extends** Department **implements** Serializable |

**- 1 DetailDepartment có 1 Address.**

**Config Table DetailDepartment:**

|  |
| --- |
| @OneToOne(cascade = CascadeType.***ALL***)  @JoinColumn(name = "AddressID", referencedColumnName = "AddressID")  **private** Address address; |

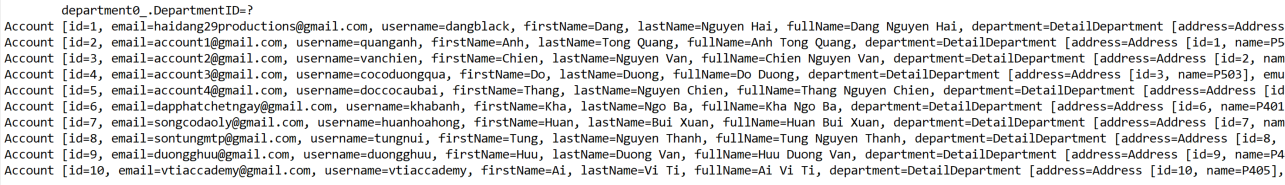
**Config Table Address:**

|  |
| --- |
| @OneToOne(mappedBy = "address")  **private** DetailDepartment department; |

# **2. Thực hiện Demo.**

# **Giao diện chính chương trình**

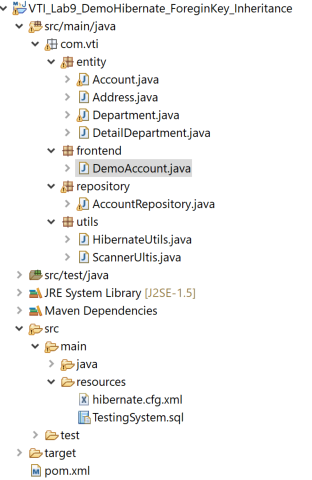
## Giao diện chính:



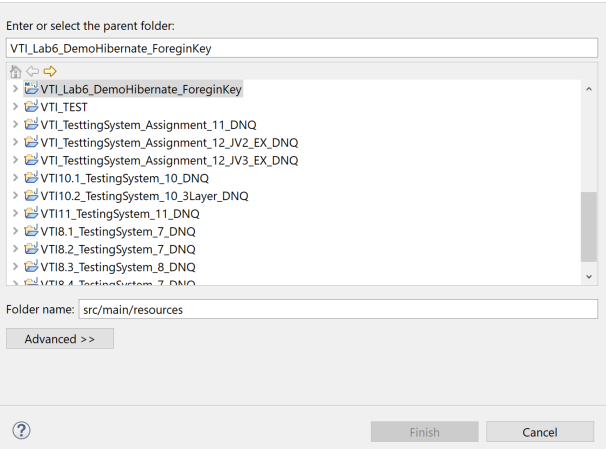
# **Tạo** khung **chương trình:**

Tạo mới 1 project: VTI\_Lab9\_DemoHibernate\_ForeginKey\_Inheritance

Tạo các Package trong src: com.vti.repository, com.vti.entiy, com.vti.frontend, com.vti.utils



Tạo thư mục src/main/resources:



Tạo file: 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=*"connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>  <property name=*"connection.url"*>jdbc:mysql://localhost:3306/TestingSystem</property>  <property name=*"connection.username"*>root</property>  <property name=*"connection.password"*>root</property>  <!-- format code SQL -->  <property name=*"show\_sql"*>true</property>  <property name=*"hibernate.format\_sql"*>true</property>  <property name=*"connection.pool\_size"*>10</property>  <!-- other -->  <property name=*"hibernate.connection.characterEncoding"*>utf8</property>  </session-factory>  </hibernate-configuration> |



Sửa file: pom.xml

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.vti</groupId>  <artifactId>VTI\_Lab6\_DemoHibernate\_ForeginKey</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>VTI\_Lab6\_DemoHibernate\_ForeginKey</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.4.17.Final</version>  </dependency>  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.20</version>  </dependency>  </dependencies>  </project> |



Tạo file: TestingSystem.sql



# **Tạo Class HibernateUtils trong package ultis:**

|  |
| --- |
| **package** com.vti.utils;  **import** org.hibernate.Session;  **import** org.hibernate.SessionFactory;  **import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;  **import** org.hibernate.cfg.Configuration;  **import** org.hibernate.service.ServiceRegistry;  **import** com.vti.entity.Account;  **import** com.vti.entity.Address;  **import** com.vti.entity.Department;  **import** com.vti.entity.DetailDepartment;  **public** **class** HibernateUtils {  **private** **static** HibernateUtils instance;  **private** Configuration configuration;  **private** SessionFactory sessionFactory;  **public** **static** HibernateUtils getInstance() {  **if** (**null** == instance) {  instance = **new** HibernateUtils();  }  **return** instance;  }  **private** HibernateUtils() {  configure();  }  **private** **void** configure() {  // load configuration  configuration = **new** Configuration();  configuration.configure("hibernate.cfg.xml");  // add entity  configuration.addAnnotatedClass(Account.**class**);  configuration.addAnnotatedClass(Department.**class**);  configuration.addAnnotatedClass(Address.**class**);  configuration.addAnnotatedClass(DetailDepartment.**class**);  }  **private** SessionFactory buildSessionFactory() {  **if** (**null** == sessionFactory || sessionFactory.isClosed()) {  ServiceRegistry serviceRegistry = **new** StandardServiceRegistryBuilder()  .applySettings(configuration.getProperties()).build();  sessionFactory = configuration.buildSessionFactory(serviceRegistry);  }  **return** sessionFactory;  }  **public** **void** closeFactory() {  **if** (**null** != sessionFactory && sessionFactory.isOpen()) {  sessionFactory.close();  }  }  **public** Session openSession() {  buildSessionFactory();  **return** sessionFactory.openSession();  }  } |

****

# **Tạo Class ScannerUltis trong package ultis:**

|  |
| --- |
| **package** com.vti.utils;  **import** java.text.SimpleDateFormat;  **import** java.time.LocalDate;  **import** java.util.Scanner;  **public** **class** ScannerUltis {  **private** **static** Scanner *sc* = **new** Scanner(System.***in***);  **public** **static** **int** inputInt() {  **while** (**true**) {  **try** {  **return** Integer.*parseInt*(*sc*.next().trim());  } **catch** (Exception e) {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** **int** inputIntPositive() {  **while** (**true**) {  **try** {  **int** intPositive = Integer.*parseInt*(*sc*.next());  **if** (intPositive >= 0) {  **return** intPositive;  } **else** {  System.***err***.println("Nhập lại:");  }  } **catch** (Exception e) {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** Float inputFloat() {  **while** (**true**) {  **try** {  **return** Float.*parseFloat*(*sc*.next());  } **catch** (Exception e) {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** Double inputDouble() {  **while** (**true**) {  **try** {  **return** Double.*parseDouble*(*sc*.next());  } **catch** (Exception e) {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** String inputString() {  **while** (**true**) {  String string = *sc*.next().trim();  **if** (!string.isEmpty()) {  **return** string;  } **else** {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** LocalDate inputLocalDate() {  System.***out***.println("Nhập theo định dạng yyyy-MM-dd");  SimpleDateFormat format = **new** SimpleDateFormat("yyyy-MM-dd");  **while** (**true**) {  String localdate = *sc*.next().trim();  **try** {  **if** (format.parse(localdate) != **null**) {  LocalDate lc = LocalDate.*parse*(localdate);  **return** lc;  }  } **catch** (Exception e) {  System.***err***.println("Nhập lại:");  }  }  }  **public** **static** String inputEmail() {  **while** (**true**) {  String email = ScannerUltis.*inputString*();  **if** (email == **null** || !email.contains("@")) {  System.***out***.print("Nhập lại: ");  } **else** {  **return** email;  }  }  }  **public** **static** **int** inputFunction(**int** a, **int** b, String errorMessage) {  **while** (**true**) {  **int** number = ScannerUltis.*inputInt*();  **if** (number >= a && number <= b) {  **return** number;  } **else** {  System.***err***.println(errorMessage);  }  }  }  **public** **static** String inputPassword() {  **while** (**true**) {  String password = ScannerUltis.*inputString*();  **if** (password.length() < 6 || password.length() > 12) {  System.***out***.print("Nhập lại: ");  **continue**;  }  **boolean** hasAtLeast1Character = **false**;  **for** (**int** i = 0; i < password.length(); i++) {  **if** (Character.*isUpperCase*(password.charAt(i)) == **true**) {  hasAtLeast1Character = **true**;  **break**;  }  }  **if** (hasAtLeast1Character == **true**) {  **return** password;  } **else** {  System.***out***.print("Mời bạn nhập lại password: ");  }  }  }  **public** **static** String inputPhoneNumber() {  **while** (**true**) {  String number = ScannerUltis.*inputString*();  **if** (number.length() > 12 || number.length() < 9) {  **continue**;  }  **if** (number.charAt(0) != '0') {  **continue**;  }  **boolean** isNumber = **true**;  **for** (**int** i = 0; i < number.length(); i++) {  **if** (Character.*isDigit*(number.charAt(i)) == **false**) {  isNumber = **false**;  **break**;  }  }  **if** (isNumber == **true**) {  **return** number;  } **else** {  System.***out***.print("Nhập lại: ");  }  }  }  } |

****

# **Tạo Class Account trong Entity:**

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** java.util.Date;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.JoinColumn;  **import** javax.persistence.ManyToOne;  **import** javax.persistence.Table;  **import** javax.persistence.Temporal;  **import** javax.persistence.TemporalType;  **import** org.hibernate.annotations.CreationTimestamp;  **import** org.hibernate.annotations.Formula;  **import** lombok.Getter;  **import** lombok.NoArgsConstructor;  **import** lombok.Setter;  **import** lombok.ToString;  @Entity  @Table(name = "`Account`", catalog = "TestingSystem")  @Getter  @Setter  @NoArgsConstructor  @ToString  **public** **class** Account **implements** Serializable {  @Column(name = "AccountID")  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  **private** **short** id;  @Column(name = "Email", length = 50, nullable = **false**, unique = **true**, updatable = **false**)  **private** String email;  @Column(name = "Username", length = 50, nullable = **false**, unique = **true**, updatable = **false**)  **private** String username;  @Column(name = "FirstName", length = 50, nullable = **false**)  **private** String firstName;  @Column(name = "LastName", length = 50, nullable = **false**)  **private** String lastName;  @Formula(" concat(FirstName, ' ', LastName) ")  **private** String fullName;  @ManyToOne  @JoinColumn(name = "DepartmentID", nullable = **false**)  **private** Department department;  @Column(name = "CreateDate")  @Temporal(TemporalType.TIMESTAMP)  @CreationTimestamp  **private** Date createDate;  } |

****

# **Tạo Class Department trong Entity:**

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** java.util.List;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.FetchType;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Inheritance;  **import** javax.persistence.InheritanceType;  **import** javax.persistence.OneToMany;  **import** javax.persistence.Table;  **import** lombok.Getter;  **import** lombok.NoArgsConstructor;  **import** lombok.Setter;  **import** lombok.ToString;  @Entity  @Table(name = "Department", catalog = "TestingSystem")  @Setter  @Getter  @NoArgsConstructor  @ToString  @Inheritance(strategy = InheritanceType.***JOINED***)  **public** **class** Department **implements** Serializable {  @Column(name = "DepartmentID")  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **short** id;  @Column(name = "DepartmentName", length = 30, nullable = **false**, unique = **true**)  **private** String name;  @OneToMany(mappedBy = "department", fetch = FetchType.***EAGER***)  **private** List<Account> account;  } |

****

# **Tạo Class DetailDepartment trong Entity:**

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** javax.persistence.CascadeType;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.JoinColumn;  **import** javax.persistence.OneToOne;  **import** javax.persistence.PrimaryKeyJoinColumn;  **import** javax.persistence.Table;  **import** lombok.Getter;  **import** lombok.NoArgsConstructor;  **import** lombok.Setter;  **import** lombok.ToString;  @Entity  @Table(name = "DetailDepartment", catalog = "TestingSystem")  @Setter  @Getter  @NoArgsConstructor  @ToString  @PrimaryKeyJoinColumn(name = "DepartmentID")  **public** **class** DetailDepartment **extends** Department **implements** Serializable {  **private** **static** **final** **long** serialVersionUID = 1L;  @OneToOne(cascade = CascadeType.ALL)  @JoinColumn(name = "AddressID", referencedColumnName = "AddressID")  **private** Address address;  @Column(name = "EmulationPoint")  **private** **short** emulationPoint;  } |

****

# **Tạo Class Address trong Entity:**

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.OneToOne;  **import** javax.persistence.Table;  **import** lombok.Getter;  **import** lombok.NoArgsConstructor;  **import** lombok.Setter;  **import** lombok.ToString;  @Entity  @Table(name = "Address", catalog = "TestingSystem")  @Setter  @Getter  @NoArgsConstructor  @ToString  **public** **class** Address **implements** Serializable {  **private** **static** **final** **long** serialVersionUID = 1L;  @Column(name = "AddressID")  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  **private** **short** id;  @Column(name = "AddressName", length = 100, nullable = **false**, unique = **true**)  **private** String name;  @OneToOne(mappedBy = "address")  **private** DetailDepartment department;  } |

****

# **Tạo Class AccountRepository trong Repository:**

|  |
| --- |
| **package** com.vti.repository;  **import** java.util.List;  **import** org.hibernate.Session;  **import** org.hibernate.query.Query;  **import** com.vti.entity.Account;  **import** com.vti.ultis.HibernateUtils;  **public** **class** AccountRepository {  **private** HibernateUtils hibernateUtils;  **public** AccountRepository() {  hibernateUtils = HibernateUtils.getInstance();  }  @SuppressWarnings("unchecked")  **public** List<Account> getAllAccount() {  Session session = **null**;  **try** {  // get session  session = hibernateUtils.openSession();  // create hql query  Query<Account> query = session.createQuery("FROM Account order by id");  **return** query.list();  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  } |

****

# **Tạo Class DetailDepartmentRepository trong Repository:**

|  |
| --- |
| **package** com.vti.repository;  **import** java.util.List;  **import** org.hibernate.Session;  **import** org.hibernate.query.Query;  **import** com.vti.entity.Account;  **import** com.vti.entity.DetailDepartment;  **import** com.vti.ultis.HibernateUtils;  **public** **class** DetailDepartmentRepository {  **private** HibernateUtils hibernateUtils;  **public** DetailDepartmentRepository() {  hibernateUtils = HibernateUtils.getInstance();  }  @SuppressWarnings("unchecked")  **public** List<DetailDepartment> getDetailDepartment() {  Session session = **null**;  **try** {  // get session  session = hibernateUtils.openSession();  // create hql query  Query<DetailDepartment> query = session.createQuery("FROM DetailDepartment", DetailDepartment.**class**);  **return** query.list();  } **finally** {  **if** (session != **null**) {  session.close();  }  }  }  } |

# **Tạo Class DemoAccount trong Frontend:**

|  |
| --- |
| **package** com.vti.frontend;  **import** java.util.List;  **import** com.vti.entity.Account;  **import** com.vti.entity.DetailDepartment;  **import** com.vti.repository.AccountRepository;  **import** com.vti.repository.DetailDepartmentRepository;  **public** **class** DemoHQL {  **public** **static** **void** main(String[] args) {  System.out.println("Danh sách Account trên hệ thống");  AccountRepository accRepository = **new** AccountRepository();  List<Account> listAcc = accRepository.getAllAccount();  **for** (Account account : listAcc) {  System.out.println(account.getId() + account.getUsername() + account.getDepartment().getName());  }  System.out.println("Danh sách DetailDepartment trên hệ thống");  DetailDepartmentRepository detailDepartmentRepository = **new** DetailDepartmentRepository();  List<DetailDepartment> listDetailDepartment = detailDepartmentRepository.getDetailDepartment();  **for** (DetailDepartment detailDepartment : listDetailDepartment) {  System.out.println("ID: " + detailDepartment.getId() + " Name: " + detailDepartment.getName() + " Address: "  + detailDepartment.getAddress().getName());  }  }  } |

