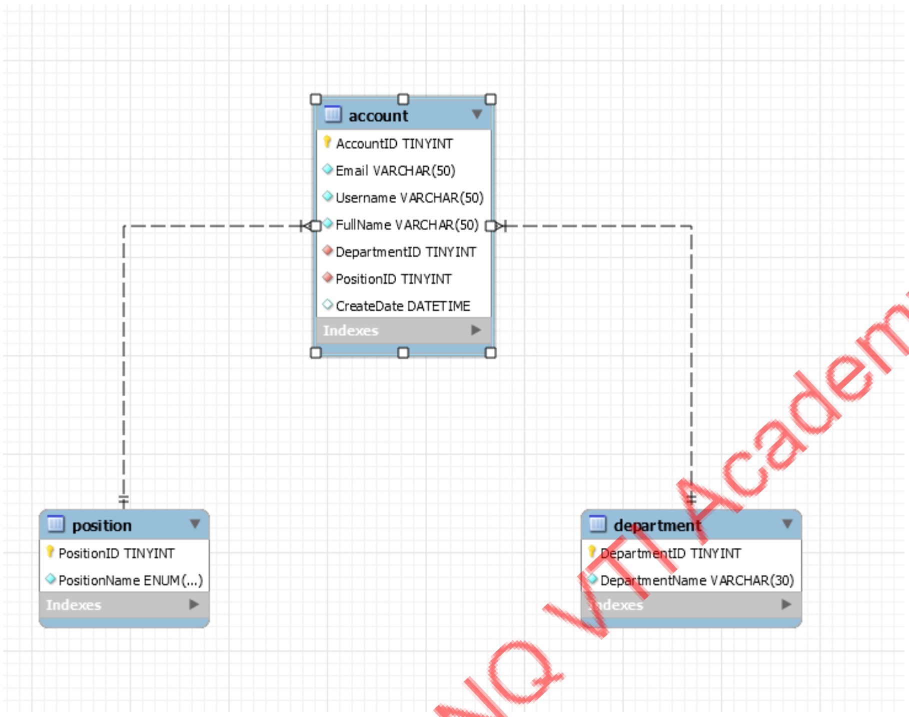
1. Hibernate One To Many (@OneToMany, @ManyToOne)

CreareTableAndInsertData.sql



- Tạo DB TestingSystem để Demo chương trình. Gồm 3 bảng Account, Possition, Department. Quan hệ giữa các bảng này như sau:

- 1 Department có nhiều Account.

- 1 Possition có nhiều Account.

```sql
-- create table 1: 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 2: Positition
DROP TABLE IF EXISTS Position;
CREATE TABLE `Position` (
        PositionID              TINYINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    PositionName                ENUM('Dev','Test','Scrum Master','PM') NOT NULL UNIQUE KEY
);


-- create table 3: Account
DROP TABLE IF EXISTS `Account`;
CREATE TABLE `Account`(
        AccountID                       TINYINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    Email                               VARCHAR(50) NOT NULL UNIQUE KEY,
    Username                            VARCHAR(50) NOT NULL UNIQUE KEY,
    FullName                            NVARCHAR(50) NOT NULL,
    DepartmentID                        TINYINT UNSIGNED NOT NULL,
    PositionID                          TINYINT UNSIGNED NOT NULL,
    CreateDate                          DATETIME DEFAULT NOW(),
    FOREIGN KEY(DepartmentID) REFERENCES Department(DepartmentID),
    FOREIGN KEY(PositionID) REFERENCES `Position`(PositionID)
);
```

VTI_Lab6_DemoHibernate_ForeginKey.rar

Config Table Account:

```
    @ManyToOne
    @JoinColumn(name = "DepartmentID", nullable = false)
    @Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
    private Department department;
```

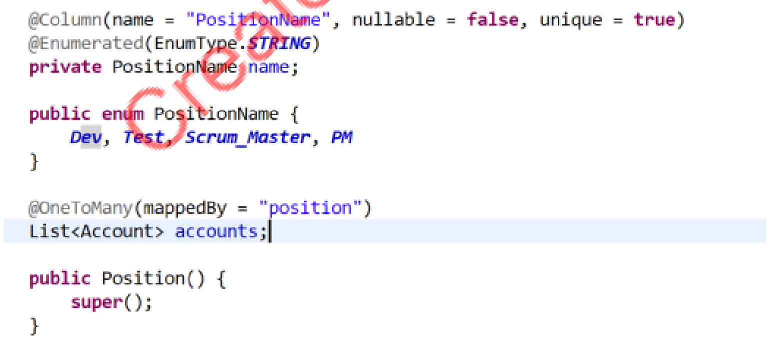
```
        @ManyToOne
        @JoinColumn(name = "DepartmentID", nullable = false)
        @Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
        private Department department;
```

Config Table Department:

```
  @OneToMany(mappedBy = "department", fetch = FetchType.EAGER)
  @Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
  private List<Account> account;
```

```
        @OneToMany(mappedBy = "department", fetch = FetchType.EAGER)
        @Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
        private List<Account> account;
```

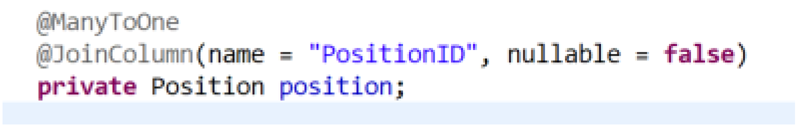
Config Table Possition



```
  @Column(name = "PositionName", nullable = false, unique = true)
  @Enumerated(EnumType.STRING)
  private PositionName name;

  public enum PositionName {
      Dev, Test, Scrum_Master, PM
  }

  @OneToMany(mappedBy = "position")
  List<Account> accounts;

  public Position() {
      super();
  }
```

```
        @Column(name = "PositionName", nullable = false, unique = true)
        @Enumerated(EnumType.STRING)
        private PositionName name;

        public enum PositionName {
              Dev, Test, Scrum_Master, PM
        }

        @OneToMany(mappedBy = "position")
        List<Account> accounts;
```

Config Table Account



```
  @ManyToOne
  @JoinColumn(name = "PositionID", nullable = false)
  private Position position;
```

```
        @ManyToOne
        @JoinColumn(name = "PositionID", nullable = false)
        private Position position;
```

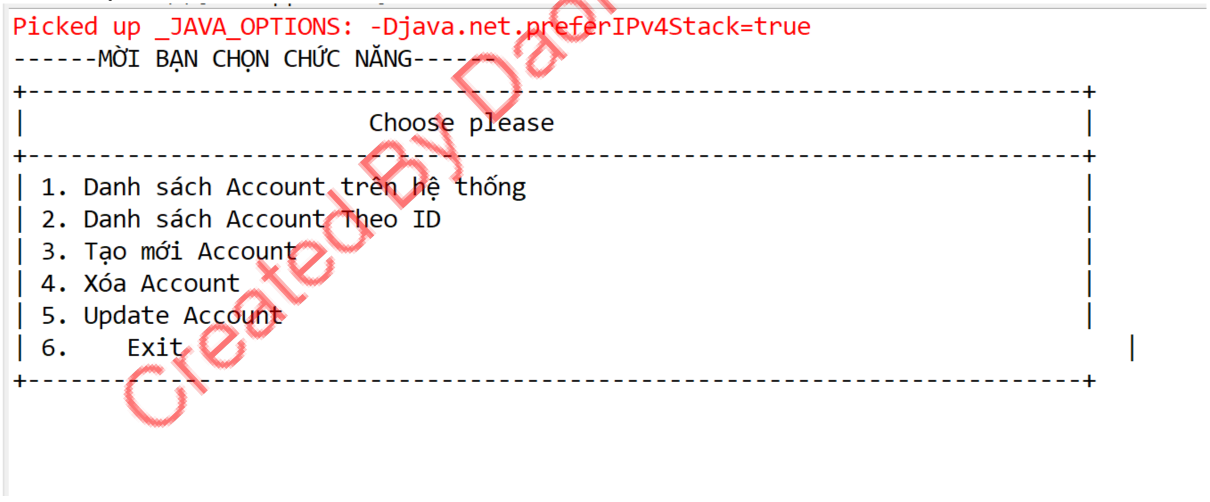
2. Thực hiện Demo trên 2 bảng Account và Department trước.

RainlWay9_10_TesttingSystem_1_4_Code_Demo_Lombok.rar

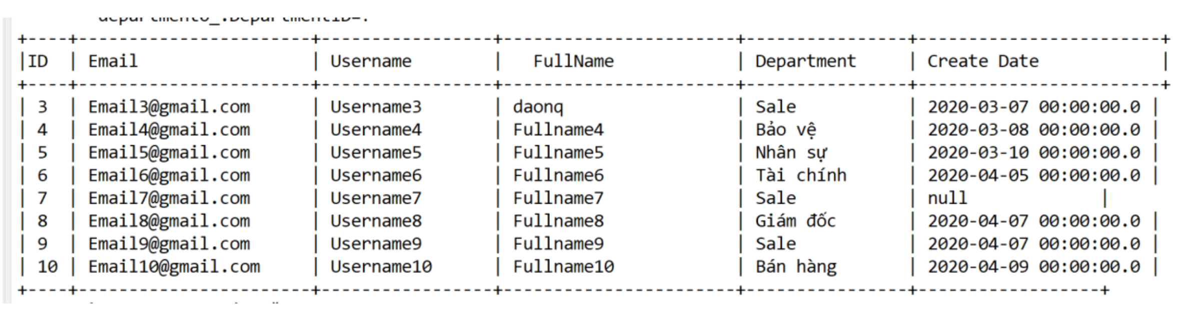
Có thể sử dụng Lombok để tích hợp tối ưu các đoạn code.

Giao diện chính chương trình

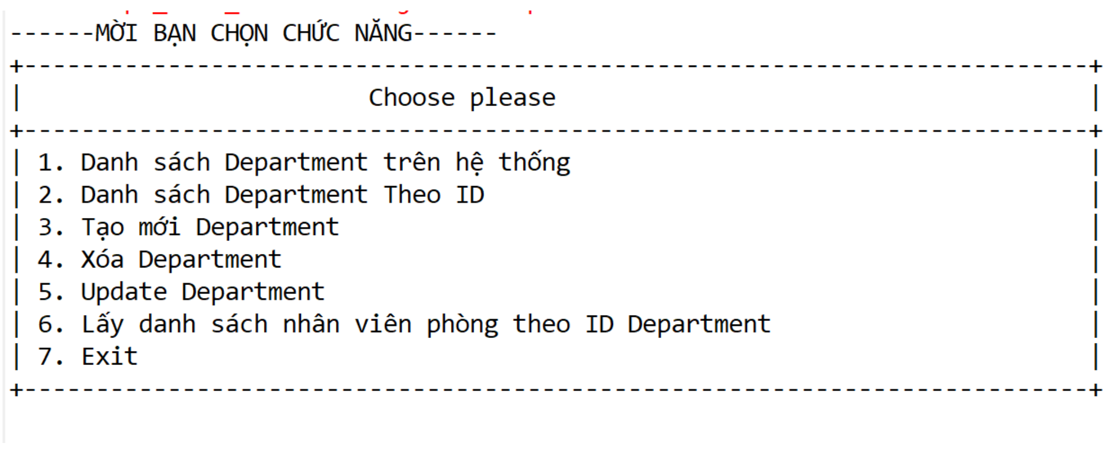
Giao diện chính:



```
Picked up _JAVA_OPTIONS: -Djava.net.preferIPv4Stack=true
------MỜI BẠN CHỌN CHỨC NĂNG------
+----------------------------------------------------------------------------+
|                              Choose please                                 |
+----------------------------------------------------------------------------+
| 1. Danh sách Account trên hệ thống                                         |
| 2. Danh sách Account Theo ID                                               |
| 3. Tạo mới Account                                                         |
| 4. Xóa Account                                                             |
| 5. Update Account                                                          |
| 6.    Exit                                                                    |
+----------------------------------------------------------------------------+
```



```
+----+----------------------+--------------+------------------+-------------+---------------------+
|ID  | Email                | Username     |  FullName        | Department  | Create Date         |
+----+----------------------+--------------+------------------+-------------+---------------------+
| 3  | Email3@gmail.com     | Username3    | daonq            | Sale        | 2020-03-07 00:00:00.0 |
| 4  | Email4@gmail.com     | Username4    | Fullname4        | Bảo vệ      | 2020-03-08 00:00:00.0 |
| 5  | Email5@gmail.com     | Username5    | Fullname5        | Nhân sự     | 2020-03-10 00:00:00.0 |
| 6  | Email6@gmail.com     | Username6    | Fullname6        | Tài chính   | 2020-04-05 00:00:00.0 |
| 7  | Email7@gmail.com     | Username7    | Fullname7        | Sale        | null                |
| 8  | Email8@gmail.com     | Username8    | Fullname8        | Giám đốc    | 2020-04-07 00:00:00.0 |
| 9  | Email9@gmail.com     | Username9    | Fullname9        | Sale        | 2020-04-07 00:00:00.0 |
| 10 | Email10@gmail.com    | Username10   | Fullname10       | Bán hàng    | 2020-04-09 00:00:00.0 |
+----+----------------------+--------------+------------------+-------------+---------------------+
```

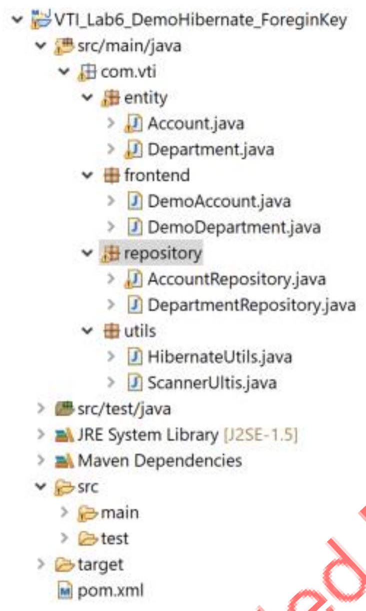


```
------MỜI BẠN CHỌN CHỨC NĂNG------
+---------------------------------------------------------------------------+
|                          Choose please                                    |
+---------------------------------------------------------------------------+
| 1. Danh sách Department trên hệ thống                                     |
| 2. Danh sách Department Theo ID                                           |
| 3. Tạo mới Department                                                     |
| 4. Xóa Department                                                         |
| 5. Update Department                                                      |
| 6. Lấy danh sách nhân viên phòng theo ID Department                       |
| 7. Exit                                                                   |
+---------------------------------------------------------------------------+
```

Tạo khung chương trình:

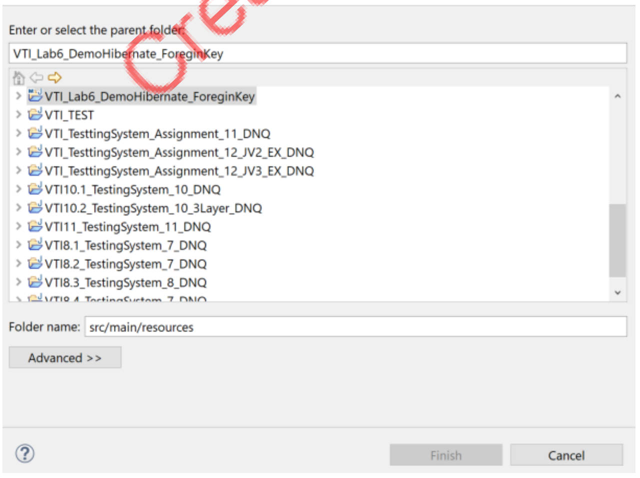
Tạo mới 1 project: VTI_Lab6_DemoHibernate_ForeginKey

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



```
VTI_Lab6_DemoHibernate_ForeginKey
  src/main/java
    com.vti
      entity
        Account.java
        Department.java
      frontend
        DemoAccount.java
        DemoDepartment.java
      repository
        AccountRepository.java
        DepartmentRepository.java
      utils
        HibernateUtils.java
        ScannerUltis.java
  src/test/java
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
    test
  target
  pom.xml
```

Tạo thư mục src/main/resources:



Enter or select the parent folder:

VTI_Lab6_DemoHibernate_ForeginKey

- VTI_Lab6_DemoHibernate_ForeginKey
- VTI_TEST
- VTI_TesttingSystem_Assignment_11_DNQ
- VTI_TesttingSystem_Assignment_12_JV2_EX_DNQ
- VTI_TesttingSystem_Assignment_12_JV3_EX_DNQ
- VTI10.1_TestingSystem_10_DNQ
- VTI10.2_TestingSystem_10_3Layer_DNQ
- VTI11_TestingSystem_11_DNQ
- VTI8.1_TestingSystem_7_DNQ
- VTI8.2_TestingSystem_7_DNQ
- VTI8.3_TestingSystem_8_DNQ

Folder name: src/main/resources

Tạo file: hibernate.cfg.xml

```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>
```

hibernate.cfg.xml

Sửa file: pom.xml

```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>
```

pom.xml

Tạo file: TestingSystem.sql

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.Department;

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);

	}

	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();
        }

}
```

HibernateUtils.java

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: ");
                        }

                }
        }

}
```

ScannerUltis.java

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.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.Formula;

@Entity
@Table(name = "`Account`", catalog = "TestingSystem")
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 = "FullName", length = 50, nullable = false)
	private String fullname;

	@ManyToOne
	@JoinColumn(name = "DepartmentID", nullable = false)
	@Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
	private Department department;

	@Column(name = "CreateDate")
	@Temporal(TemporalType.TIMESTAMP)
	@CreationTimestamp
	private Date createDate;

	public Account() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the email
	 */
	public String getEmail() {
		return email;
	}

	/**
	 * @param email the email to set
	 */
	public void setEmail(String email) {
		this.email = email;
	}

	/**
	 * @return the username
	 */
	public String getUsername() {
		return username;
	}

	/**
	 * @param username the username to set
	 */
	public void setUsername(String username) {
		this.username = username;
	}

	/**
	 * @return the fullname
	 */
	public String getFullname() {
		return fullname;
	}
```

```
        /**
         * @param fullname the fullname to set
         */
        public void setFullname(String fullname) {
                this.fullname = fullname;
        }

        /**
         * @return the department
         */
        public Department getDepartment() {
                return department;
        }

        /**
         * @param department the department to set
         */
        public void setDepartment(Department department) {
                this.department = department;
        }

        /**
         * @return the createDate
         */
        public Date getCreateDate() {
                return createDate;
        }

        /**
         * @param createDate the createDate to set
         */
        public void setCreateDate(Date createDate) {
                this.createDate = createDate;
        }

        @Override
        public String toString() {
                return "Account [id=" + id + ", email=" + email + ", username=" + username + ",
fullname=" + fullname
                                + ", department=" + department + ", createDate=" + createDate +
"]";
        }

}
```

Account.java

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.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name = "Department", catalog = "TestingSystem")
```

```java
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)
	@Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
	private List<Account> account;

	public Department() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the account
	 */
	public List<Account> getAccount() {
		return account;
	}

	/**
	 * @param account the account to set
	 */
	public void setAccount(List<Account> account) {
		this.account = account;
	}

	@Override
	public String toString() {
		return "Department [id=" + id + ", name=" + name + "]";
	}



}
```

Department.java

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.entity.Department;
import com.vti.utils.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();
			}
		}
	}

	public Account getAccountByID(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// get department by id
			Account account = session.get(Account.class, id);

			return account;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	@SuppressWarnings("unchecked")
	public Account getAccountByName(String name) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
```

```
				Query<Account> query = session.createQuery("FROM Account WHERE name = 
:nameParameter");

				// set parameter
				query.setParameter("nameParameter", name);

				// get result
				Account account = query.uniqueResult();

				return account;

			} finally {
				if (session != null) {
					session.close();
				}
			}
		}

		public void createAccount(Account account) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();
				session.beginTransaction();

				// create
				session.save(account);

				session.getTransaction().commit();
			} finally {
				if (session != null) {
					session.close();
				}
			}
		}

		public void updateAccount_FullName(short id, String newName) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();
				session.beginTransaction();

				// get department
				Account account = (Account) session.load(Account.class, id);

				// update
				account.setFullname(newName);

				session.getTransaction().commit();

			} finally {
				if (session != null) {
					session.close();
				}
			}
		}

		public void updateAccount(Account account) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();
				session.beginTransaction();
```

```
                        // update
                        session.update(account);

                        session.getTransaction().commit();
                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
        }

        public void deleteAccount(short id) {

                Session session = null;

                try {

                        // get session
                        session = hibernateUtils.openSession();
                        session.beginTransaction();

                        // get department
                        Account account = (Account) session.load(Account.class, id);

                        // delete
                        session.delete(account);

                        session.getTransaction().commit();

                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
        }

        public boolean isAccountExistsByID(short id) {

                // get department
                Account account = getAccountByID(id);

                // return result
                if (account == null) {
                        return false;
                }

                return true;
        }

        public boolean isAccountExistsByName(String name) {
                Account account = getAccountByName(name);

                if (account == null) {
                        return false;
                }
                return true;
        }

}
```

AccountRepository.java

Tạo Class DepartmentRepository trong Repository:

```
package com.vti.repository;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.query.Query;
import com.vti.entity.Department;
import com.vti.utils.HibernateUtils;

public class DepartmentRepository {
	private HibernateUtils hibernateUtils;

	public DepartmentRepository() {
		hibernateUtils = HibernateUtils.getInstance();
	}

	@SuppressWarnings("unchecked")
	public List<Department> getAllDepartment() {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
			Query<Department> query = session.createQuery("FROM Department order by 
id");

			return query.list();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public Department getDepartmentByID(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// get department by id
			Department department = session.get(Department.class, id);

			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	@SuppressWarnings("unchecked")
	public Department getDepartmentByName(String name) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
			Query<Department> query = session.createQuery("FROM Department WHERE name = 
:nameParameter");

			// set parameter
			query.setParameter("nameParameter", name);

			// get result
			Department department = query.uniqueResult();
			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void createDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// create
			session.save(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(short id, String newName) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Department department = (Department) session.load(Department.class, id);

			// update
			department.setName(newName);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// update
			session.update(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
```

```
        }
    }

    public void deleteDepartment(short id) {

        Session session = null;

        try {

            // get session
            session = hibernateUtils.openSession();
            session.beginTransaction();

            // get department
            Department department = (Department) session.load(Department.class, id);

            // delete
            session.delete(department);

            session.getTransaction().commit();

        } finally {
            if (session != null) {
                session.close();
            }
        }
    }

    public boolean isDepartmentExistsByID(short id) {

        // get department
        Department department = getDepartmentByID(id);

        // return result
        if (department == null) {
            return false;
        }

        return true;
    }

    public boolean isDepartmentExistsByName(String name) {
        Department department = getDepartmentByName(name);

        if (department == null) {
            return false;
        }
        return true;
    }

}
```

DepartmentRepository.java

Tạo Class DemoAccount trong Frontend:

```
package com.vti.frontend;

import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.repository.AccountRepository;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoAccount {
    public static void main(String[] args) {
        while (true) {
            System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
```

```
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format("|                        Choose please
|%n");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Account trên hệ thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Account Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Account");
                        System.out.format(leftAlignFormat, "4. Xóa Account");
                        System.out.format(leftAlignFormat, "5. Update Account");
                        System.out.format(leftAlignFormat, "6.   Exit");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllAccount();
                                break;
                        case 2:
                                getAccountByID();
                                break;
                        case 3:
                                createAccount();
                                break;
                        case 4:
                                DeleteAccount();
                                break;
                        case 5:
                                updateAccount();
                                break;
                        case 6:
                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountByID() {
                System.out.println("Tìm kiếm Account theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                AccountRepository accRepository = new AccountRepository();
                Account acc = accRepository.getAccountByID((short) idFind);
                if (acc != null) {
                        String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s |
%n";
                        System.out.format(
                                        "+----+-----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                        System.out.format(
                                        "|ID  | Email                 | Username        |
FullName              | Department       | Create Date           |%n");
                        System.out.format(
                                        "+----+-----------------------+-----------------+----------
-------------+----------------+------------------------+%n");

                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(),
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getCreateDate());

                        System.out.format(
                                        "+----+-----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                } else {
                        System.out.println("Không tồn tại account này trên HT");
                }

        }

        private static void updateAccount() {
                AccountRepository accRepository = new AccountRepository();
```

```
		System.out.println("Nhập vào Id cần Update: ");
		int id = ScannerUltis.inputIntPositive();
		System.out.println("Nhập vào tên cần Updare: ");
		String newName = ScannerUltis.inputString();
		accRepository.updateAccount_FullName((short) id, newName);
		getAllAccount();

	}

	private static void DeleteAccount() {
		AccountRepository accRepository = new AccountRepository();
		int id = getIdDel();
		accRepository.deleteAccount((short) id);
	}

	private static int getIdDel() {
		AccountRepository accRepository = new AccountRepository();
		while (true) {
			System.out.println("Nhập vào ID Account cần xóa: ");
			int id = ScannerUltis.inputIntPositive();
			if (accRepository.getAccountByID((short) id) != null) {
				return id;
			} else {
				System.out.println("Không có Account này trên hệ thống, Nhập lại: 
");
			}
		}
	}

	private static void createAccount() {
		Account acc = new Account();
		System.out.println("Nhập vào Email: ");
		acc.setEmail(ScannerUltis.inputEmail());
		System.out.println("Nhập vào UserName: ");
		acc.setUsername(ScannerUltis.inputString());
		System.out.println("Nhập vào FullName: : ");
		acc.setFullname(ScannerUltis.inputString());
		System.out.println("Hãy chọn phòng nhân viên: ");
		Department dep = getDep();
		acc.setDepartment(dep);
		AccountRepository accRepository = new AccountRepository();
		accRepository.createAccount(acc);
		getAllAccount();
	}

	private static Department getDep() {
		while (true) {
			DepartmentRepository depRepository = new DepartmentRepository();
			List<Department> listDep = depRepository.getAllDepartment();
			String leftAlignFormat = "| %-6d | %-21s |%n";

			System.out.format("+--------+-----------------------+%n");
			System.out.format("|   ID   | Depament Name         |%n");
			System.out.format("+--------+-----------------------+%n");
			for (Department department : listDep) {
				System.out.format(leftAlignFormat, department.getId(), 
department.getName());
			}
			System.out.format("+--------+-----------------------+%n");
			System.out.println("Chọn phòng theo ID:");
			int chooseDep = ScannerUltis.inputIntPositive();
			Department dep = depRepository.getDepartmentByID((short) chooseDep);
			if (dep != null) {
				return dep;
			} else {
				System.out.println("Không có phòng này, hãy chọn lại: ");
			}
		}
	}

	private static void getAllAccount() {

		System.out.println("Danh sách Account trên hệ thống");
		AccountRepository accRepository = new AccountRepository();
```

```
                List<Account> listAcc = accRepository.getAllAccount();

                String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s | %n";
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+------------------------+%n");
                System.out.format(
                                "|ID  | Email                  | Username        |   FullName
| Department      | Create Date           |%n");
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+------------------------+%n");

                for (Account acc : listAcc) {
                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(),
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getCreateDate());
                }
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+------------------+%n");
        }

}
```

DemoAccount.java

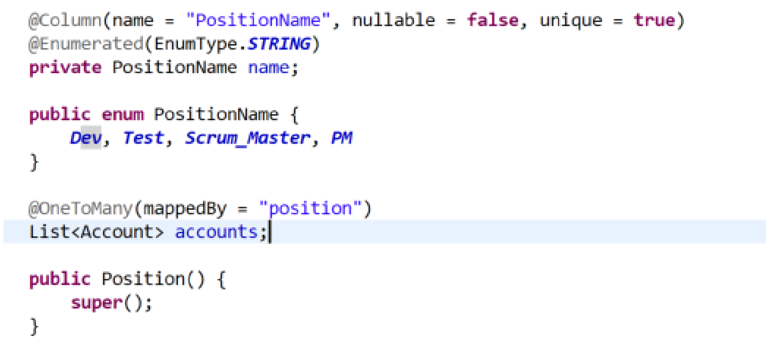
Tạo Class DemoDepartment trong Repository:

```
package com.vti.frontend;

import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoDepartment {
        public static void main(String[] args) {

                while (true) {
                        System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format("|                        Choose please
|%n");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Department trên hệ
thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Department Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Department");
                        System.out.format(leftAlignFormat, "4. Xóa Department");
                        System.out.format(leftAlignFormat, "5. Update Department");
                        System.out.format(leftAlignFormat, "6. Lấy danh sách nhân viên phòng theo
ID Department");
                        System.out.format(leftAlignFormat, "7. Exit");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllDepartment();
                                break;
                        case 2:
                                getDepartmentByID();

                                break;
                        case 3:
                                createDepartment();

                                break;
                        case 4:
                                deleteDepartment();

                                break;
                        case 5:
                                updateDepartment();

                                break;
                        case 6:
                                getAccountDepartmentByID();

                                break;
                        case 7:

                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountDepartmentByID() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int idDep = getIdUpdate();
                Department dep = depRepository.getDepartmentByID((short) idDep);
                List<Account> listAcc = dep.getAccount();
                String leftAlignFormat = "| %-6d | %-21s |%n";
                System.out.format("+--------+-----------------------+%n");
                System.out.format("|   ID   | Email                 |%n");
                System.out.format("+--------+-----------------------+%n");

                for (Account account : listAcc) {
                        System.out.format(leftAlignFormat, account.getId(), account.getEmail());
                }

                System.out.format("+--------+-----------------------+%n");
        }

        private static void updateDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int updateID = getIdUpdate();
                System.out.println("Nhập vào tên cần Updare: ");
                String newName = ScannerUltis.inputString();
                Department dep = new Department();
                dep.setId((short) updateID);
                dep.setName(newName);
                depRepository.updateDepartment(dep);
                getAllDepartment();
        }

        private static void deleteDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int updateID = getIdUpdate();
                depRepository.deleteDepartment((short) updateID);
                getAllDepartment();

        }

        private static int getIdUpdate() {
                DepartmentRepository depRepository = new DepartmentRepository();
                while (true) {
                        System.out.println("Nhập ID phòng cần thao tác: ");
                        int id = ScannerUltis.inputIntPositive();
                        Department dep = depRepository.getDepartmentByID((short) id);
                        if (dep == null) {
                                System.out.println("Không có ID này trên HT");
                        } else {
                                return id;
                        }
                }
        }

        private static void getDepartmentByID() {
                System.out.println("Tìm kiếm phòng theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                DepartmentRepository depRepository = new DepartmentRepository();
                Department depQues3 = depRepository.getDepartmentByID((short) idFind);
                if (depQues3 != null) {
                        String leftAlignFormat = "| %-6d | %-21s |%n";
                        System.out.format("+--------+-----------------------+%n");
                        System.out.format("|   ID   | Department Name       |%n");
                        System.out.format("+--------+-----------------------+%n");
                        System.out.format(leftAlignFormat, depQues3.getId(), depQues3.getName());
                        System.out.format("+--------+-----------------------+%n");
                } else {
                        System.out.println("Không tồn tại phòng này trên HT");
                }

        }

        private static void createDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                String newNameDep = getNewName();
                Department dep = new Department();
                dep.setName(newNameDep);
                depRepository.createDepartment(dep);
                depRepository.getAllDepartment();
        }

        private static String getNewName() {
                DepartmentRepository depRepository = new DepartmentRepository();
                while (true) {
                        System.out.println("Nhập vào tên phòng cần tạo: ");
                        String newName = ScannerUltis.inputString();
                        Department depQues3 = depRepository.getDepartmentByName(newName);
                        if (depQues3 != null) {
                                System.out.println("Đã có phòng trên hệ thống");
                        } else {
                                return newName;
                        }
                }
        }

        private static void getAllDepartment() {
                System.out.println("Danh sách Department trên hệ thống");
                DepartmentRepository depRepository = new DepartmentRepository();
                List<Department> listdep = depRepository.getAllDepartment();
                String leftAlignFormat = "| %-5s | %-25s |%n";
                System.out.format("+-------+---------------------------+%n");
                System.out.format("| ID    |   Department              |%n");
                System.out.format("+-------+---------------------------+%n");

                for (Department dep : listdep) {
                        System.out.format(leftAlignFormat, dep.getId(), dep.getName());
                }

        }

}
```

DemoDepartment.java

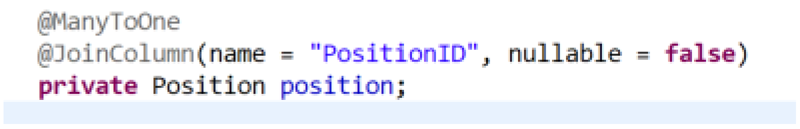
3. Thực hiện Demo trên 3 bảng Account, Possition, Department. Xử lý kiểu dữ liệu Enum. Trong chương trình này chỉ thực hiện thêm dữ liệu Enum của Possition vào phần getAllAccount khi thực hiện demo bảng Account.

Class: Possition



```java
@Column(name = "PositionName", nullable = false, unique = true)
@Enumerated(EnumType.STRING)
private PositionName name;

public enum PositionName {
    Dev, Test, Scrum_Master, PM
}

@OneToMany(mappedBy = "position")
List<Account> accounts;

public Position() {
    super();
}
```

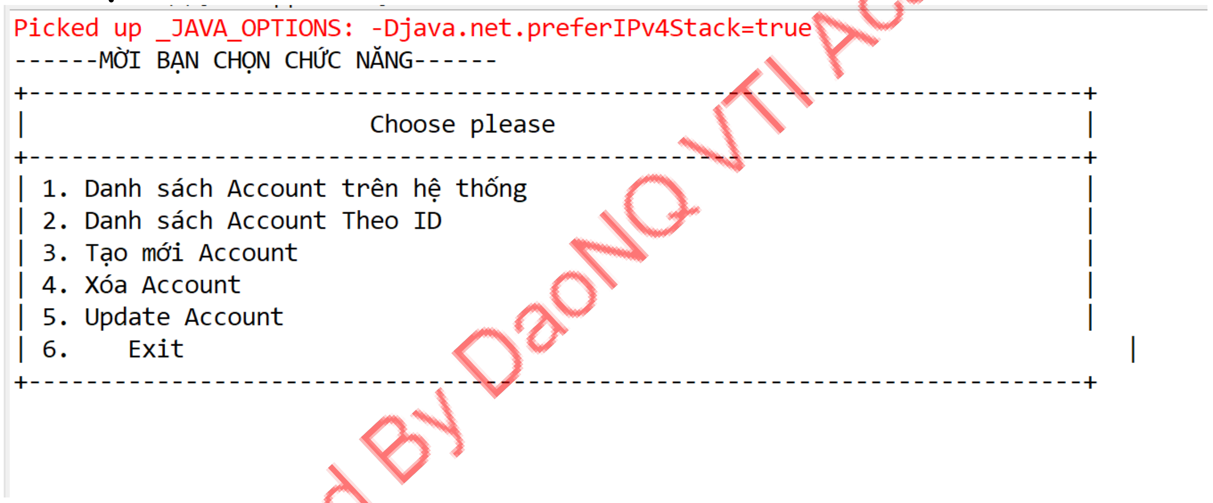
Class: Account



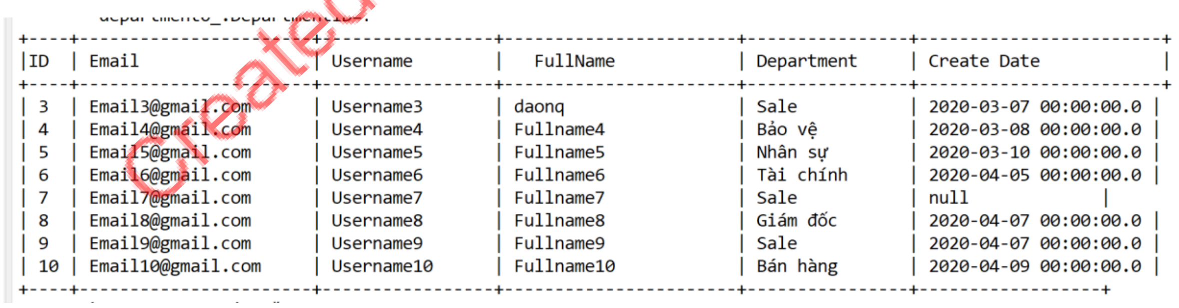
```java
@ManyToOne
@JoinColumn(name = "PositionID", nullable = false)
private Position position;
```

Giao diện chính chương trình

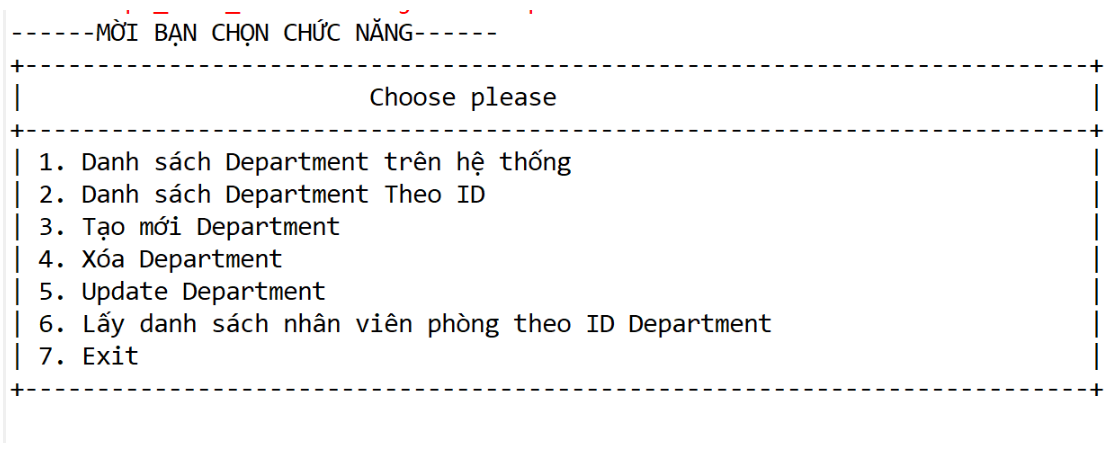
Giao diện chính:



```
Picked up _JAVA_OPTIONS: -Djava.net.preferIPv4Stack=true
------MỜI BẠN CHỌN CHỨC NĂNG------
+-----------------------------------------------------------------------------+
|                              Choose please                                  |
+-----------------------------------------------------------------------------+
| 1. Danh sách Account trên hệ thống                                          |
| 2. Danh sách Account Theo ID                                                |
| 3. Tạo mới Account                                                          |
| 4. Xóa Account                                                              |
| 5. Update Account                                                           |
| 6.    Exit                                                                     |
+-----------------------------------------------------------------------------+
```



```
+----+-------------------+-------------+-----------------+-------------+-----------------------+
|ID  | Email             | Username    |   FullName      | Department  | Create Date           |
+----+-------------------+-------------+-----------------+-------------+-----------------------+
| 3  | Email3@gmail.com  | Username3   | daonq           | Sale        | 2020-03-07 00:00:00.0 |
| 4  | Email4@gmail.com  | Username4   | Fullname4       | Bảo vệ      | 2020-03-08 00:00:00.0 |
| 5  | Email5@gmail.com  | Username5   | Fullname5       | Nhân sự     | 2020-03-10 00:00:00.0 |
| 6  | Email6@gmail.com  | Username6   | Fullname6       | Tài chính   | 2020-04-05 00:00:00.0 |
| 7  | Email7@gmail.com  | Username7   | Fullname7       | Sale        | null            |
| 8  | Email8@gmail.com  | Username8   | Fullname8       | Giám đốc    | 2020-04-07 00:00:00.0 |
| 9  | Email9@gmail.com  | Username9   | Fullname9       | Sale        | 2020-04-07 00:00:00.0 |
| 10 | Email10@gmail.com | Username10  | Fullname10      | Bán hàng    | 2020-04-09 00:00:00.0 |
+----+-------------------+-------------+-----------------+-------------+-----------------+
```

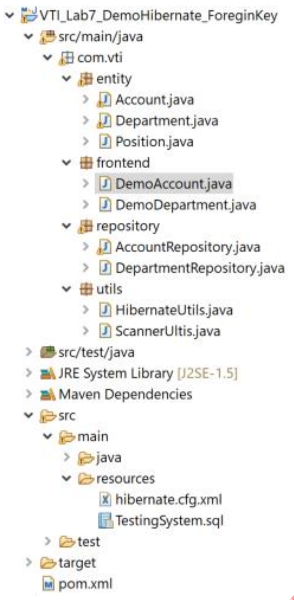


```
------MỜI BẠN CHỌN CHỨC NĂNG------
+------------------------------------------------------------------------+
|                          Choose please                                 |
+------------------------------------------------------------------------+
| 1. Danh sách Department trên hệ thống                                  |
| 2. Danh sách Department Theo ID                                        |
| 3. Tạo mới Department                                                  |
| 4. Xóa Department                                                      |
| 5. Update Department                                                   |
| 6. Lấy danh sách nhân viên phòng theo ID Department                    |
| 7. Exit                                                                |
+------------------------------------------------------------------------+
```

Tạo khung chương trình:

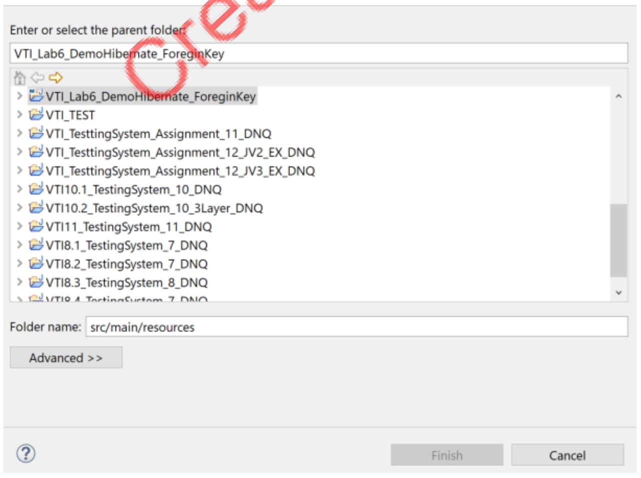
Tạo mới 1 project: VTI_Lab7_DemoHibernate_ForeginKey

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



```
VTI_Lab7_DemoHibernate_ForeginKey
  src/main/java
    com.vti
      entity
        Account.java
        Department.java
        Position.java
      frontend
        DemoAccount.java
        DemoDepartment.java
      repository
        AccountRepository.java
        DepartmentRepository.java
      utils
        HibernateUtils.java
        ScannerUltis.java
  src/test/java
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
      java
      resources
        hibernate.cfg.xml
        TestingSystem.sql
    test
  target
  pom.xml
```

Tạo thư mục src/main/resources:



```
Enter or select the parent folder:
VTI_Lab6_DemoHibernate_ForeginKey
  VTI_Lab6_DemoHibernate_ForeginKey
  VTI_TEST
  VTI_TesttingSystem_Assignment_11_DNQ
  VTI_TesttingSystem_Assignment_12_JV2_EX_DNQ
  VTI_TesttingSystem_Assignment_12_JV3_EX_DNQ
  VTI10.1_TestingSystem_10_DNQ
  VTI10.2_TestingSystem_10_3Layer_DNQ
  VTI11_TestingSystem_11_DNQ
  VTI8.1_TestingSystem_7_DNQ
  VTI8.2_TestingSystem_7_DNQ
  VTI8.3_TestingSystem_8_DNQ
Folder name: 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>
```

hibernate.cfg.xml

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>
```

pom.xml

Tạo file: TestingSystem.sql

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.Department;
import com.vti.entity.Position;

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(Position.class);
        }

        private SessionFactory buildSessionFactory() {
                if (null == sessionFactory || sessionFactory.isClosed()) {
                        ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
                                        .applySettings(configuration.getProperties()).build();

                        sessionFactory = configuration.buildSessionFactory(serviceRegistry);
                }

                return sessionFactory;
```

```java
        }

        public void closeFactory() {
                if (null != sessionFactory && sessionFactory.isOpen()) {
                        sessionFactory.close();
                }
        }

        public Session openSession() {
                buildSessionFactory();
                return sessionFactory.openSession();
        }

}
```

HibernateUtils.java

Tạo Class ScannerUltis trong package ultis:

```java
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: ");
                        }

                }
        }

}
```

ScannerUltis.java

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.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.Formula;

@Entity
@Table(name = "`Account`", catalog = "TestingSystem")
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 = "FullName", length = 50, nullable = false)
	private String fullname;

	@ManyToOne
	@JoinColumn(name = "DepartmentID", nullable = false)
	@Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
	private Department department;

	@ManyToOne
	@JoinColumn(name = "PositionID", nullable = false)
	private Position position;
	@Column(name = "CreateDate")

	@Temporal(TemporalType.TIMESTAMP)
	@CreationTimestamp
	private Date createDate;

	public Account() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the email
	 */
	public String getEmail() {
		return email;
	}

	/**
	 * @param email the email to set
	 */
	public void setEmail(String email) {
		this.email = email;
	}

	/**
	 * @return the username
	 */
	public String getUsername() {
		return username;
	}

	/**
	 * @param username the username to set
	 */
	public void setUsername(String username) {
		this.username = username;
	}

	/**
	 * @return the fullname
```

```
         */
        public String getFullname() {
                return fullname;
        }

        /**
         * @param fullname the fullname to set
         */
        public void setFullname(String fullname) {
                this.fullname = fullname;
        }

        /**
         * @return the department
         */
        public Department getDepartment() {
                return department;
        }

        /**
         * @param department the department to set
         */
        public void setDepartment(Department department) {
                this.department = department;
        }

        /**
         * @return the createDate
         */
        public Date getCreateDate() {
                return createDate;
        }

        /**
         * @param createDate the createDate to set
         */
        public void setCreateDate(Date createDate) {
                this.createDate = createDate;
        }

        /**
         * @return the position
         */
        public Position getPosition() {
                return position;
        }

        /**
         * @param position the position to set
         */
        public void setPosition(Position position) {
                this.position = position;
        }

        @Override
        public String toString() {
                return "Account [id=" + id + ", email=" + email + ", username=" + username + ",
fullname=" + fullname
                                + ", department=" + department + ", createDate=" + createDate +
"]";
        }

}
```

Account.java

Tạo Class Position 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.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "Position", catalog = "TestingSystem")
public class Position implements Serializable {
	private static final long serialVersionUID = 1L;

	@Column(name = "PositionID")
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private short id;

	@Column(name = "PositionName", nullable = false, unique = true)
	@Enumerated(EnumType.STRING)
	private PositionName name;

	public enum PositionName {
		Dev, Test, Scrum_Master, PM
	}

	@OneToMany(mappedBy = "position")
	List<Account> accounts;

	public Position() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the name
	 */
	public PositionName getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(PositionName name) {
		this.name = name;
	}

	/**
	 * @return the accounts
	 */
	public List<Account> getAccounts() {
		return accounts;
	}
	/**
	 * @param accounts the accounts to set
	 */
	public void setAccounts(List<Account> accounts) {
		this.accounts = accounts;
	}

}
```

Position.java

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.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name = "Department", catalog = "TestingSystem")
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)
	@Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
	private List<Account> account;

	public Department() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the name
```

```
         */
        public String getName() {
                return name;
        }

        /**
         * @param name the name to set
         */
        public void setName(String name) {
                this.name = name;
        }

        /**
         * @return the account
         */
        public List<Account> getAccount() {
                return account;
        }

        /**
         * @param account the account to set
         */
        public void setAccount(List<Account> account) {
                this.account = account;
        }

        @Override
        public String toString() {
                return "Department [id=" + id + ", name=" + name + "]";
        }



}
```

Department.java

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.entity.Department;
import com.vti.utils.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();
                        }
                }
        }

        public Account getAccountByID(short id) {

                Session session = null;

                try {

                        // get session
                        session = hibernateUtils.openSession();

                        // get department by id
                        Account account = session.get(Account.class, id);

                        return account;

                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
        }

        @SuppressWarnings("unchecked")
        public Account getAccountByName(String name) {

                Session session = null;

                try {

                        // get session
                        session = hibernateUtils.openSession();

                        // create hql query
                        Query<Account> query = session.createQuery("FROM Account WHERE name =
:nameParameter");

                        // set parameter
                        query.setParameter("nameParameter", name);

                        // get result
                        Account account = query.uniqueResult();

                        return account;

                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
        }

        public void createAccount(Account account) {

                Session session = null;

                try {

                        // get session
                        session = hibernateUtils.openSession();
                        session.beginTransaction();

                        // create
                        session.save(account);

                        session.getTransaction().commit();
```

```
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateAccount_FullName(short id, String newName) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Account account = (Account) session.load(Account.class, id);

			// update
			account.setFullname(newName);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateAccount(Account account) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// update
			session.update(account);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void deleteAccount(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Account account = (Account) session.load(Account.class, id);

			// delete
			session.delete(account);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
```

```
                        }
                }
        }

        public boolean isAccountExistsByID(short id) {

                // get department
                Account account = getAccountByID(id);

                // return result
                if (account == null) {
                        return false;
                }

                return true;
        }

        public boolean isAccountExistsByName(String name) {
                Account account = getAccountByName(name);

                if (account == null) {
                        return false;
                }
                return true;
        }

}
```

AccountRepository.java

Tạo Class DepartmentRepository trong Repository:

```
package com.vti.repository;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.query.Query;

import com.vti.entity.Department;
import com.vti.utils.HibernateUtils;

public class DepartmentRepository {
        private HibernateUtils hibernateUtils;

        public DepartmentRepository() {
                hibernateUtils = HibernateUtils.getInstance();
        }

        @SuppressWarnings("unchecked")
        public List<Department> getAllDepartment() {

                Session session = null;

                try {

                        // get session
                        session = hibernateUtils.openSession();

                        // create hql query
                        Query<Department> query = session.createQuery("FROM Department order by 
id");

                        return query.list();

                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
```

```
	}

	public Department getDepartmentByID(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// get department by id
			Department department = session.get(Department.class, id);

			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	@SuppressWarnings("unchecked")
	public Department getDepartmentByName(String name) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
			Query<Department> query = session.createQuery("FROM Department WHERE name = 
:nameParameter");

			// set parameter
			query.setParameter("nameParameter", name);

			// get result
			Department department = query.uniqueResult();

			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void createDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// create
			session.save(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(short id, String newName) {
		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Department department = (Department) session.load(Department.class, id);

			// update
			department.setName(newName);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// update
			session.update(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void deleteDepartment(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Department department = (Department) session.load(Department.class, id);

			// delete
			session.delete(department);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public boolean isDepartmentExistsByID(short id) {

		// get department
		Department department = getDepartmentByID(id);
```

```
                // return result
                if (department == null) {
                        return false;
                }

                return true;
        }

        public boolean isDepartmentExistsByName(String name) {
                Department department = getDepartmentByName(name);

                if (department == null) {
                        return false;
                }
                return true;
        }

}
```

DepartmentRepository.java

Tạo Class DemoAccount trong Frontend:

```
package com.vti.frontend;

import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.repository.AccountRepository;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoAccount {
        public static void main(String[] args) {
                while (true) {
                        System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format("|                        Choose please
|%n");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Account trên hệ thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Account Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Account");
                        System.out.format(leftAlignFormat, "4. Xóa Account");
                        System.out.format(leftAlignFormat, "5. Update Account");
                        System.out.format(leftAlignFormat, "6.   Exit");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllAccount();
                                break;
                        case 2:
                                getAccountByID();
                                break;
                        case 3:
                                createAccount();
                                break;
                        case 4:
                                DeleteAccount();
                                break;
                        case 5:
                                updateAccount();
                                break;
                        case 6:
                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountByID() {
                System.out.println("Tìm kiếm Account theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                AccountRepository accRepository = new AccountRepository();
                Account acc = accRepository.getAccountByID((short) idFind);
                if (acc != null) {
                        String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s |
%n";
                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                        System.out.format(
                                        "|ID  | Email                 | Username        |
FullName              | Department      | Create Date            |%n");
                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");

                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(),
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getCreateDate());

                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                } else {
                        System.out.println("Không tồn tại account này trên HT");
                }

        }

        private static void updateAccount() {
                AccountRepository accRepository = new AccountRepository();
                System.out.println("Nhập vào Id cần Update: ");
                int id = ScannerUltis.inputIntPositive();
                System.out.println("Nhập vào tên cần Updare: ");
                String newName = ScannerUltis.inputString();
                accRepository.updateAccount_FullName((short) id, newName);
                getAllAccount();

        }

        private static void DeleteAccount() {
                AccountRepository accRepository = new AccountRepository();
                int id = getIdDel();
                accRepository.deleteAccount((short) id);
        }

        private static int getIdDel() {
                AccountRepository accRepository = new AccountRepository();
                while (true) {
                        System.out.println("Nhập vào ID Account cần xóa: ");
                        int id = ScannerUltis.inputIntPositive();
                        if (accRepository.getAccountByID((short) id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có Account này trên hệ thống, Nhập lại:
");
                        }
                }
        }

        private static void createAccount() {
                Account acc = new Account();
                System.out.println("Nhập vào Email: ");
                acc.setEmail(ScannerUltis.inputEmail());
                System.out.println("Nhập vào UserName: ");
                acc.setUsername(ScannerUltis.inputString());
                System.out.println("Nhập vào FullName: : ");
                acc.setFullname(ScannerUltis.inputString());
                System.out.println("Hãy chọn phòng nhân viên: ");
                Department dep = getDep();
                acc.setDepartment(dep);
                AccountRepository accRepository = new AccountRepository();
                accRepository.createAccount(acc);
                getAllAccount();
        }

        private static Department getDep() {
                while (true) {
                        DepartmentRepository depRepository = new DepartmentRepository();
                        List<Department> listDep = depRepository.getAllDepartment();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+-----------------------+%n");
                        System.out.format("|   ID   | Depament Name         |%n");
                        System.out.format("+--------+-----------------------+%n");
                        for (Department department : listDep) {
                                System.out.format(leftAlignFormat, department.getId(), 
department.getName());
                        }
                        System.out.format("+--------+-----------------------+%n");
                        System.out.println("Chọn phòng theo ID:");
                        int chooseDep = ScannerUltis.inputIntPositive();
                        Department dep = depRepository.getDepartmentByID((short) chooseDep);
                        if (dep != null) {
                                return dep;
                        } else {
                                System.out.println("Không có phòng này, hãy chọn lại: ");
                        }
                }
        }

        private static void getAllAccount() {

                System.out.println("Danh sách Account trên hệ thống");
                AccountRepository accRepository = new AccountRepository();
                List<Account> listAcc = accRepository.getAllAccount();

                String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-14s | %-16s |
%n";
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+-----------------------+%n");
                System.out.format(
                                "|ID  | Email                   | Username         |   FullName
| Department     | Possition       | Create Date             |%n");
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+-----------------------+%n");

                for (Account acc : listAcc) {
                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(), 
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getPosition().getName(), 
acc.getCreateDate());
                }
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+-----------------+%n");
        }

}
```

DemoAccount.java

Tạo Class DemoDepartment trong Frontend:

```
package com.vti.frontend;

import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoDepartment {
        public static void main(String[] args) {

                while (true) {
                        System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+---------------------------------------------------------
-------------------+%n");
                        System.out.format("|                          Choose please                                     
|%n");
                        System.out.format("+---------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Department trên hệ 
thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Department Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Department");
                        System.out.format(leftAlignFormat, "4. Xóa Department");
                        System.out.format(leftAlignFormat, "5. Update Department");
                        System.out.format(leftAlignFormat, "6. Lấy danh sách nhân viên phòng theo 
ID Department");
                        System.out.format(leftAlignFormat, "7. Exit");
                        System.out.format("+---------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllDepartment();
                                break;
                        case 2:
                                getDepartmentByID();

                                break;
                        case 3:
                                createDepartment();

                                break;
                        case 4:
                                deleteDepartment();

                                break;
                        case 5:
                                updateDepartment();

                                break;
                        case 6:
                                getAccountDepartmentByID();

                                break;
                        case 7:

                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountDepartmentByID() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int idDep = getIdUpdate();
                Department dep = depRepository.getDepartmentByID((short) idDep);
                List<Account> listAcc = dep.getAccount();
                String leftAlignFormat = "| %-6d | %-21s |%n";
                System.out.format("+--------+-----------------------+%n");
                System.out.format("|   ID   | Email                 |%n");
                System.out.format("+--------+-----------------------+%n");

                for (Account account : listAcc) {
                        System.out.format(leftAlignFormat, account.getId(), account.getEmail());
                }

                System.out.format("+--------+-----------------------+%n");
        }

        private static void updateDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int updateID = getIdUpdate();
                System.out.println("Nhập vào tên cần Updare: ");
                String newName = ScannerUltis.inputString();
                Department dep = new Department();
                dep.setId((short) updateID);
                dep.setName(newName);
                depRepository.updateDepartment(dep);
                getAllDepartment();
        }

        private static void deleteDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int updateID = getIdUpdate();
                depRepository.deleteDepartment((short) updateID);
                getAllDepartment();

        }

        private static int getIdUpdate() {
                DepartmentRepository depRepository = new DepartmentRepository();
                while (true) {
                        System.out.println("Nhập ID phòng cần thao tác: ");
                        int id = ScannerUltis.inputIntPositive();
                        Department dep = depRepository.getDepartmentByID((short) id);
                        if (dep == null) {
                                System.out.println("Không có ID này trên HT");
                        } else {
                                return id;
                        }
                }
        }

        private static void getDepartmentByID() {
                System.out.println("Tìm kiếm phòng theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                DepartmentRepository depRepository = new DepartmentRepository();
                Department depQues3 = depRepository.getDepartmentByID((short) idFind);
                if (depQues3 != null) {
                        String leftAlignFormat = "| %-6d | %-21s |%n";
                        System.out.format("+--------+-----------------------+%n");
                        System.out.format("|   ID   | Department Name       |%n");
                        System.out.format("+--------+-----------------------+%n");
                        System.out.format(leftAlignFormat, depQues3.getId(), depQues3.getName());
                        System.out.format("+--------+-----------------------+%n");
                } else {
                        System.out.println("Không tồn tại phòng này trên HT");
                }

        }

        private static void createDepartment() {
                DepartmentRepository depRepository = new DepartmentRepository();
                String newNameDep = getNewName();
                Department dep = new Department();
                dep.setName(newNameDep);
                depRepository.createDepartment(dep);
                depRepository.getAllDepartment();
        }

        private static String getNewName() {
```

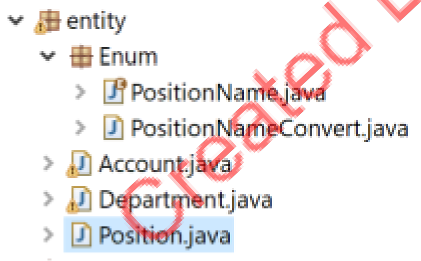
```
                DepartmentRepository depRepository = new DepartmentRepository();
                while (true) {
                        System.out.println("Nhập vào tên phòng cần tạo: ");
                        String newName = ScannerUltis.inputString();
                        Department depQues3 = depRepository.getDepartmentByName(newName);
                        if (depQues3 != null) {
                                System.out.println("Đã có phòng trên hệ thống");
                        } else {
                                return newName;
                        }
                }
        }

        private static void getAllDepartment() {
                System.out.println("Danh sách Department trên hệ thống");
                DepartmentRepository depRepository = new DepartmentRepository();
                List<Department> listdep = depRepository.getAllDepartment();
                String leftAlignFormat = "| %-5s | %-25s |%n";
                System.out.format("+-------+--------------------------+%n");
                System.out.format("| ID     |   Department              |%n");
                System.out.format("+-------+--------------------------+%n");

                for (Department dep : listdep) {
                        System.out.format(leftAlignFormat, dep.getId(), dep.getName());
                }

        }
}
```

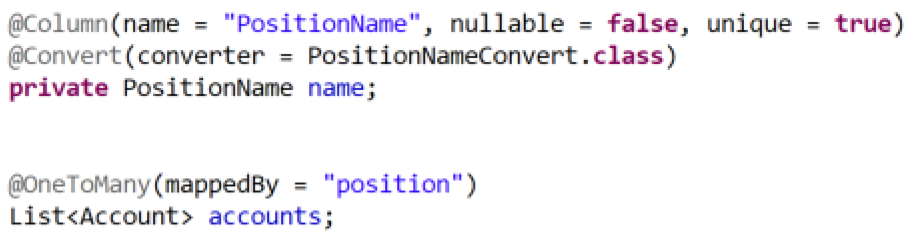
DemoDepartment.java

VTI_Lab7_DemoHibe
rnate_ForeginKey.rar

4. Thực hiện Demo kiểu dữ liệu Enum, sử dụng Converter.

Class: Possition





```
@Column(name = "PositionName", nullable = false, unique = true)
@Convert(converter = PositionNameConvert.class)
private PositionName name;


@OneToMany(mappedBy = "position")
List<Account> accounts;
```

```
        @Column(name = "PositionName", nullable = false, unique = true)
        @Convert(converter = PositionNameConvert.class)
        private PositionName name;
        @OneToMany(mappedBy = "position")
        List<Account> accounts;
```

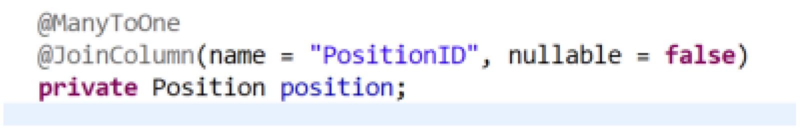
Class: PositionName

```
package com.vti.entity.Enum;

public enum PositionName {

        DEV("Dev"), TEST("Test"), SCRUM_MASTER("Scrum_Master"), PM("PM");

        private String value;

        private PositionName(String value) {
                this.value = value;
        }

        public String getValue() {
                return value;
        }

        public static PositionName of(String value) {
                if (value == null) {
                        return null;
                }

                for (PositionName name : PositionName.values()) {
                        if (name.getValue().equals(value)) {
                                return name;
                        }
                }

                return null;
        }

}
```

Class: PositionNameConvert

```
package com.vti.entity.Enum;

import javax.persistence.AttributeConverter;
import javax.persistence.Converter;

@Converter
public class PositionNameConvert implements AttributeConverter<PositionName, String> {

        @Override
        public String convertToDatabaseColumn(PositionName name) {
                if (name == null) {
                        return null;
                }

                return name.getValue();
        }

        @Override
        public PositionName convertToEntityAttribute(String value) {
                return PositionName.of(value);
        }
}
```

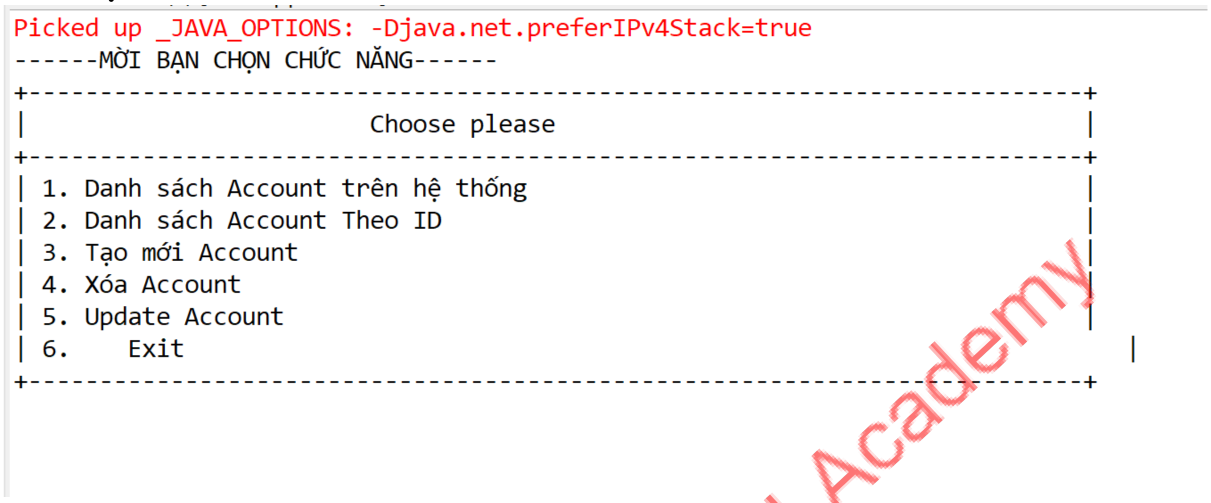
Class: Account



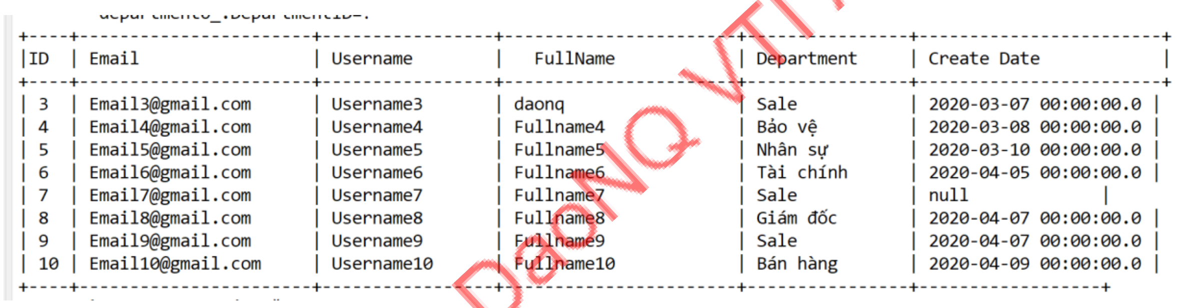
```java
@ManyToOne
@JoinColumn(name = "PositionID", nullable = false)
private Position position;
```

Giao diện chính chương trình

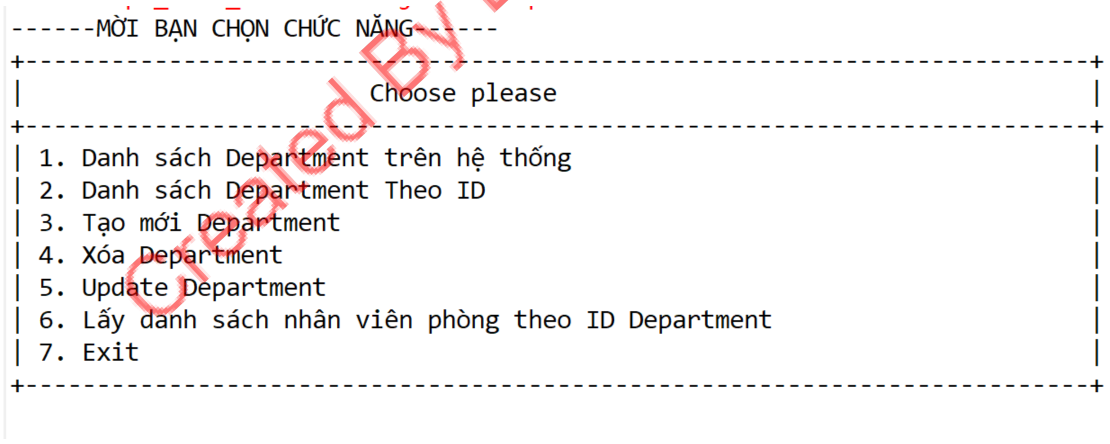
Giao diện chính:



```
Picked up _JAVA_OPTIONS: -Djava.net.preferIPv4Stack=true
------MỜI BẠN CHỌN CHỨC NĂNG------
+---------------------------------------------------------------------------+
|                         Choose please                                     |
+---------------------------------------------------------------------------+
| 1. Danh sách Account trên hệ thống                                        |
| 2. Danh sách Account Theo ID                                              |
| 3. Tạo mới Account                                                        |
| 4. Xóa Account                                                            |
| 5. Update Account                                                         |
| 6.    Exit                                                                   |
+---------------------------------------------------------------------------+
```



```
+----+-----------------------+--------------+----------------+---------------+----------------------+
|ID  | Email                 | Username     |  FullName      | Department    | Create Date          |
+----+-----------------------+--------------+----------------+---------------+----------------------+
| 3  | Email3@gmail.com      | Username3    | daonq          | Sale          | 2020-03-07 00:00:00.0 |
| 4  | Email4@gmail.com      | Username4    | Fullname4      | Bảo vệ        | 2020-03-08 00:00:00.0 |
| 5  | Email5@gmail.com      | Username5    | Fullname5      | Nhân sự       | 2020-03-10 00:00:00.0 |
| 6  | Email6@gmail.com      | Username6    | Fullname6      | Tài chính     | 2020-04-05 00:00:00.0 |
| 7  | Email7@gmail.com      | Username7    | Fullname7      | Sale          | null           |
| 8  | Email8@gmail.com      | Username8    | Fullname8      | Giám đốc      | 2020-04-07 00:00:00.0 |
| 9  | Email9@gmail.com      | Username9    | Fullname9      | Sale          | 2020-04-07 00:00:00.0 |
| 10 | Email10@gmail.com     | Username10   | Fullname10     | Bán hàng      | 2020-04-09 00:00:00.0 |
+----+-----------------------+--------------+----------------+---------------+----------------------+
```

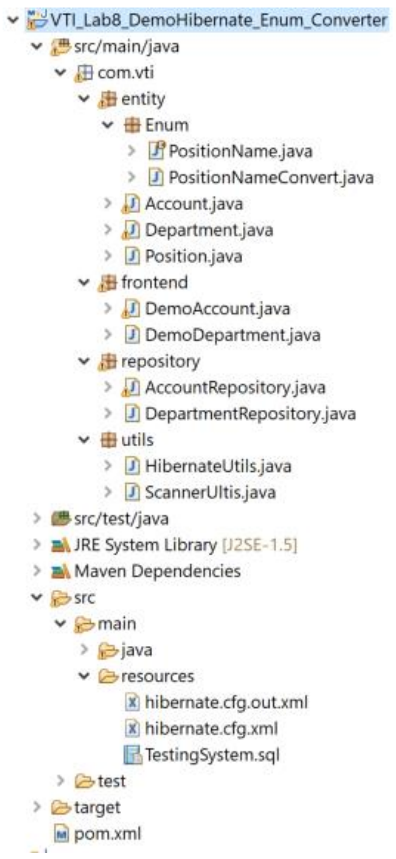


```
------MỜI BẠN CHỌN CHỨC NĂNG------
+---------------------------------------------------------------------------+
|                         Choose please                                     |
+---------------------------------------------------------------------------+
| 1. Danh sách Department trên hệ thống                                     |
| 2. Danh sách Department Theo ID                                           |
| 3. Tạo mới Department                                                     |
| 4. Xóa Department                                                         |
| 5. Update Department                                                      |
| 6. Lấy danh sách nhân viên phòng theo ID Department                       |
| 7. Exit                                                                   |
+---------------------------------------------------------------------------+
```

Tạo khung chương trình:

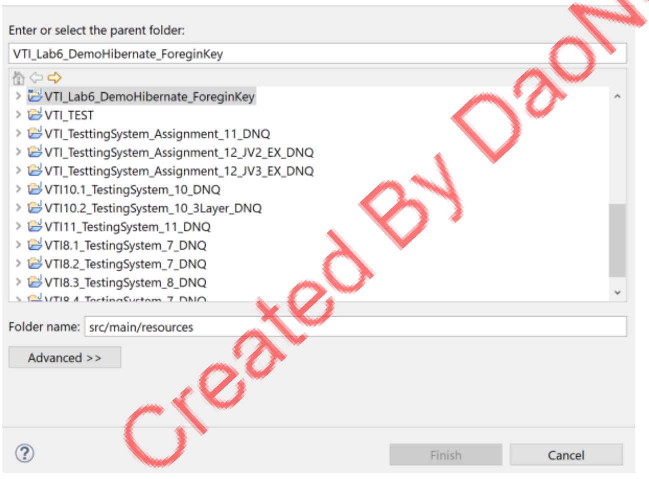
Tạo mới 1 project: VTI_Lab8_DemoHibernate_Enum_Converter

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



```
VTI_Lab8_DemoHibernate_Enum_Converter
  src/main/java
    com.vti
      entity
        Enum
          PositionName.java
          PositionNameConvert.java
        Account.java
        Department.java
        Position.java
      frontend
        DemoAccount.java
        DemoDepartment.java
      repository
        AccountRepository.java
        DepartmentRepository.java
      utils
        HibernateUtils.java
        ScannerUltis.java
  src/test/java
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
      java
      resources
        hibernate.cfg.out.xml
        hibernate.cfg.xml
        TestingSystem.sql
    test
  target
  pom.xml
```

Tạo thư mục src/main/resources:



Enter or select the parent folder:
VTI_Lab6_DemoHibernate_ForeginKey

```
VTI_Lab6_DemoHibernate_ForeginKey
VTI_TEST
VTI_TesttingSystem_Assignment_11_DNQ
VTI_TesttingSystem_Assignment_12_JV2_EX_DNQ
VTI_TesttingSystem_Assignment_12_JV3_EX_DNQ
VTI10.1_TestingSystem_10_DNQ
VTI10.2_TestingSystem_10_3Layer_DNQ
VTI11_TestingSystem_11_DNQ
VTI8.1_TestingSystem_7_DNQ
VTI8.2_TestingSystem_7_DNQ
VTI8.3_TestingSystem_8_DNQ
VTI8.4_TestingSystem_7_DNQ
```

Folder name: 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>
```

hibernate.cfg.xml

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>
```

pom.xml

Tạo file: TestingSystem.sql

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.Department;
import com.vti.entity.Position;

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(Position.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();
        }

}
```

HibernateUtils.java

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: ");
                        }

                }
        }

}
```

ScannerUltis.java

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.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.Formula;

@Entity
@Table(name = "`Account`", catalog = "TestingSystem")
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 = "FullName", length = 50, nullable = false)
        private String fullname;

        @ManyToOne
        @JoinColumn(name = "DepartmentID", nullable = false)
```

```
	@Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
	private Department department;

	@ManyToOne
	@JoinColumn(name = "PositionID", nullable = false)
	private Position position;
	@Column(name = "CreateDate")

	@Temporal(TemporalType.TIMESTAMP)
	@CreationTimestamp
	private Date createDate;

	public Account() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the email
	 */
	public String getEmail() {
		return email;
	}

	/**
	 * @param email the email to set
	 */
	public void setEmail(String email) {
		this.email = email;
	}

	/**
	 * @return the username
	 */
	public String getUsername() {
		return username;
	}

	/**
	 * @param username the username to set
	 */
	public void setUsername(String username) {
		this.username = username;
	}

	/**
	 * @return the fullname
	 */
	public String getFullname() {
		return fullname;
	}

	/**
	 * @param fullname the fullname to set
	 */
	public void setFullname(String fullname) {
		this.fullname = fullname;
	}

	/**
	 * @return the department
	 */
	public Department getDepartment() {
		return department;
	}

	/**
	 * @param department the department to set
	 */
	public void setDepartment(Department department) {
		this.department = department;
	}

	/**
	 * @return the createDate
	 */
	public Date getCreateDate() {
		return createDate;
	}

	/**
	 * @param createDate the createDate to set
	 */
	public void setCreateDate(Date createDate) {
		this.createDate = createDate;
	}

	/**
	 * @return the position
	 */
	public Position getPosition() {
		return position;
	}

	/**
	 * @param position the position to set
	 */
	public void setPosition(Position position) {
		this.position = position;
	}

	@Override
	public String toString() {
		return "Account [id=" + id + ", email=" + email + ", username=" + username + ", 
fullname=" + fullname
				+ ", department=" + department + ", createDate=" + createDate + 
"]";
	}

}
```

Account.java

Tạo thêm package com.vti.entity.Enum sau đó tạo Class PositionName trong Enum :

```
package com.vti.entity.Enum;

public enum PositionName {

	DEV("Dev"), TEST("Test"), SCRUM_MASTER("Scrum_Master"), PM("PM");

	private String value;

	private PositionName(String value) {
		this.value = value;
	}
	public String getValue() {
		return value;
	}

	public static PositionName of(String value) {
		if (value == null) {
			return null;
		}

		for (PositionName name : PositionName.values()) {
			if (name.getValue().equals(value)) {
				return name;
			}
		}

		return null;
	}

}
```

PositionName.java

Tạo Class PositionNameConverttrong Enum :

```
package com.vti.entity.Enum;

import javax.persistence.AttributeConverter;
import javax.persistence.Converter;

@Converter
public class PositionNameConvert implements AttributeConverter<PositionName, String> {

	@Override
	public String convertToDatabaseColumn(PositionName name) {
		if (name == null) {
			return null;
		}

		return name.getValue();
	}

	@Override
	public PositionName convertToEntityAttribute(String value) {
		return PositionName.of(value);
	}
}
```

PositionNameConvert.java

Tạo Class Position trong Entity:

```
package com.vti.entity;

import java.io.Serializable;
import java.util.List;
```

```java
import javax.persistence.Column;
import javax.persistence.Convert;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import com.vti.entity.Enum.PositionName;
import com.vti.entity.Enum.PositionNameConvert;

@Entity
@Table(name = "Position", catalog = "TestingSystem")
public class Position implements Serializable {
	private static final long serialVersionUID = 1L;

	@Column(name = "PositionID")
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private short id;

	@Column(name = "PositionName", nullable = false, unique = true)
	@Convert(converter = PositionNameConvert.class)
	private PositionName name;


	@OneToMany(mappedBy = "position")
	List<Account> accounts;

	public Position() {
		super();
	}

	/**
	 * @return the id
	 */
	public short getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(short id) {
		this.id = id;
	}

	/**
	 * @return the name
	 */
	public PositionName getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(PositionName name) {
		this.name = name;
	}

	/**
	 * @return the accounts
	 */
	public List<Account> getAccounts() {
		return accounts;
	}

	/**
	 * @param accounts the accounts to set
	 */
	public void setAccounts(List<Account> accounts) {
		this.accounts = accounts;
```

```
        }

}
```

Position.java

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.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name = "Department", catalog = "TestingSystem")
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)
        @Cascade(value = { CascadeType.REMOVE, CascadeType.SAVE_UPDATE })
        private List<Account> account;

        public Department() {
                super();
        }

        /**
         * @return the id
         */
        public short getId() {
                return id;
        }

        /**
         * @param id the id to set
         */
        public void setId(short id) {
                this.id = id;
        }

        /**
         * @return the name
         */
        public String getName() {
                return name;
        }

        /**
```

```
         * @param name the name to set
         */
        public void setName(String name) {
                this.name = name;
        }

        /**
         * @return the account
         */
        public List<Account> getAccount() {
                return account;
        }

        /**
         * @param account the account to set
         */
        public void setAccount(List<Account> account) {
                this.account = account;
        }

        @Override
        public String toString() {
                return "Department [id=" + id + ", name=" + name + "]";
        }



}
```

Department.java

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.entity.Department;
import com.vti.utils.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();
```

```
				}
			}
		}

		public Account getAccountByID(short id) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();

				// get department by id
				Account account = session.get(Account.class, id);

				return account;

			} finally {
				if (session != null) {
					session.close();
				}
			}
		}

		@SuppressWarnings("unchecked")
		public Account getAccountByName(String name) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();

				// create hql query
				Query<Account> query = session.createQuery("FROM Account WHERE name =
:nameParameter");

				// set parameter
				query.setParameter("nameParameter", name);

				// get result
				Account account = query.uniqueResult();

				return account;

			} finally {
				if (session != null) {
					session.close();
				}
			}
		}

		public void createAccount(Account account) {

			Session session = null;

			try {

				// get session
				session = hibernateUtils.openSession();
				session.beginTransaction();

				// create
				session.save(account);

				session.getTransaction().commit();
			} finally {
				if (session != null) {
					session.close();
				}
			}
		}
```

```
	public void updateAccount_FullName(short id, String newName) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Account account = (Account) session.load(Account.class, id);

			// update
			account.setFullname(newName);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateAccount(Account account) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// update
			session.update(account);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void deleteAccount(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Account account = (Account) session.load(Account.class, id);

			// delete
			session.delete(account);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public boolean isAccountExistsByID(short id) {
		// get department
		Account account = getAccountByID(id);

		// return result
		if (account == null) {
			return false;
		}

		return true;
	}

	public boolean isAccountExistsByName(String name) {
		Account account = getAccountByName(name);

		if (account == null) {
			return false;
		}
		return true;
	}

}
```

AccountRepository.java

Tạo Class DepartmentRepository trong Repository:

```
package com.vti.repository;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.query.Query;

import com.vti.entity.Department;
import com.vti.utils.HibernateUtils;

public class DepartmentRepository {
	private HibernateUtils hibernateUtils;

	public DepartmentRepository() {
		hibernateUtils = HibernateUtils.getInstance();
	}

	@SuppressWarnings("unchecked")
	public List<Department> getAllDepartment() {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
			Query<Department> query = session.createQuery("FROM Department order by 
id");

			return query.list();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public Department getDepartmentByID(short id) {

		Session session = null;
		try {

			// get session
			session = hibernateUtils.openSession();

			// get department by id
			Department department = session.get(Department.class, id);

			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	@SuppressWarnings("unchecked")
	public Department getDepartmentByName(String name) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();

			// create hql query
			Query<Department> query = session.createQuery("FROM Department WHERE name =
:nameParameter");

			// set parameter
			query.setParameter("nameParameter", name);

			// get result
			Department department = query.uniqueResult();

			return department;

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void createDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// create
			session.save(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(short id, String newName) {

		Session session = null;

		try {

			// get session
```

```java
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Department department = (Department) session.load(Department.class, id);

			// update
			department.setName(newName);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void updateDepartment(Department department) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// update
			session.update(department);

			session.getTransaction().commit();
		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public void deleteDepartment(short id) {

		Session session = null;

		try {

			// get session
			session = hibernateUtils.openSession();
			session.beginTransaction();

			// get department
			Department department = (Department) session.load(Department.class, id);

			// delete
			session.delete(department);

			session.getTransaction().commit();

		} finally {
			if (session != null) {
				session.close();
			}
		}
	}

	public boolean isDepartmentExistsByID(short id) {

		// get department
		Department department = getDepartmentByID(id);

		// return result
		if (department == null) {
			return false;
		}
```

```
                return true;
        }

        public boolean isDepartmentExistsByName(String name) {
                Department department = getDepartmentByName(name);

                if (department == null) {
                        return false;
                }
                return true;
        }

}
```

DepartmentRepository.java

Tạo Class DemoAccount trong Frontend: ở đây chỉ Demo trường hợp GetAllAccount để xem dữ liệu Possition

```
package com.vti.frontend;

import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.entity.Enum.PositionNameConvert;
import com.vti.repository.AccountRepository;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoAccount {
        public static void main(String[] args) {
                while (true) {
                        System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format("|                        Choose please
|%n");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Account trên hệ thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Account Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Account");
                        System.out.format(leftAlignFormat, "4. Xóa Account");
                        System.out.format(leftAlignFormat, "5. Update Account");
                        System.out.format(leftAlignFormat, "6.   Exit");
                        System.out.format("+-------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllAccount();
                                break;
                        case 2:
                                getAccountByID();
                                break;
                        case 3:
                                createAccount();
                                break;
                        case 4:
                                DeleteAccount();
                                break;
                        case 5:
                                updateAccount();
                                break;
                        case 6:
                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountByID() {
                System.out.println("Tìm kiếm Account theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                AccountRepository accRepository = new AccountRepository();
                Account acc = accRepository.getAccountByID((short) idFind);
                if (acc != null) {
                        String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s |
%n";
                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                        System.out.format(
                                        "|ID  | Email                | Username        |
FullName             | Department      | Create Date            |%n");
                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");

                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(),
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getCreateDate());

                        System.out.format(
                                        "+----+----------------------+-----------------+----------
-------------+----------------+------------------------+%n");
                } else {
                        System.out.println("Không tồn tại account này trên HT");
                }

        }

        private static void updateAccount() {
                AccountRepository accRepository = new AccountRepository();
                System.out.println("Nhập vào Id cần Update: ");
                int id = ScannerUltis.inputIntPositive();
                System.out.println("Nhập vào tên cần Updare: ");
                String newName = ScannerUltis.inputString();
                accRepository.updateAccount_FullName((short) id, newName);
                getAllAccount();

        }

        private static void DeleteAccount() {
                AccountRepository accRepository = new AccountRepository();
                int id = getIdDel();
                accRepository.deleteAccount((short) id);
        }

        private static int getIdDel() {
                AccountRepository accRepository = new AccountRepository();
                while (true) {
                        System.out.println("Nhập vào ID Account cần xóa: ");
                        int id = ScannerUltis.inputIntPositive();
                        if (accRepository.getAccountByID((short) id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có Account này trên hệ thống, Nhập lại:
");
                        }
                }
        }

        private static void createAccount() {
                Account acc = new Account();
                System.out.println("Nhập vào Email: ");
                acc.setEmail(ScannerUltis.inputEmail());
                System.out.println("Nhập vào UserName: ");
                acc.setUsername(ScannerUltis.inputString());
                System.out.println("Nhập vào FullName: : ");
                acc.setFullname(ScannerUltis.inputString());
                System.out.println("Hãy chọn phòng nhân viên: ");
                Department dep = getDep();
                acc.setDepartment(dep);
                AccountRepository accRepository = new AccountRepository();
                accRepository.createAccount(acc);
                getAllAccount();
        }

        private static Department getDep() {
                while (true) {
                        DepartmentRepository depRepository = new DepartmentRepository();
                        List<Department> listDep = depRepository.getAllDepartment();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+-----------------------+%n");
                        System.out.format("|   ID   | Depament Name         |%n");
                        System.out.format("+--------+-----------------------+%n");
                        for (Department department : listDep) {
                                System.out.format(leftAlignFormat, department.getId(),
department.getName());
                        }
                        System.out.format("+--------+-----------------------+%n");
                        System.out.println("Chọn phòng theo ID:");
                        int chooseDep = ScannerUltis.inputIntPositive();
                        Department dep = depRepository.getDepartmentByID((short) chooseDep);
                        if (dep != null) {
                                return dep;
                        } else {
                                System.out.println("Không có phòng này, hãy chọn lại: ");
                        }
                }
        }

        private static void getAllAccount() {

                System.out.println("Danh sách Account trên hệ thống");
                AccountRepository accRepository = new AccountRepository();
                List<Account> listAcc = accRepository.getAllAccount();

                String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-14s | %-16s |
%n";
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+------------------------+%n");
                System.out.format(
                                "|ID  | Email                 | Username        |   FullName
| Department      | Possition       | Create Date            |%n");
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+------------------------+%n");
//              PositionNameConvert pnc = new PositionNameConvert();
                for (Account acc : listAcc) {
                        System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(),
acc.getUsername(), acc.getFullname(),
                                        acc.getDepartment().getName(), acc.getPosition().getName(),
                                        acc.getCreateDate());
                }
                System.out.format(
                                "+----+-----------------------+-----------------+------------------
-----+----------------+----------------+------------------+%n");
        }

}
```

DemoAccount.java

Tạo Class DemoDepartment trong Repository:

```
package com.vti.frontend;
```

```
import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.repository.DepartmentRepository;
import com.vti.utils.ScannerUltis;

public class DemoDepartment {
        public static void main(String[] args) {

                while (true) {
                        System.out.println("------MỜI BẠN CHỌN CHỨC NĂNG------");
                        String leftAlignFormat = "| %-72s |%n";
                        System.out.format("+--------------------------------------------------------
-------------------+%n");
                        System.out.format("|                        Choose please
|%n");
                        System.out.format("+--------------------------------------------------------
-------------------+%n");
                        System.out.format(leftAlignFormat, "1. Danh sách Department trên hệ
thống");
                        System.out.format(leftAlignFormat, "2. Danh sách Department Theo ID");
                        System.out.format(leftAlignFormat, "3. Tạo mới Department");
                        System.out.format(leftAlignFormat, "4. Xóa Department");
                        System.out.format(leftAlignFormat, "5. Update Department");
                        System.out.format(leftAlignFormat, "6. Lấy danh sách nhân viên phòng theo
ID Department");
                        System.out.format(leftAlignFormat, "7. Exit");
                        System.out.format("+--------------------------------------------------------
-------------------+%n");
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                getAllDepartment();
                                break;
                        case 2:
                                getDepartmentByID();

                                break;
                        case 3:
                                createDepartment();

                                break;
                        case 4:
                                deleteDepartment();

                                break;
                        case 5:
                                updateDepartment();

                                break;
                        case 6:
                                getAccountDepartmentByID();

                                break;
                        case 7:

                                return;
                        default:
                                System.out.println("Nhập lại:");
                                break;
                        }
                }
        }

        private static void getAccountDepartmentByID() {
                DepartmentRepository depRepository = new DepartmentRepository();
                int idDep = getIdUpdate();
                Department dep = depRepository.getDepartmentByID((short) idDep);
                List<Account> listAcc = dep.getAccount();
                String leftAlignFormat = "| %-6d | %-21s |%n";
                System.out.format("+--------+-----------------------+%n");
                System.out.format("|   ID   | Email                 |%n");
                System.out.format("+--------+-----------------------+%n");
```

```
		for (Account account : listAcc) {
			System.out.format(leftAlignFormat, account.getId(), account.getEmail());
		}

		System.out.format("+--------+----------------------+%n");
	}

	private static void updateDepartment() {
		DepartmentRepository depRepository = new DepartmentRepository();
		int updateID = getIdUpdate();
		System.out.println("Nhập vào tên cần Updare: ");
		String newName = ScannerUltis.inputString();
		Department dep = new Department();
		dep.setId((short) updateID);
		dep.setName(newName);
		depRepository.updateDepartment(dep);
		getAllDepartment();
	}

	private static void deleteDepartment() {
		DepartmentRepository depRepository = new DepartmentRepository();
		int updateID = getIdUpdate();
		depRepository.deleteDepartment((short) updateID);
		getAllDepartment();

	}

	private static int getIdUpdate() {
		DepartmentRepository depRepository = new DepartmentRepository();
		while (true) {
			System.out.println("Nhập ID phòng cần thao tác: ");
			int id = ScannerUltis.inputIntPositive();
			Department dep = depRepository.getDepartmentByID((short) id);
			if (dep == null) {
				System.out.println("Không có ID này trên HT");
			} else {
				return id;
			}
		}
	}

	private static void getDepartmentByID() {
		System.out.println("Tìm kiếm phòng theo ID: ");
		System.out.println("Nhập vào ID cần tìm kiếm: ");
		int idFind = ScannerUltis.inputIntPositive();
		DepartmentRepository depRepository = new DepartmentRepository();
		Department depQues3 = depRepository.getDepartmentByID((short) idFind);
		if (depQues3 != null) {
			String leftAlignFormat = "| %-6d | %-21s |%n";
			System.out.format("+--------+----------------------+%n");
			System.out.format("|   ID   | Department Name      |%n");
			System.out.format("+--------+----------------------+%n");
			System.out.format(leftAlignFormat, depQues3.getId(), depQues3.getName());
			System.out.format("+--------+----------------------+%n");
		} else {
			System.out.println("Không tồn tại phòng này trên HT");
		}

	}

	private static void createDepartment() {
		DepartmentRepository depRepository = new DepartmentRepository();
		String newNameDep = getNewName();
		Department dep = new Department();
		dep.setName(newNameDep);
		depRepository.createDepartment(dep);
		depRepository.getAllDepartment();
	}

	private static String getNewName() {
		DepartmentRepository depRepository = new DepartmentRepository();
		while (true) {
			System.out.println("Nhập vào tên phòng cần tạo: ");
```

```
            String newName = ScannerUltis.inputString();
            Department depQues3 = depRepository.getDepartmentByName(newName);
            if (depQues3 != null) {
                System.out.println("Đã có phòng trên hệ thống");
            } else {
                return newName;
            }
        }
    }

    private static void getAllDepartment() {
        System.out.println("Danh sách Department trên hệ thống");
        DepartmentRepository depRepository = new DepartmentRepository();
        List<Department> listdep = depRepository.getAllDepartment();
        String leftAlignFormat = "| %-5s | %-25s |%n";
        System.out.format("+-------+---------------------------+%n");
        System.out.format("| ID    |   Department              |%n");
        System.out.format("+-------+---------------------------+%n");

        for (Department dep : listdep) {
            System.out.format(leftAlignFormat, dep.getId(), dep.getName());
        }

    }

}
```

DemoDepartment.java

VTI_Lab8_DemoHibe
rnate_Enum_Convert