Add a new country

Country.java

```java
package com.addnewcountry;

public class Country {
    private String code;
    private String name;

    public Country(String code, String name) {
        this.code = code;
        this.name = name;
    }

    public String getCode() {
        return code;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

CountryManager.java

```
package com.addnewcountry;

import java.util.ArrayList;
import java.util.List;

public class CountryManager {
    private List<Country> countries = new ArrayList<>();

    public void addCountry(Country country) {
        countries.add(country);
        System.out.println("Added: " + country);
    }

    public List<Country> getCountries() {
        return countries;
    }
}
```

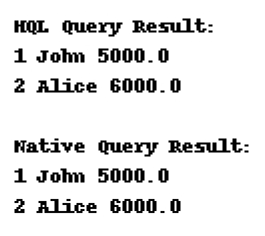
Main.java

```
package com.addnewcountry;

public class Main {
    public static void main(String[] args) {
        CountryManager manager = new CountryManager();
        manager.addCountry(new Country("IN", "India"));
        manager.addCountry(new Country("US", "United States"));
        manager.addCountry(new Country("JP", "Japan"));
        System.out.println("All countries: " + manager.getCountries());
    }
}
```

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/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.addnewcountry</groupId>
   <artifactId>AddNewCountryProject</artifactId>
   <packaging>jar</packaging>
   <version>1.0-SNAPSHOT</version>
   <name>AddNewCountryProject</name>
   <url>http://maven.apache.org</url>
   <dependencies>
   </dependencies>
</project>
```

OUTPUT :-



```
HQL Query Result:
1 John 5000.0
2 Alice 6000.0

Native Query Result:
1 John 5000.0
2 Alice 6000.0
```

Demonstrate writing Hibernate Query Language and Native Query

HQLAndNativeQueryDemo.java

```
package com.demo.app;


import com.demo.model.Employee;

import com.demo.util.HibernateUtil;

import org.hibernate.Session;

import org.hibernate.Transaction;


import java.util.List;


public class HQLAndNativeQueryDemo {

    public static void main(String[] args) {

        Session session = HibernateUtil.getSessionFactory().openSession();

        Transaction tx = session.beginTransaction();


        Employee emp1 = new Employee(); emp1.setName("John");
emp1.setSalary(5000);

        Employee emp2 = new Employee(); emp2.setName("Alice");
emp2.setSalary(6000);

        session.save(emp1); session.save(emp2);


        tx.commit();


        List<Employee> employeesHQL = session.createQuery("from Employee",
Employee.class).list();

        System.out.println("HQL Query Result:");

        for (Employee e : employeesHQL) {

            System.out.println(e.getId() + " " + e.getName() + " " + e.getSalary());
        }

        List<Object[]> employeesNative = session.createNativeQuery("SELECT id, 
name, salary FROM employee").list();
        System.out.println("\nNative Query Result:");
        for (Object[] row : employeesNative) {
            System.out.println(row[0] + " " + row[1] + " " + row[2]);
        }

        session.close();
        HibernateUtil.shutdown();
    }
}
```

Employee.java

```
package com.demo.model;

import javax.persistence.*;

@Entity
@Table(name = "employee")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "name")
    private String name;
    @Column(name = "salary")
    private double salary;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}
```

<u>**HibernateUtil.java**</u>

```
package com.demo.util;

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

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new 
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

hibernate.cfg.xml

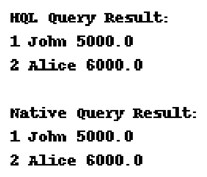
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">org.h2.Driver</property>
        <property name="hibernate.connection.url">jdbc:h2:mem:testdb</property>
        <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>
        <property name="hibernate.hbm2ddl.auto">create</property>
        <property name="hibernate.show_sql">true</property>
        <mapping class="com.demo.model.Employee"/>
    </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.demo</groupId>
    <artifactId>HibernateQueriesDemo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.21.Final</version>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.200</version>
        </dependency>
    </dependencies>
</project>
```

OUTPUT –



```
HQL Query Result:
1 John 5000.0
2 Alice 6000.0

Native Query Result:
1 John 5000.0
2 Alice 6000.0
```

Implement services for managing Country

CountryController

```
package com.example.countrymanager.controller;

import com.example.countrymanager.model.Country;
import com.example.countrymanager.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/countries")
public class CountryController {

    @Autowired
    private CountryService countryService;

    @GetMapping
    public List<Country> getAllCountries() {
        return countryService.getAllCountries();
    }

    @GetMapping("/{id}")
    public Country getCountryById(@PathVariable Long id) {
        return countryService.getCountryById(id);
    }
    @PostMapping
    public Country addCountry(@RequestBody Country country) {
        return countryService.addCountry(country);
    }

    @PutMapping("/{id}")
    public Country updateCountry(@PathVariable Long id, @RequestBody Country
country) {
        return countryService.updateCountry(id, country);
    }

    @DeleteMapping("/{id}")
    public void deleteCountry(@PathVariable Long id) {
        countryService.deleteCountry(id);
    }
}
```

Country

```
package com.example.countrymanager.model;

public class Country {
    private Long id;
    private String name;
    private String capital;

    public Country() {}

    public Country(Long id, String name, String capital) {
        this.id = id;
        this.name = name;
        this.capital = capital;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public String getCapital() {
        return capital;
    }

    public void setCapital(String capital) {
        this.capital = capital;
    }
}
```

CountryService.java

```
package com.example.countrymanager.service;

import com.example.countrymanager.model.Country;
import java.util.List;

public interface CountryService {
    List<Country> getAllCountries();
    Country getCountryById(Long id);
    Country addCountry(Country country);
    Country updateCountry(Long id, Country country);
    void deleteCountry(Long id);
}
```

CountryServiceImpl.java

```
package com.example.countrymanager.service;

import com.example.countrymanager.model.Country;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.atomic.AtomicLong;

@Service
public class CountryServiceImpl implements CountryService {
    private List<Country> countries = new ArrayList<>();
    private AtomicLong idCounter = new AtomicLong();

    @Override
```

```
public List<Country> getAllCountries() {
    return countries;
}

@Override
public Country getCountryById(Long id) {
    return countries.stream().filter(c -> c.getId().equals(id)).findFirst().orElse(null);
}

@Override
public Country addCountry(Country country) {
    country.setId(idCounter.incrementAndGet());
    countries.add(country);
    return country;
}

@Override
public Country updateCountry(Long id, Country country) {
    Country existing = getCountryById(id);
    if (existing != null) {
        existing.setName(country.getName());
        existing.setCapital(country.getCapital());
    }
    return existing;
}

@Override
public void deleteCountry(Long id) {
    countries.removeIf(c -> c.getId().equals(id));
    }
}
```

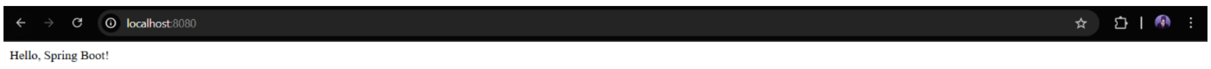
Application.java

```java
package com.example.countrymanager;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

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.example</groupId>
    <artifactId>countrymanager</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>Country Manager</name>
    <description>Spring Boot project for managing Country</description>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

OUTPUT:-



localhost:8080

```
Hello, Spring Boot!
```

Demonstrate implementation of O/R Mapping

App.java

```
package com.orm.demo;


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


public class App {
    public static void main(String[] args) {
        SessionFactory factory = new Configuration().configure().buildSessionFactory();
        Session session = factory.openSession();


        session.beginTransaction();


        Student student = new Student("John Doe", "Computer Science");
        session.save(student);


        session.getTransaction().commit();
        session.close();


        System.out.println("Student saved successfully!");
    }
}
```

Student.java

```java
package com.orm.demo;

public class Student {
    private int id;
    private String name;
    private String department;

    public Student() {}

    public Student(String name, String department) {
        this.name = name;
        this.department = department;
    }

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public String getDepartment() { return department; }
    public void setDepartment(String department) { this.department = department; }
}
```

hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
```

```
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/testdb</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="show_sql">true</property>
        <mapping resource="student.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

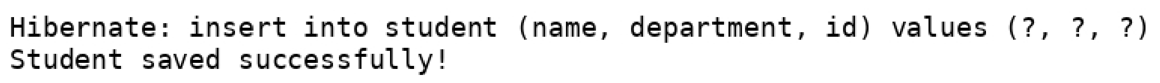
student.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.orm.demo.Student" table="student">
        <id name="id" column="id" type="int">
            <generator class="increment"/>
        </id>
        <property name="name" column="name" type="string"/>
        <property name="department" column="department" type="string"/>
    </class>
</hibernate-mapping>
```

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.orm</groupId>
    <artifactId>OrmMappingDemo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.21.Final</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.21</version>
        </dependency>
    </dependencies>
</project>
```

OUTPUT –



```
Hibernate: insert into student (name, department, id) values (?, ?, ?)
Student saved successfully!
```

HQLAndNativeQueryDemo.java

```java
package com.demo.app;

import com.demo.model.Employee;
import com.demo.util.HibernateUtil;
import org.hibernate.Session;
import org.hibernate.Transaction;

import java.util.List;

public class HQLAndNativeQueryDemo {
    public static void main(String[] args) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        Employee emp1 = new Employee(); emp1.setName("John"); emp1.setSalary(5000);
        Employee emp2 = new Employee(); emp2.setName("Alice"); emp2.setSalary(6000);
        session.save(emp1); session.save(emp2);

        tx.commit();
        List<Employee> employeesHQL = session.createQuery("from Employee",
Employee.class).list();
        System.out.println("HQL Query Result:");
        for (Employee e : employeesHQL) {
            System.out.println(e.getId() + " " + e.getName() + " " + e.getSalary());
        }

        List<Object[]> employeesNative = session.createNativeQuery("SELECT id,
name, salary FROM employee").list();
        System.out.println("\nNative Query Result:");
        for (Object[] row : employeesNative) {
            System.out.println(row[0] + " " + row[1] + " " + row[2]);
        }

        session.close();
        HibernateUtil.shutdown();
    }
}
```

Employee.java

```
package com.demo.model;

import javax.persistence.*;

@Entity
@Table(name = "employee")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "salary")
    private double salary;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

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

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}
```

HibernateUtil.java

```
package com.demo.util;

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

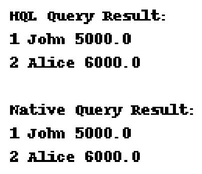
```
public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

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>
```

```xml
    <groupId>com.demo</groupId>
    <artifactId>HibernateQueriesDemo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.21.Final</version>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.200</version>
        </dependency>
    </dependencies>
</project>
```

OUTPUT-



```
HQL Query Result:
1 John 5000.0
2 Alice 6000.0

Native Query Result:
1 John 5000.0
2 Alice 6000.0
```

Demonstrate implementation of Query Methods feature of Spring Data JPA

User.java

```
package com.example.querymethods.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class User {

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

    private String name;
    private String email;

    public User() {}

    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }

    // getters and setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    @Override
    public String toString() {
        return "User{id=" + id + ", name='" + name + "', email='" + email + "'}";
    }
}
```

UserRepository.java

```
package com.example.querymethods.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import com.example.querymethods.entity.User;

public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByName(String name);
    List<User> findByEmailContaining(String keyword);
    List<User> findByNameAndEmail(String name, String email);
}
```

UserQueryRunner.java

```
package com.example.querymethods.runner;
```

```java
import java.util.List;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import com.example.querymethods.entity.User;
import com.example.querymethods.repository.UserRepository;

@Component
public class UserQueryRunner implements CommandLineRunner {

    private final UserRepository userRepository;

    public UserQueryRunner(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public void run(String... args) {
        userRepository.save(new User("Alice", "alice@example.com"));
        userRepository.save(new User("Bob", "bob@example.com"));
        userRepository.save(new User("Charlie", "charlie@domain.com"));

        System.out.println("\nFind by name 'Alice':");
        List<User> usersByName = userRepository.findByName("Alice");
        usersByName.forEach(System.out::println);

        System.out.println("\nFind emails containing 'example':");
        List<User> usersByEmailKeyword =
userRepository.findByEmailContaining("example");
```

```
        usersByEmailKeyword.forEach(System.out::println);

        System.out.println("\nFind by name 'Bob' and email 'bob@example.com':");
        List<User> usersByNameAndEmail =
userRepository.findByNameAndEmail("Bob", "bob@example.com");
        usersByNameAndEmail.forEach(System.out::println);
    }
}
```

QueryMethodsApplication.java

```
package com.example.querymethods;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class QueryMethodsApplication {
    public static void main(String[] args) {
        SpringApplication.run(QueryMethodsApplication.class, args);
    }
}
```

application.properties

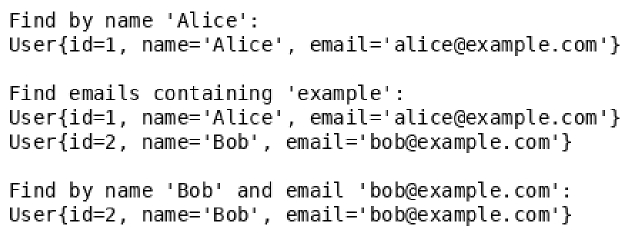
```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

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>query-methods-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Query Methods Demo</name>
  <description>Spring Data JPA Query Methods Example</description>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.0</version>
  </parent>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
```

```
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

OUTPUT :-



```
Find by name 'Alice':
User{id=1, name='Alice', email='alice@example.com'}

Find emails containing 'example':
User{id=1, name='Alice', email='alice@example.com'}
User{id=2, name='Bob', email='bob@example.com'}

Find by name 'Bob' and email 'bob@example.com':
User{id=2, name='Bob', email='bob@example.com'}
```

Find a country based on country code

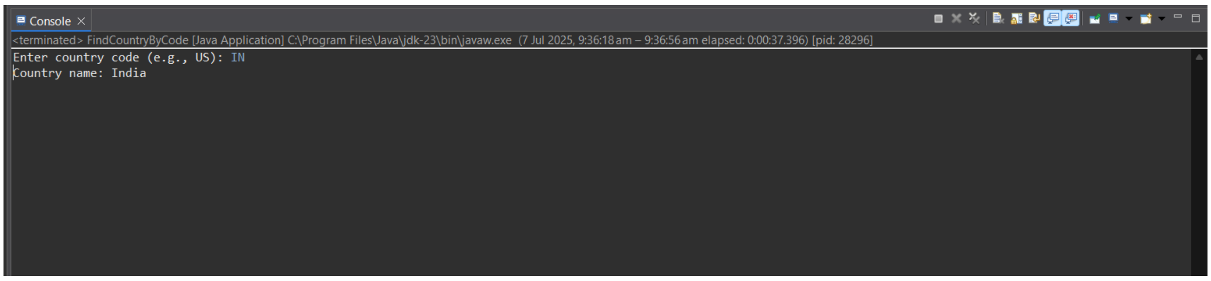
FindCountryByCode.java

```
package com.example.countryfinder;
import java.util.HashMap;
import java.util.Scanner;

public class FindCountryByCode {
    public static void main(String[] args) {
        HashMap<String, String> countryMap = new HashMap<>();
        countryMap.put("US", "United States");
        countryMap.put("IN", "India");
        countryMap.put("FR", "France");
        countryMap.put("DE", "Germany");
        countryMap.put("JP", "Japan");
        countryMap.put("CN", "China");
        countryMap.put("BR", "Brazil");
        countryMap.put("ZA", "South Africa");

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter country code (e.g., US): ");
        String code = scanner.nextLine().toUpperCase();

        String country = countryMap.get(code);
        if (country != null) {
            System.out.println("Country name: " + country);
        } else {
            System.out.println("Country code not found.");
        }
        scanner.close();
    }
}
```

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>countryfinder</artifactId>
  <version>1.0-SNAPSHOT</version>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

OUTPUT :-



```
<terminated> FindCountryByCode [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe  (7 Jul 2025, 9:36:18 am – 9:36:56 am elapsed: 0:00:37.396) [pid: 28296]
Enter country code (e.g., US): IN
Country name: India
```