✅ Case Study Title: Citizen and Passport Management System

🎯 Business Context:

A national government agency maintains records of citizens and the passports issued to them. The rule of the system is:

- **Each citizen can hold exactly one passport**
- **Each passport must be assigned to only one citizen**

This kind of relationship is a textbook example of a **One-to-One association**, where **one record in the Citizen table corresponds to one record in the Passport table**, and vice versa.

📘 Objective:

To design and implement a Hibernate-based application using **One-to-One mapping** between two entities:

1. **Citizen**
2. **Passport**

This application should be capable of:

- Creating a citizen and passport record together
- Retrieving citizen and their associated passport
- Maintaining referential integrity between the two

📐 Entity Design:

1. Citizen Entity

- Represents the individual citizen.
- Fields: `id`, `name`, and a reference to their **Passport**.
- Establishes a **foreign key relationship** with the Passport entity.

2. Passport Entity

- Represents the government-issued passport.
- Fields: `id`, `passportNumber`, and optionally a back-reference to the **Citizen**.

🗂️ Project Folder Structure

```
CitizenPassportHibernate/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           ├── app/
│   │   │           │   └── App.java
│   │   │           ├── entity/
│   │   │           │   ├── Citizen.java
│   │   │           │   └── Passport.java
│   │   │           └── util/
│   │   │               └── HibernateUtil.java
│   │   └── resources/
│   │       └── hibernate.cfg.xml
├── pom.xml
```

🧭 Mapping Strategy:

Hibernate supports multiple ways to implement One-to-One relationships. In this case study, we use the **foreign key association** strategy:

- The `Citizen` table will have a foreign key column `passport_id`, referencing the primary key of the `Passport`table.
- The mapping ensures that one citizen is linked to one passport.
- Cascade operations are used so that when a Citizen is saved, the corresponding Passport is automatically persisted.

🔄 Relationship Flow:

- When a **new Citizen** object is created, a **Passport** object is also created and associated with the citizen.
- On saving the Citizen entity, both the Citizen and Passport records are inserted into the database in a single transaction.
- When retrieving a Citizen, Hibernate also loads the associated Passport (depending on fetch type).

🔐 Data Integrity:

- Enforced through **foreign key constraint** in the database.
- Hibernate manages the **referential integrity** via annotations and session transactions.
- The relationship prevents orphan Passport records from existing without a corresponding Citizen.

🔧 Technical Requirements:

- **Hibernate ORM** (version 6+)
- **Jakarta Persistence API (JPA)** (version 3.1 or compatible)
- **MySQL database**
- **Maven** for dependency management
- **Eclipse IDE** or IntelliJ for development

📂 Files & Configuration:

The application includes:

- Entity classes for `Citizen` and `Passport`
- Hibernate configuration file with database details
- A utility class to bootstrap Hibernate
- A main application class to create and retrieve entities

App.java

```
package com.example.app;

import org.hibernate.Session;
import org.hibernate.Transaction;

import com.example.entity.Citizen;
import com.example.entity.Passport;
import com.example.util.HibernateUtil;

public class App {
    public static void main(String[] args) {
        Passport passport = new Passport("A12345678");
        Citizen citizen = new Citizen("John Doe", passport);

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        session.persist(citizen);

        tx.commit();
        session.close();

    }
}
```

//Entity
Citizen.java:

```
package com.example.entity;

import jakarta.persistence.*;

@Entity
public class Citizen {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "passport_id")
    private Passport passport;

    public Citizen() {}

    public Citizen(String name, Passport passport) {
        this.name = name;
        this.passport = passport;
    }
```

```
    // Getters and Setters
    public int getId() { return id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public Passport getPassport() { return passport; }
    public void setPassport(Passport passport) { this.passport = passport; }
}
```

//entity

Passport.java

```
package com.example.entity;

import jakarta.persistence.*;

@Entity
public class Passport {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String passportNumber;

    public Passport() {}

    public Passport(String passportNumber) {
        this.passportNumber = passportNumber;
    }

    // Getters and Setters
    public int getId() { return id; }

    public String getPassportNumber() { return passportNumber; }
    public void setPassportNumber(String passportNumber) { this.passportNumber = passportNumber;
}
}
```

HibernateUtil.java

```
package com.example.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import com.example.entity.Citizen;
import com.example.entity.Passport;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;
```

```
    static {
        try {
            sessionFactory = new Configuration()
                .configure("hibernate.cfg.xml")
                .addAnnotatedClass(Citizen.class)
                .addAnnotatedClass(Passport.class)
                .buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("SessionFactory creation failed: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-configuration>
    <session-factory>
                <!-- Database connection settings -->
            <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
            <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/citizen_db</property>
            <property name="hibernate.connection.username">root</property>
            <property name="hibernate.connection.password">PASS@word1</property>


            <!-- JDBC Connection pool -->
            <property name="connection.pool_size">10</property>

            <!-- SQL dialect -->
            <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

            <!-- logging -->
            <property name="show_sql">true</property>

            <!-- Automatically create/update table -->
            <property name="hbm2ddl.auto">update</property>

        <mapping class="com.example.entity.Citizen"/>
                <mapping class="com.example.entity.Passport"/>

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

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.example</groupId>
  <artifactId>CitizenPasswortHibernate</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <dependency>
        <groupId>org.hibernate.orm</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>7.0.8.Final</version>
    </dependency>
    <dependency>
        <groupId>jakarta.persistence</groupId>
        <artifactId>jakarta.persistence-api</artifactId>
        <version>3.2.0</version>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>9.3.0</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>2.0.12</version>
    </dependency>
</dependencies>


</project>
```