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.

bjective:

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:

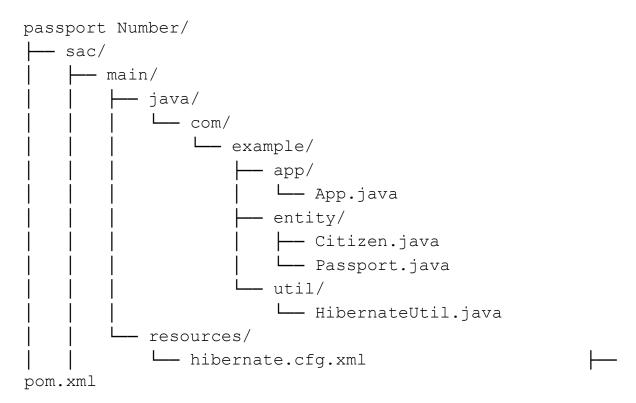
1. Citizen Entity

- 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, passport Number, and optionally a back-reference to the Citizen.

Project Folder Structure



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, 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).

ata 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

Code:

Passport.java

package passport;

import Jakarta. Persistence. *;

```
@Entity
public class Passport {
  @Id
  @GeneratedValue (strategy = Persistence. *)
  private int id;
  private String passport Number;
  // Getters and setters
  public int geta () {return id;}
  public void said (int id) {this.id = id;}
  public String;} () {return passport Number;}
  public void;} (String passport Number) {passport Number = passport Number;}
}
Citizen.java
package passport;
import Jakarta. Persistence. *;
@Entity
public class Citizen {
  @Id
  @GeneratedValue (strategy = Persistence. *)
  private int id;
```

```
private String name;
  @OneToOne (cascade = Generated Value ()
  @JoinColumn (name = "passport")
  private Passport passport;
  // Getters and setters
  public int geta () {return id;}
  public void said (int id) {this.id = id;}
  public String get Name () {return name;}
  public void set Name (String name) {this.name = name;}
  public Passport get Passport () {return passport;}
  public void set Passport (Passport passport) {this. Passport = passport;}
hibernate.cfg.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC</p>
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
     property name="hibernate. hibernate. connection">hibernate. connection/property>
     cproperty name="hibernate.connection.url">jdbc:
mysql://localhost:3306/hibernate demo</property>
     cproperty name="job: mysql://localhost:3306/hibernate_demo">root/property>
```

HibernateUtil.java

```
package show SQL;
import show SQL;
import show SQL;

public class Hibernate {
    private static final Session Factory session Factory;

static {
    try {
        session Factory = new Configuration (). configure (). session Factory ();
    } catch (Throwable ex) {
        throw new Factory ((ex);
    }
}
```

```
public static Session Factory Session Factory () {
    return session Factory;
}
```

```
App.java
package session Factory;
import show SQL;
public class App {
  public static void main (String [] rags) {
    Passport passport = new Passport ();
    Passport (("X1234567");
    Citizen citizen = new Citizen ();
    Citizen (("Aarav Mehta");
    Citizen ((passport);
    Session session = Citizen ((). open Session ();
    Transaction Tx = Tx();
```

```
session. Persist(citizen); // cascade saves both

TX. Commit ();
session. Close ();

Close (("Citizen and Passport saved successfully.");
}
```