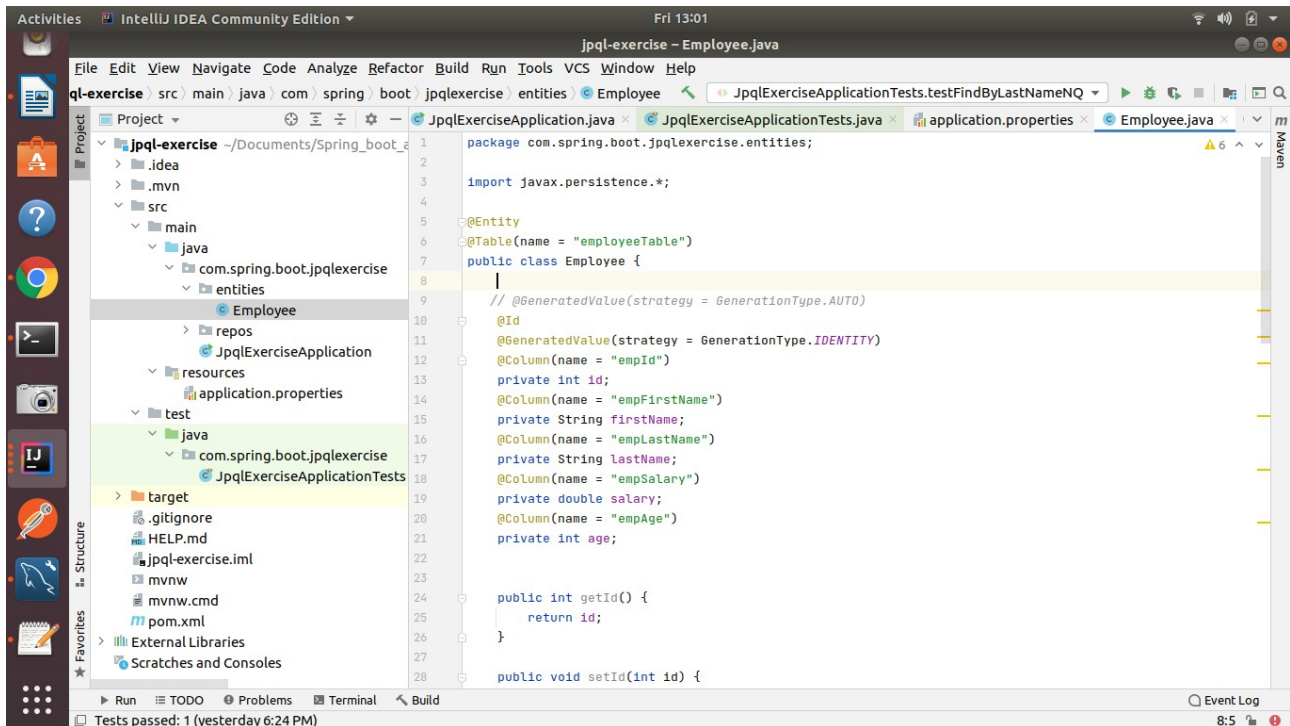


Spring Data JPA with Hibernate Part 2 Exercise

JPQL:

1. Display the first name, last name of all employees having salary greater than average salary ordered in ascending by their age and in descending by their salary.



```
package com.spring.boot.jpqlexercise.repos;
```

```
import com.spring.boot.jpqlexercise.entities.Employee;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;
```

```
import javax.transaction.Transactional;
import java.util.List;
```

```
public interface EmployeeRepository extends CrudRepository<Employee, Integer> {
```

```
    //Q1 jpql
    @Query("select firstName, lastName from Employee where " +
        "salary > (select AVG(salary) from Employee)" +
        " ORDER BY age ASC, salary DESC")
    List<Object[]> findAllEmployee();
```


3. Delete all employees with minimum salary.

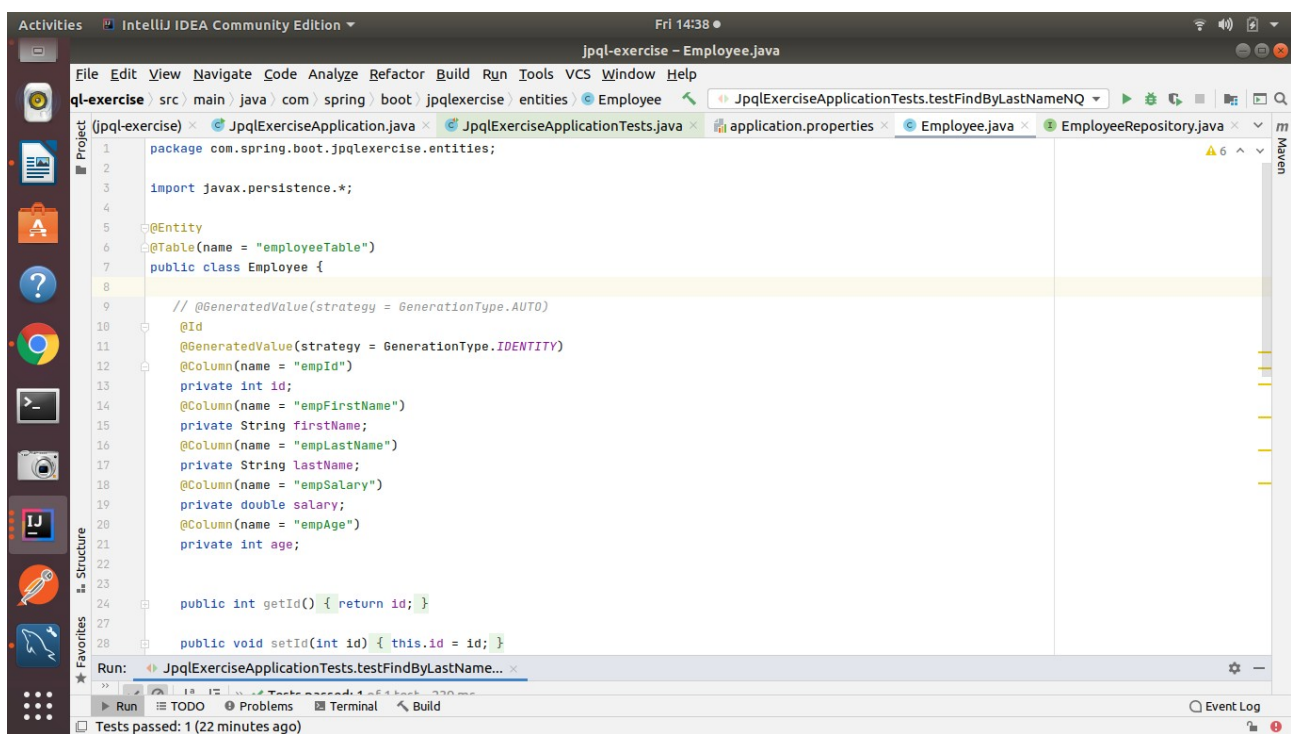
```
//delete min salary
@Query("select min(salary) from Employee")
Integer minSalary();

}
@Modifying
@Query(value = "Delete FROM Employee WHERE salary=:minsalary")
void deleteMinSalary(@Param("minsalary")Integer minsalary);

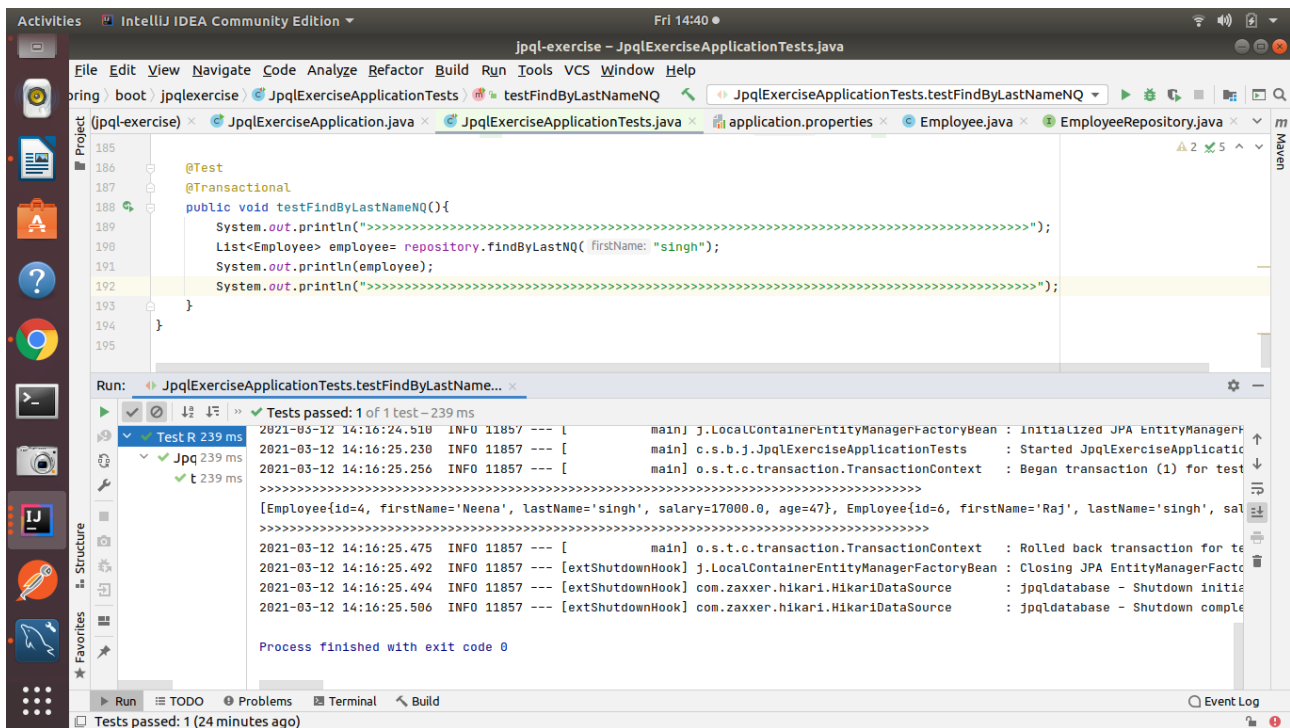
@Transactional
public void deleteMinSalary()
{
    Integer minSal = repository.minSalary();
    System.out.println(">>>>>>>>>" + minSal);
    repository.deleteMinSalary(minSal);
}
```

Native SQL Query:

1. Display the id, first name, age of all employees where last name ends with "singh"



```
//Q1 NQ
@Query(value = "select * from employee_table where emp_last_name=:lastName", nativeQuery = true)
List<Employee> findByLastNQ(@Param("lastName") String firstName);
```



2. Delete all employees with age greater than 45(Should be passed as a parameter)

```

//Q2 NQ
3 @Modifying
3 @Query(value = "delete from employee_table where emp_age>:age", nativeQuery = true)
void deleteEmployeeHavingAgeGreaterThan(@Param("age") int age);

```

```

//Q2 NQ
1 @Test
1 @Transactional
1 @Rollback(value = false)
1 public void testDeleteEmployeeHavingAgeGreaterThan(){
1
1     repository.deleteEmployeeHavingAgeGreaterThan(45);
1 }

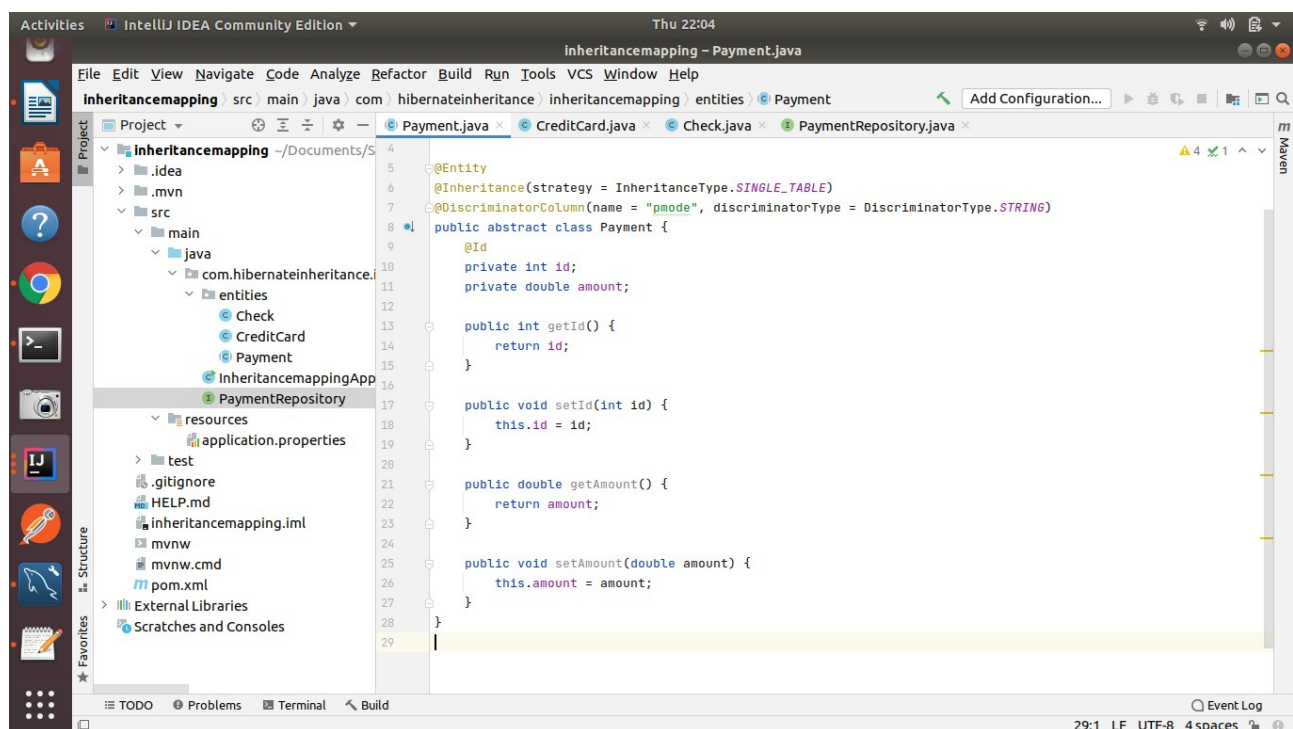
```

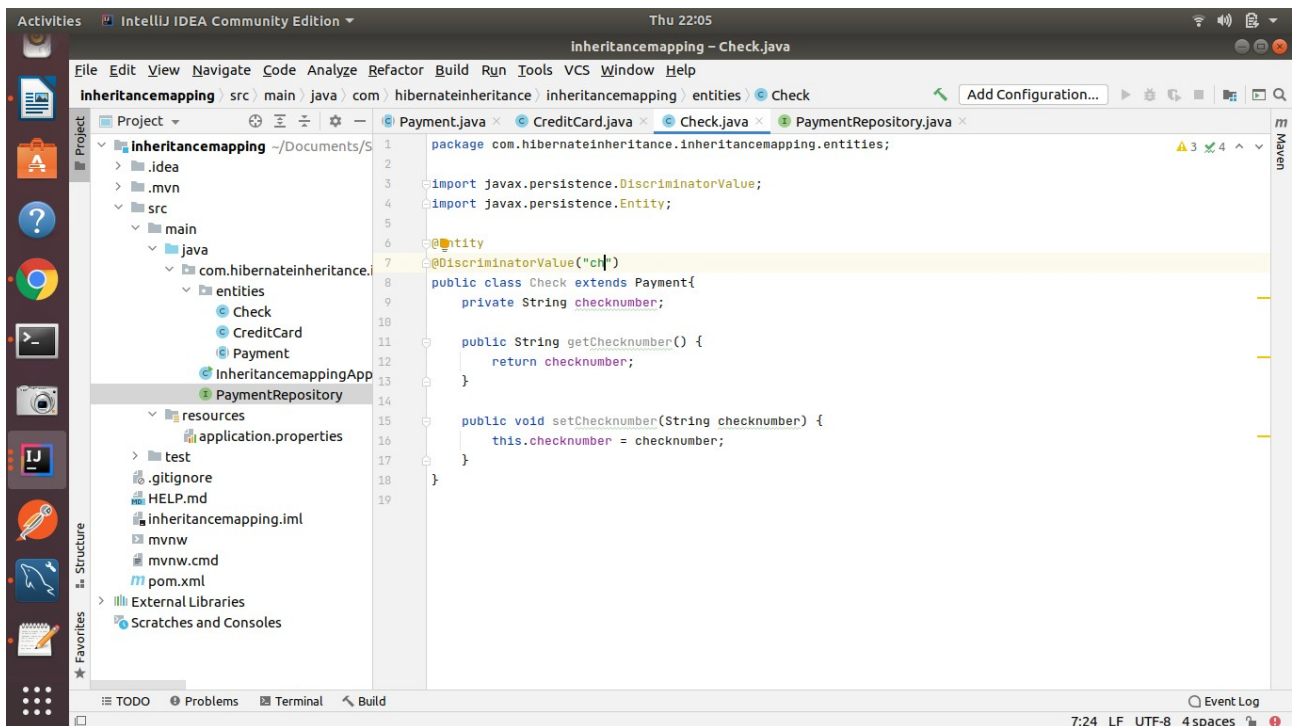
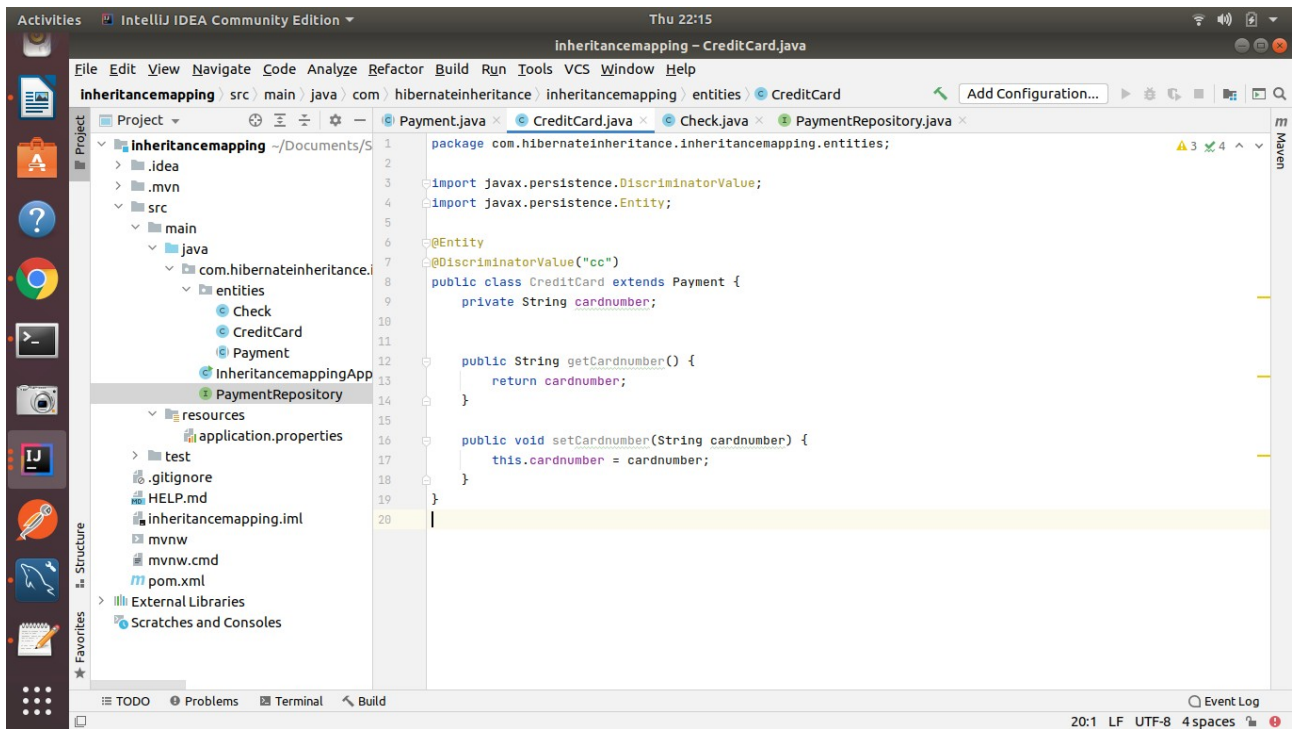

| # | emp_id | emp_first_name | emp_last_name | emp_salary | emp_age | |
|---|--------|----------------|---------------|------------|---------|--|
| 1 | 1 | Naveen | Garg | 20000.00 | 25 | |
| 2 | 2 | Bill | Gates | 50000.00 | 35 | |
| 3 | 5 | Lex | De Hann | 17000.00 | 23 | |
| 4 | 6 | Raj | singh | 30000.00 | 28 | |
| 5 | 7 | Jitender | Kumar | 70000.00 | 36 | |
| 6 | 8 | Ankit | Saini | 45000.00 | 31 | |
| * | NULL | NULL | NULL | NULL | NULL | |

Inheritance Mapping:

1. Implement and demonstrate Single Table strategy.

```
create table payment(
id int PRIMARY KEY,
pmode varchar(2),
amount decimal(8,3) ,
cardnumber varchar(20),
checknumber varchar(20)
);
```





```

@Test
public void createPayment(){
    CreditCard cc = new CreditCard();
    cc.setId(213);
    cc.setAmount(5000);
    cc.setCardnumber("9876543210");
    repository.save(cc);
}

```

```

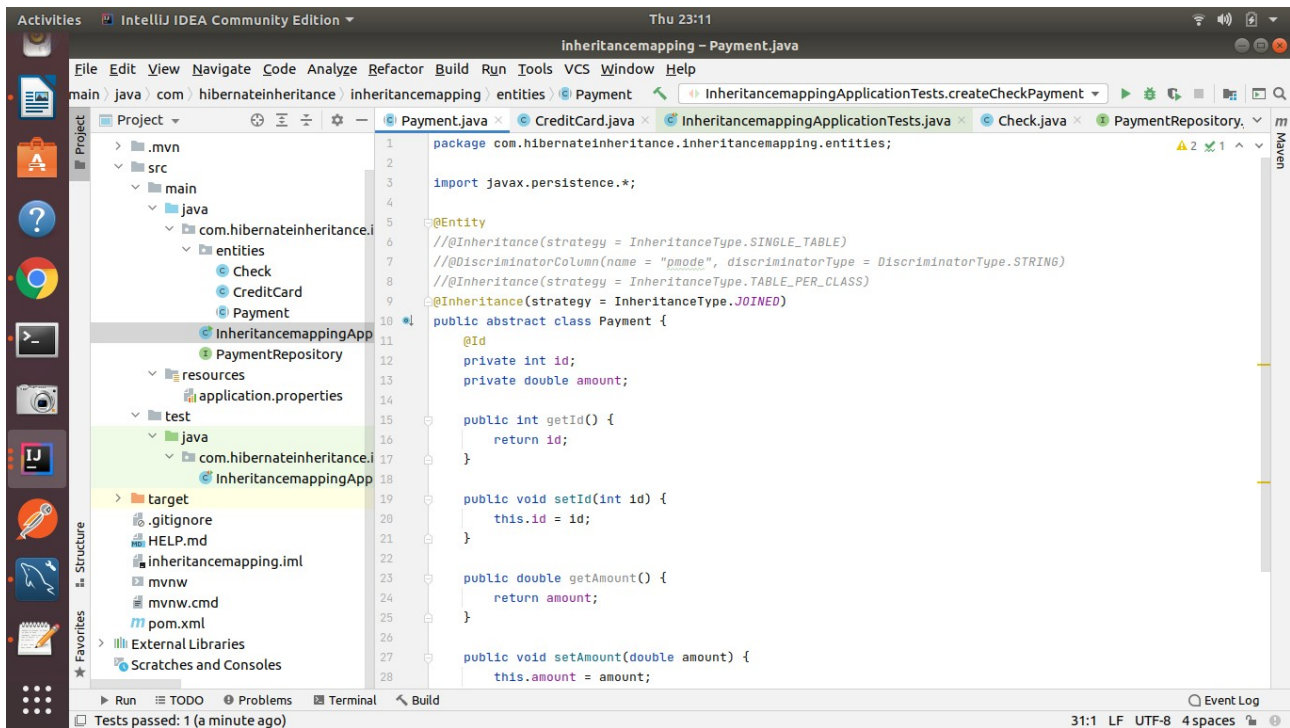
@Test
public void createCheckPayment(){
    Check ch = new Check();
    ch.setId(214);
    ch.setAmount(6000);
    ch.setChecknumber("9876543210");
    repository.save(ch);
}

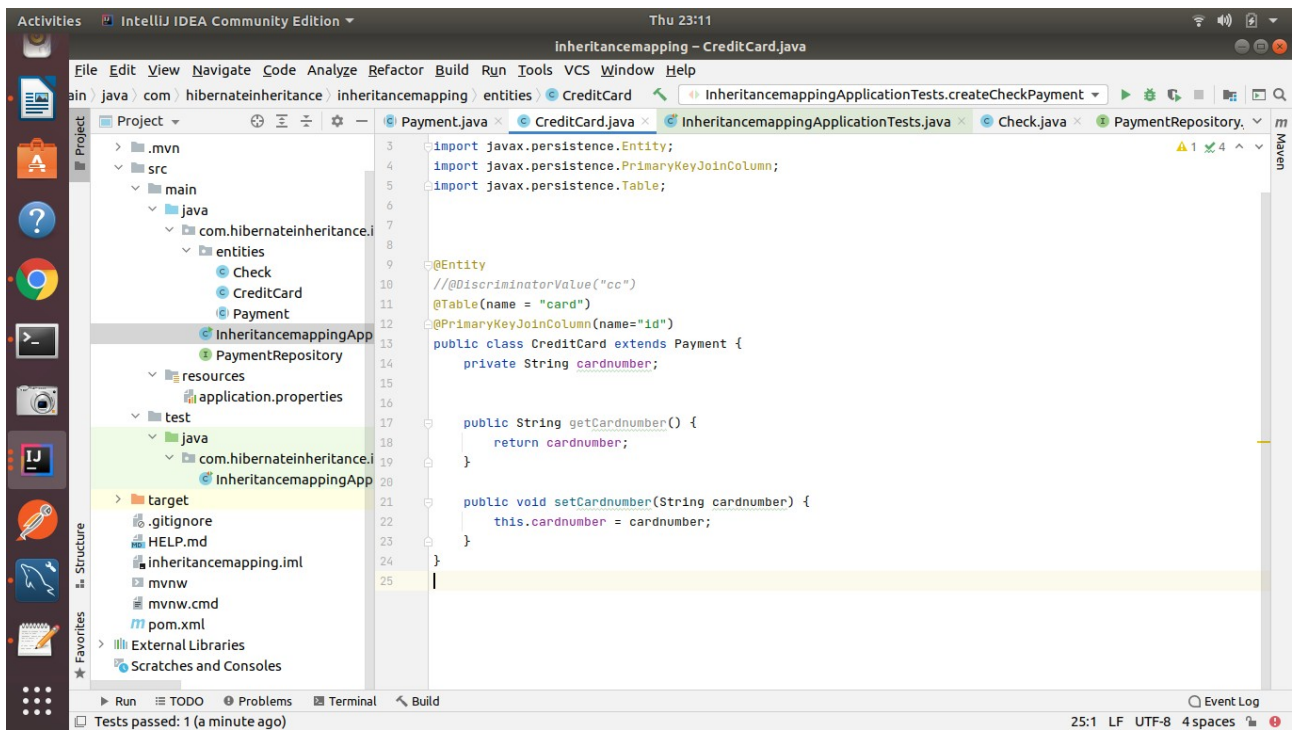
```

| # | id | pmode | amount | cardnumber | checknumber |
|---|------|-------|----------|------------|-------------|
| 1 | 213 | cc | 5000.000 | 9876543210 | NULL |
| 2 | 214 | ch | 6000.000 | NULL | 9876543210 |
| * | NULL | NULL | NULL | NULL | NULL |

2. Implement and demonstrate Joined strategy.

```
create table payment(  
id int PRIMARY KEY NOT NULL AUTO_INCREMENT,  
amount decimal(8,3)  
);  
  
create table card(  
id int ,  
cardnumber varchar(20),  
FOREIGN KEY (id)  
REFERENCES payment(id)  
)  
  
create table bankcheck(  
id int ,  
checknumber varchar(20),  
FOREIGN KEY (id)  
REFERENCES payment(id)  
)
```





```
@Test
public void createPayment(){
    CreditCard cc = new CreditCard();
    cc.setId(213);
    cc.setAmount(5000);
    cc.setCardnumber("9876543210");
    repository.save(cc);
}
```

```
@Test
public void createCheckPayment(){
    Check ch = new Check();
    ch.setId(214);
    ch.setAmount(6000);
    ch.setChecknumber("9876543210");
    repository.save(ch);
}
```

| # | id | amount | |
|---|------|----------|--|
| 1 | 213 | 5000.000 | |
| 2 | 214 | 6000.000 | |
| * | NULL | NULL | |

| # | id | checknumber | |
|---|-----|-------------|--|
| 1 | 214 | 9876543210 | |

| # | id | cardnumber | |
|---|-----|------------|--|
| 1 | 213 | 9876543210 | |

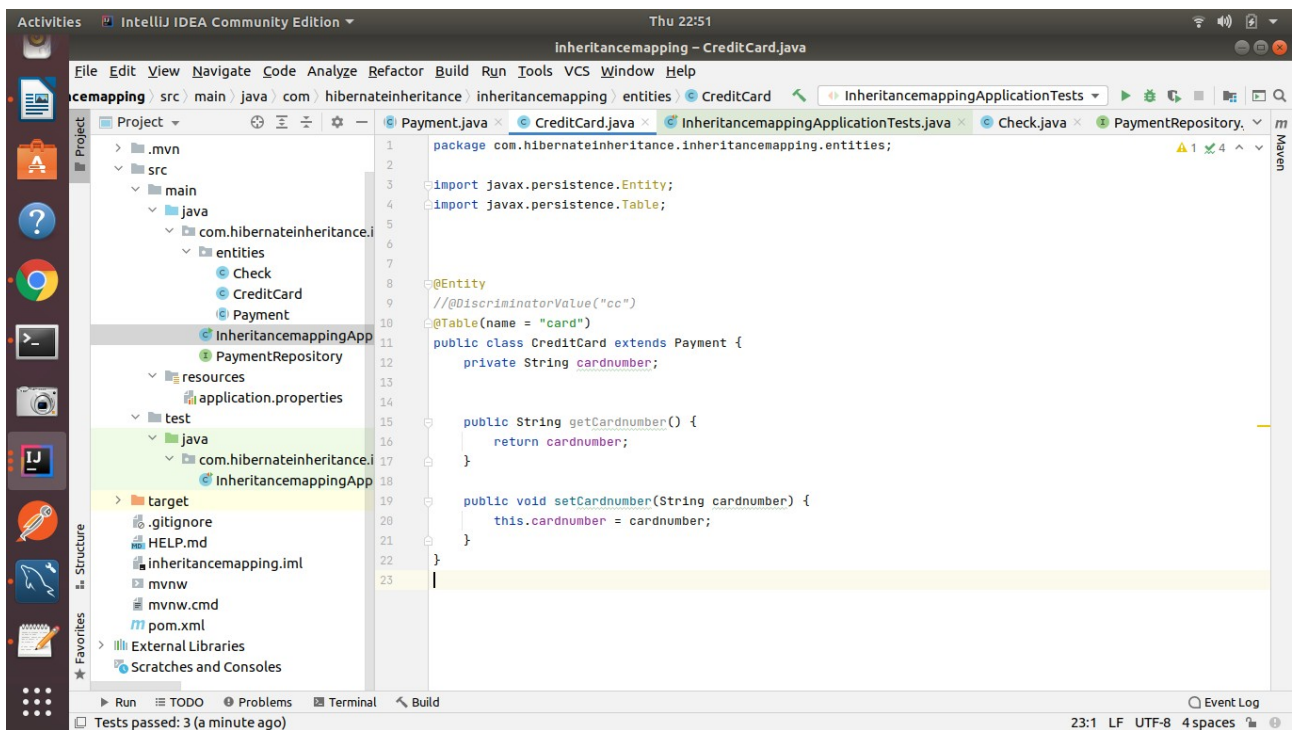
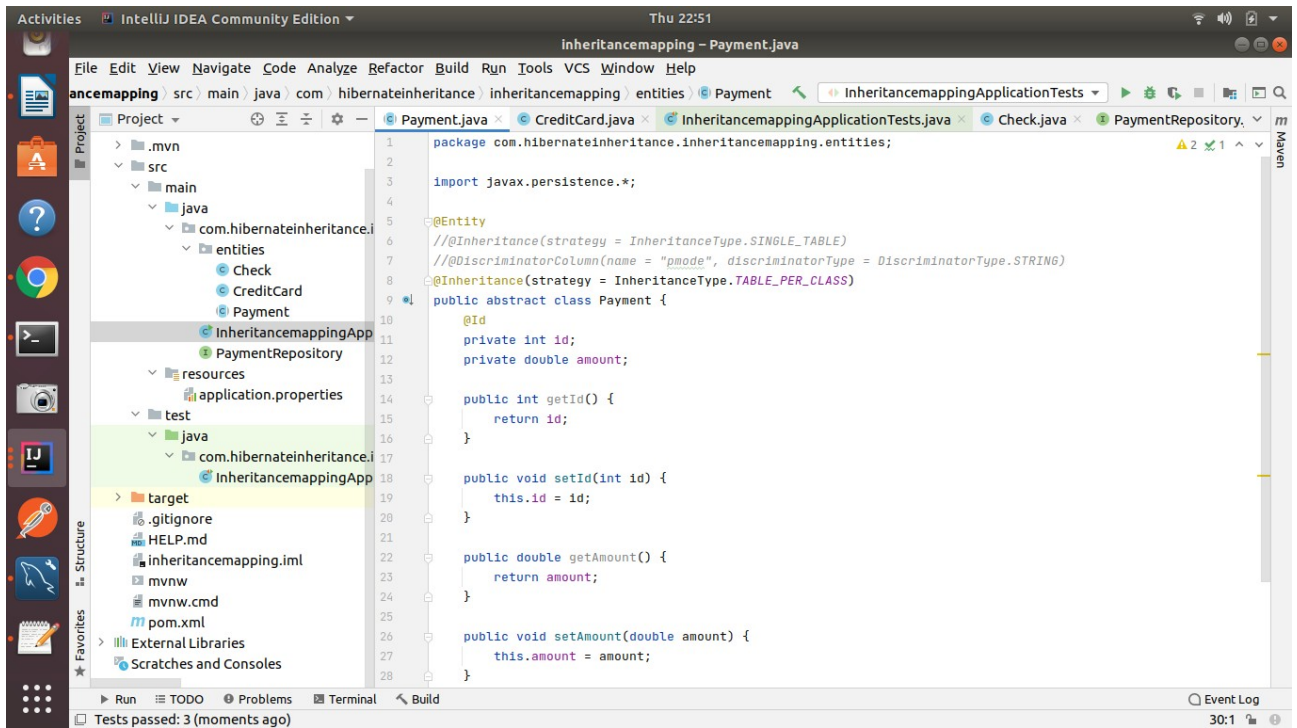
3. Implement and demonstrate Table Per Class strategy.

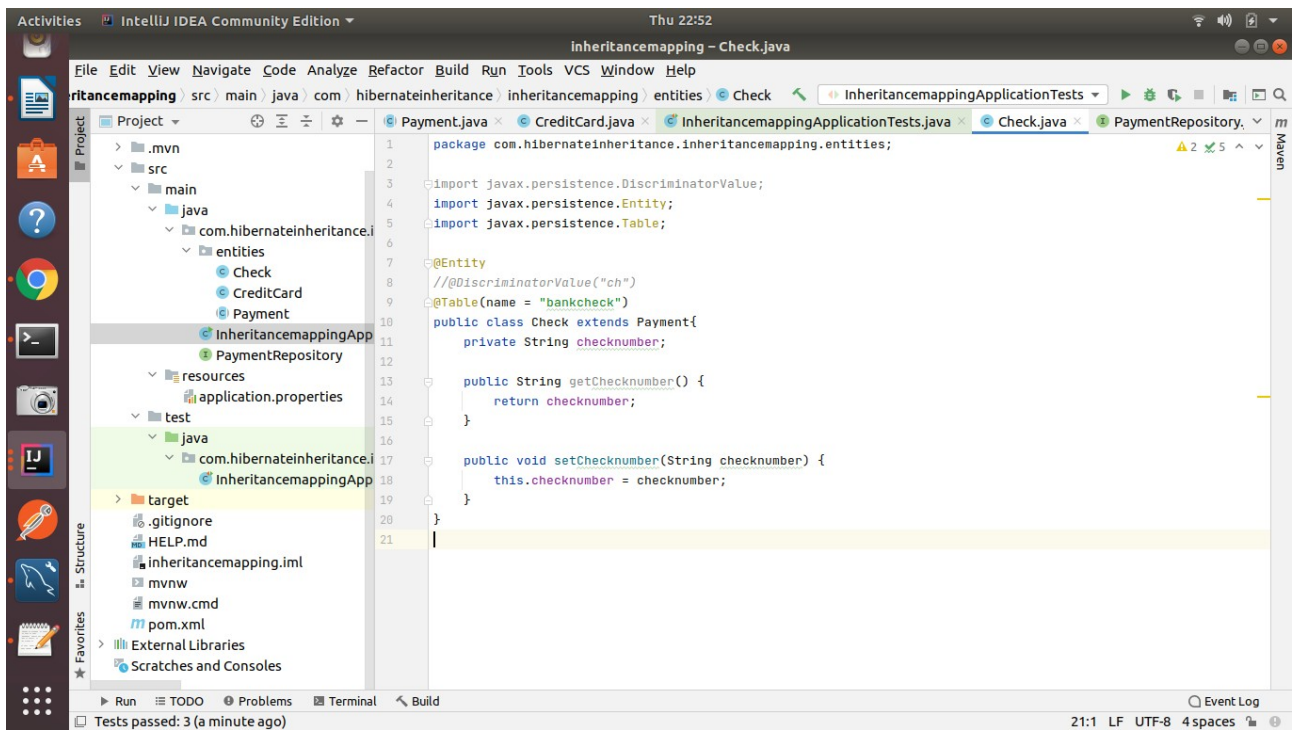
```

create table card(
id int PRIMARY KEY,
amount decimal(8,3),
cardnumber varchar(20)
)

create table bankcheck(
id int PRIMARY KEY,
amount decimal(8,3),
checknumber varchar(20)
)

```





```

@Test
public void createPayment(){
    CreditCard cc = new CreditCard();
    cc.setId(213);
    cc.setAmount(5000);
    cc.setCardnumber("9876543210");
    repository.save(cc);
}

@Test
public void createCheckPayment(){
    Check ch = new Check();
    ch.setId(214);
    ch.setAmount(6000);
    ch.setChecknumber("9876543210");
    repository.save(ch);
}
}

```

| # | id | amount | checknumber | |
|---|------|----------|-------------|--|
| 1 | 214 | 6000.000 | 9876543210 | |
| * | NULL | NULL | NULL | |

| # | id | amount | cardnumber | |
|---|------|----------|------------|--|
| 1 | 213 | 5000.000 | 9876543210 | |
| * | NULL | NULL | NULL | |

Component Mapping:

1. Implement and demonstrate Embedded mapping using employee table having following fields: id, firstName, lastName, age, basicSalary, bonusSalary, taxAmount, specialAllowanceSalary.

```
create table employee(
  id int,
  firstname varchar(20),
  lastname varchar(20),
  age int,
  basicsalary decimal(8,3),
  bonussalary decimal(8,3),
  taxamount decimal(8,3),
  specialallowancesalary decimal(8,3)
)

select * from employee
```

The screenshot displays the IntelliJ IDEA Community Edition interface. The main editor shows the `Employee.java` file in the `com.spring.componentmapping.cmapping.entities` package. The code defines an `Employee` class with the following attributes and methods:

```
package com.spring.componentmapping.cmapping.entities;

import javax.persistence.*;

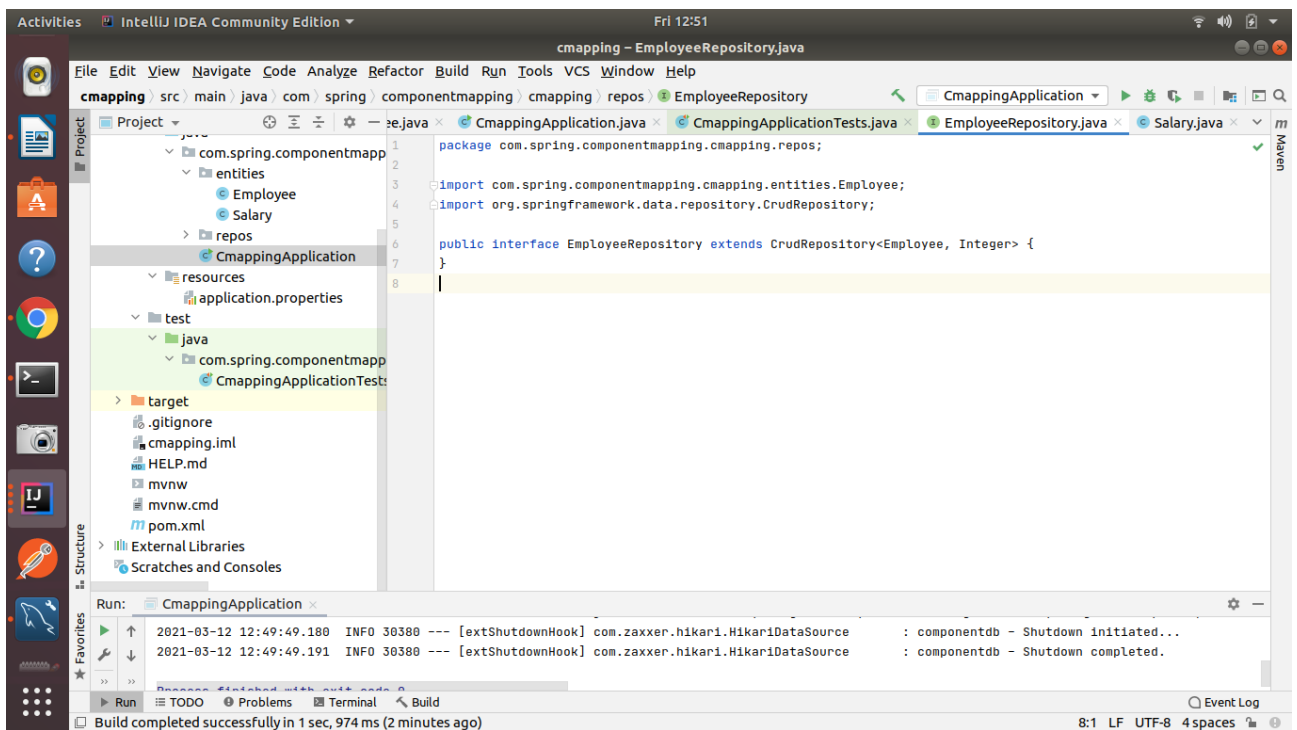
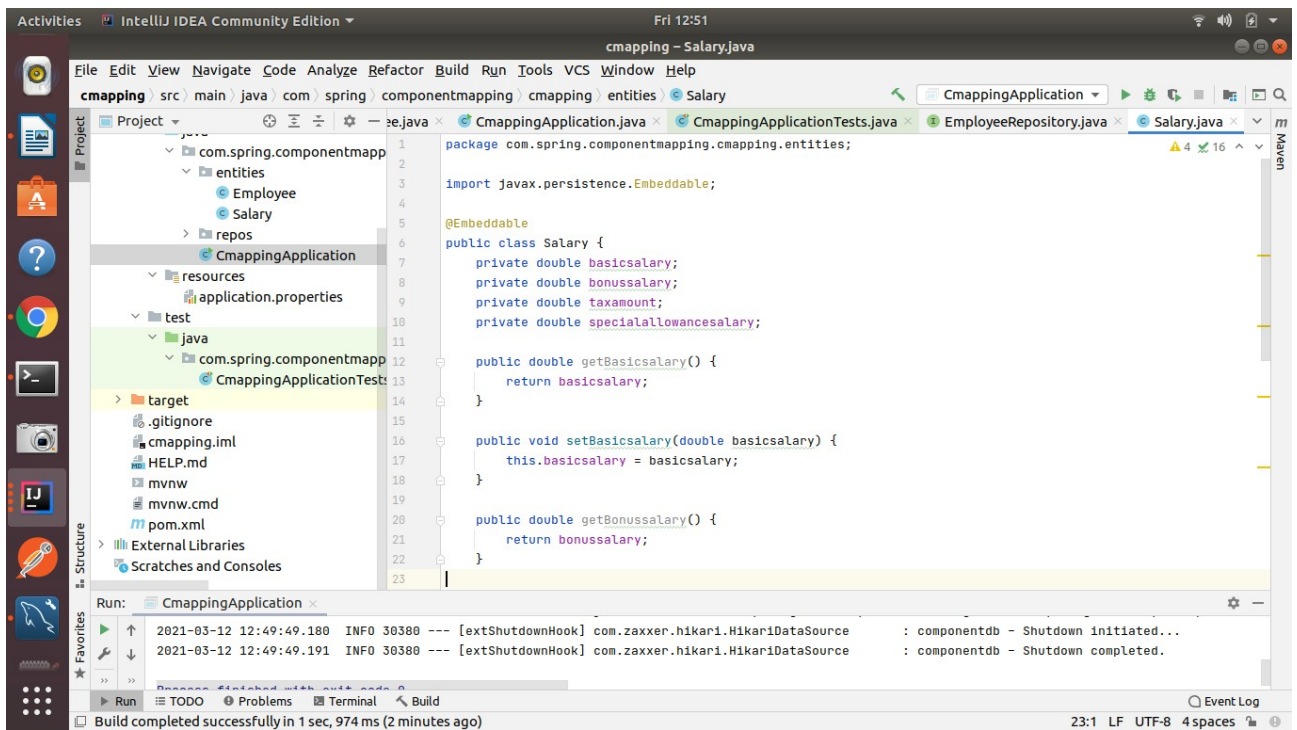
@Entity
@Table(name = "employee")
public class Employee {

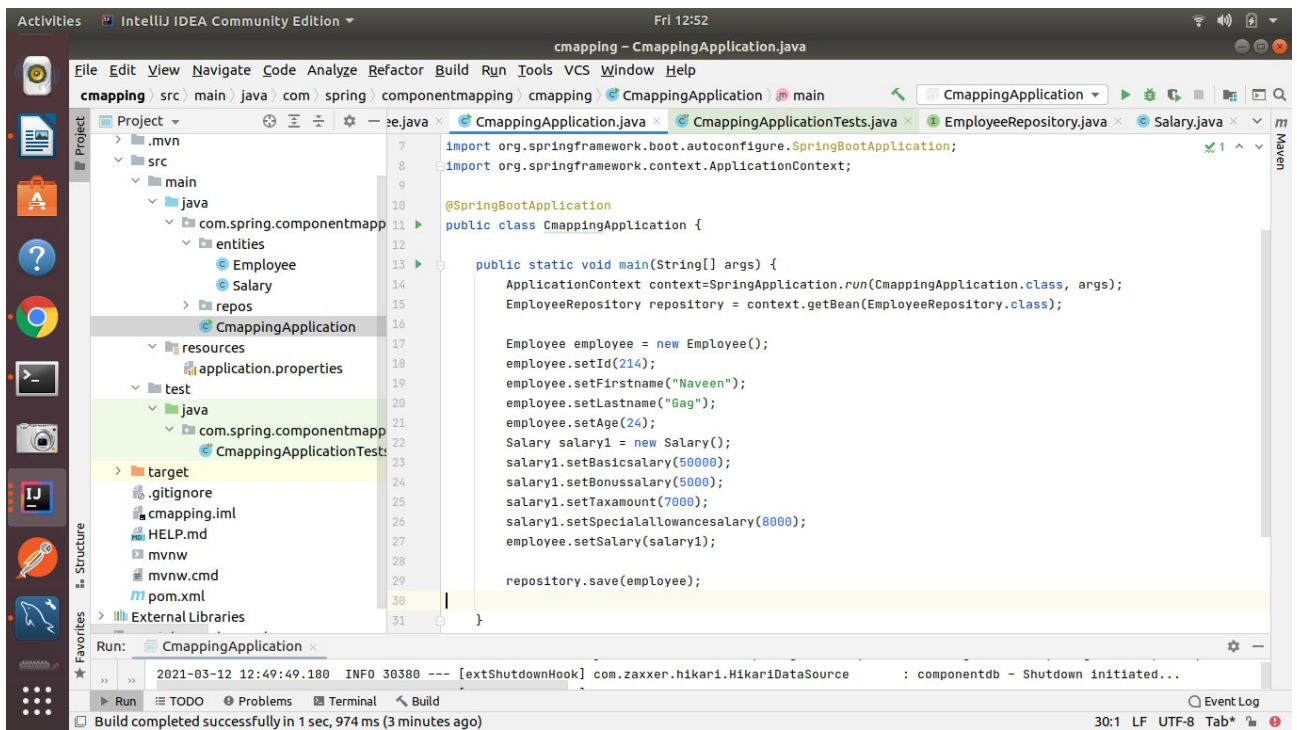
    @Id
    @GeneratedValue
    private int id;
    private String firstname;
    private String lastname;
    private int age;

    @Embedded
    private Salary salary;

    public int getId() {
        return id;
    }
}
```

The left sidebar shows the project structure, including the `cmapping` module and its sub-packages. The bottom status bar indicates that the build completed successfully in 1 second, 974 ms.





| Result Grid | | | | | | | | | |
|----------------------------|----|-----------|----------|-----|-------------|-------------|-----------|------------------------|--|
| Filter Rows: | | | | | | | | | |
| Export: Wrap Cell Content: | | | | | | | | | |
| # | id | firstname | lastname | age | basicsalary | bonussalary | taxamount | specialallowancesalary | |
| 1 | 5 | Naveen | Gag | 24 | 50000.000 | 5000.000 | 7000.000 | 8000.000 | |