

Demonstrate implementation of Query Methods feature of Spring Data JPA

```
package com.example.querymethods_demo.entity;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
public class Product {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private String name;
```

```
    private Double price;
```

```
    private String category;
```

```
    public Product() {}
```

```
    public Product(String name, Double price, String category) {
```

```
        this.name = name;
```

```
        this.price = price;
```

```
        this.category = category;
```

```
    }
```

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

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

```
public Double getPrice() { return price; }  
public String getCategory() { return category; }
```

```
public void setId(Long id) { this.id = id; }  
public void setName(String name) { this.name = name; }  
public void setPrice(Double price) { this.price = price; }  
public void setCategory(String category) { this.category = category; }
```

```
@Override
```

```
public String toString() {  
    return "Product{" +  
        "id=" + id +  
        ", name=" + name + "\" +  
        ", price=" + price +  
        ", category=" + category + "\" +  
        '}'";  
}
```

```
}
```

```
package com.example.querymethods_demo.repository;
```

```
import com.example.querymethods_demo.entity.Product;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import java.util.List;
```

```
public interface ProductRepository extends JpaRepository<Product, Long> {  
    List<Product> findByCategory(String category);  
    List<Product> findByPriceLessThan(Double price);  
}
```

```

        Product findByName(String name);

        List<Product> findByNameContaining(String keyword);
    }

package com.example.querymethods_demo;

import com.example.querymethods_demo.entity.Product;
import com.example.querymethods_demo.repository.ProductRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class DataLoader implements CommandLineRunner {

    private final ProductRepository repo;

    public DataLoader(ProductRepository repo) {
        this.repo = repo;
    }

    @Override
    public void run(String... args) {
        repo.save(new Product("Laptop", 80000.0, "Electronics"));
        repo.save(new Product("Keyboard", 1500.0, "Electronics"));
        repo.save(new Product("Notebook", 50.0, "Stationery"));
        repo.save(new Product("Pen", 10.0, "Stationery"));
        repo.save(new Product("Refrigerator", 40000.0, "Appliances"));

        System.out.println("Products in Electronics:");
    }
}

```

```

repo.findByCategory("Electronics").forEach(System.out::println);

System.out.println("\nProducts cheaper than 1000:");

repo.findByPriceLessThan(1000.0).forEach(System.out::println);

System.out.println("\nProduct named 'Laptop':");

System.out.println(repo.findByName("Laptop"));

System.out.println("\nProducts with name containing 'note':");

repo.findByNameContaining("note").forEach(System.out::println);

}

}

```

OUTPUT:

The screenshot shows the Eclipse IDE with a Java project named 'querymethods\_demo'. The left sidebar displays the Project Explorer with a tree view of the project structure, including 'src/main/java' and 'src/main/resources'. The main editor window shows the 'Product.java' file with the following code:

```

28 public String getCategory() { return category; }
29
30 public void setId(Long id) { this.id = id; }
31 public void setName(String name) { this.name = name; }
32 public void setPrice(Double price) { this.price = price; }
33 public void setCategory(String category) { this.category = category; }
34
35 @Override

```

The bottom console window shows the output of the application, which includes SQL queries and the results of the repository methods:

```

Hibernate: select p1_0.id,p1_0.category,p1_0.name,p1_0.price from product p1_0 where p1_0.category='Electronics'
Product{id=1, name='Laptop', price=80000.0, category='Electronics'}
Product{id=2, name='Keyboard', price=1500.0, category='Electronics'}

Products cheaper than 1000:
Hibernate: select p1_0.id,p1_0.category,p1_0.name,p1_0.price from product p1_0 where p1_0.price<1000.0
Product{id=3, name='Notebook', price=50.0, category='Stationery'}
Product{id=4, name='Pen', price=10.0, category='Stationery'}

Product named 'Laptop':
Hibernate: select p1_0.id,p1_0.category,p1_0.name,p1_0.price from product p1_0 where p1_0.name='Laptop'
Product{id=1, name='Laptop', price=80000.0, category='Electronics'}

Products with name containing 'note':
Hibernate: select p1_0.id,p1_0.category,p1_0.name,p1_0.price from product p1_0 where p1_0.name like '%note%'
Product{id=3, name='Notebook', price=50.0, category='Stationery'}

```