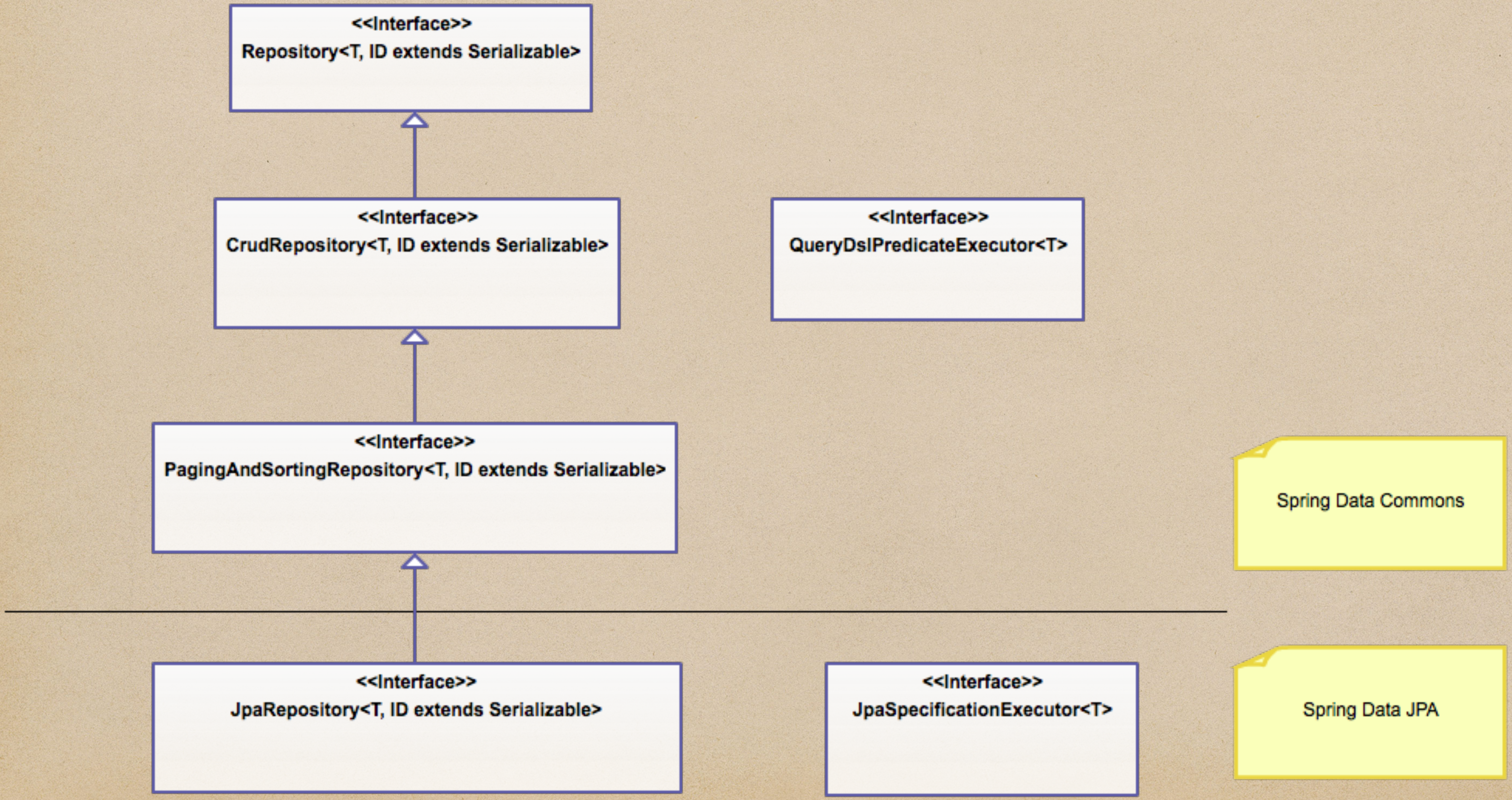
How to use Spring Data JPA Repository

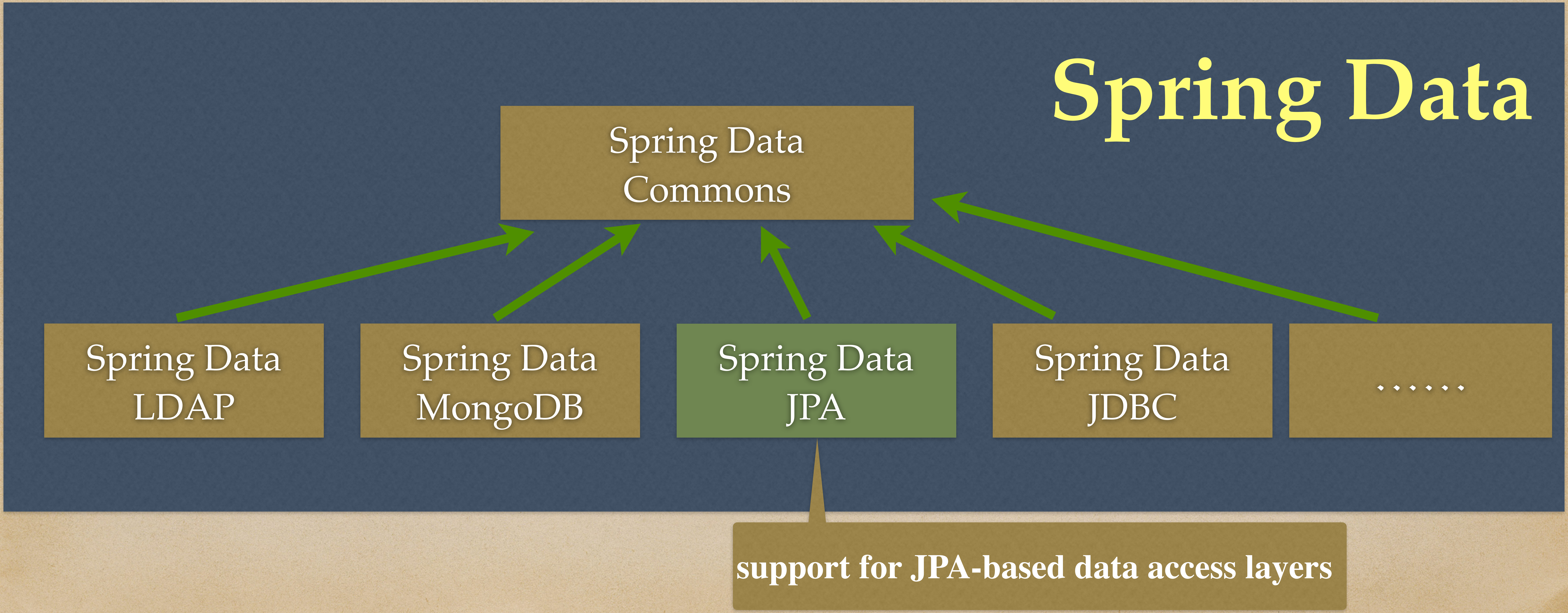
By Ramesh Fadatare (Java Guides)

Spring Data Commons and Spring Data JPA Repository Interfaces



<<Interface>>
Repository<T, ID extends Serializable>
<<Interface>>
CrudRepository<T, ID extends Serializable>
<<Interface>>
QueryDslPredicateExecutor<T>
<<Interface>>
PagingAndSortingRepository<T, ID extends Serializable>
Spring Data Commons
<<Interface>>
JpaRepository<T, ID extends Serializable>
<<Interface>>
JpaSpecificationExecutor<T>
Spring Data JPA

Spring Data


Spring Data
Spring Data Commons
Spring Data LDAP
Spring Data MongoDB
Spring Data JPA
Spring Data JDBC
……
support for JPA-based data access layers

JpaRepository

```java
public interface ProductRepository extends JpaRepository<Product,Integer> {

}
```

Entity Type

Primary Key

CRUD Operations

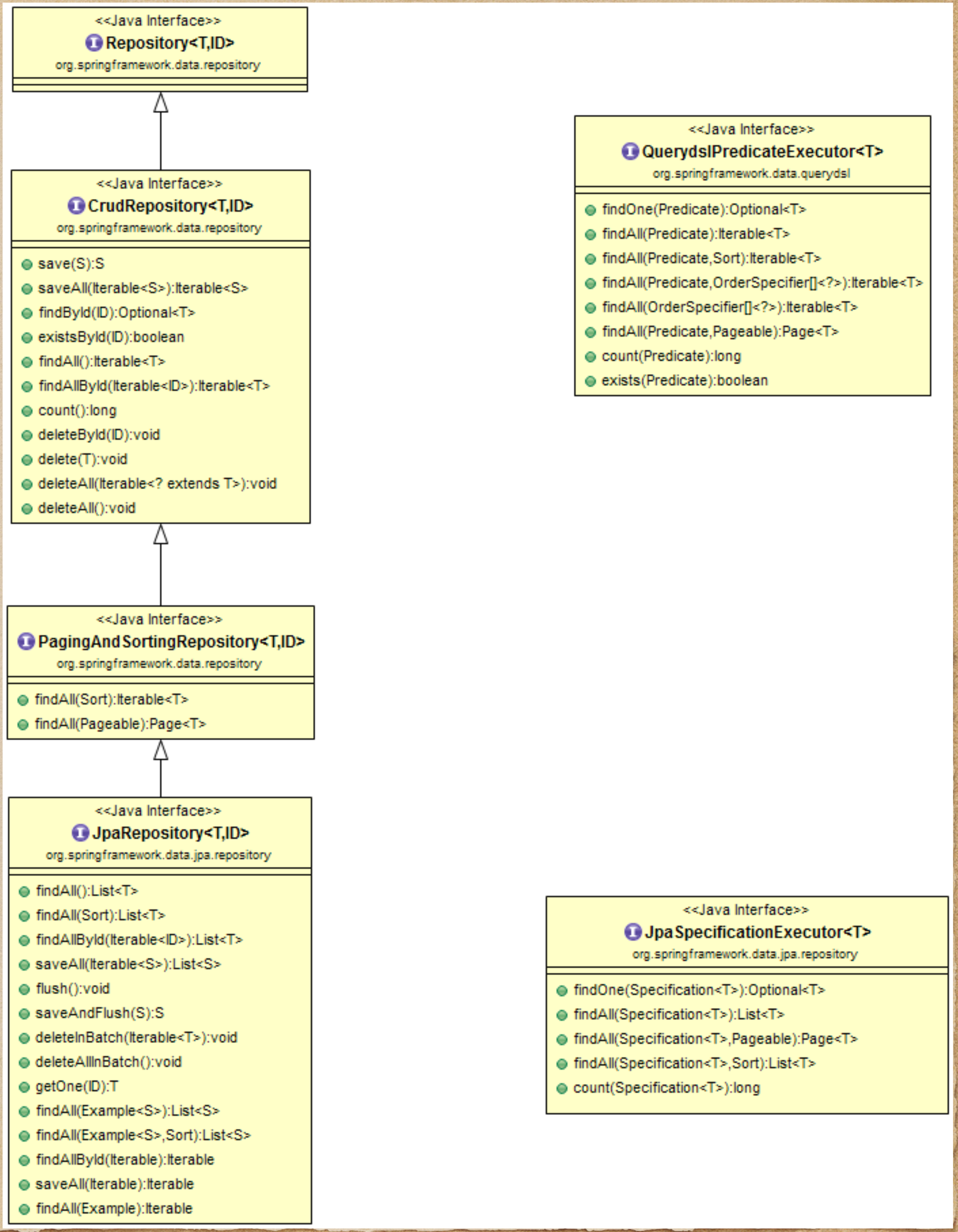
findAll()

findById()

save()

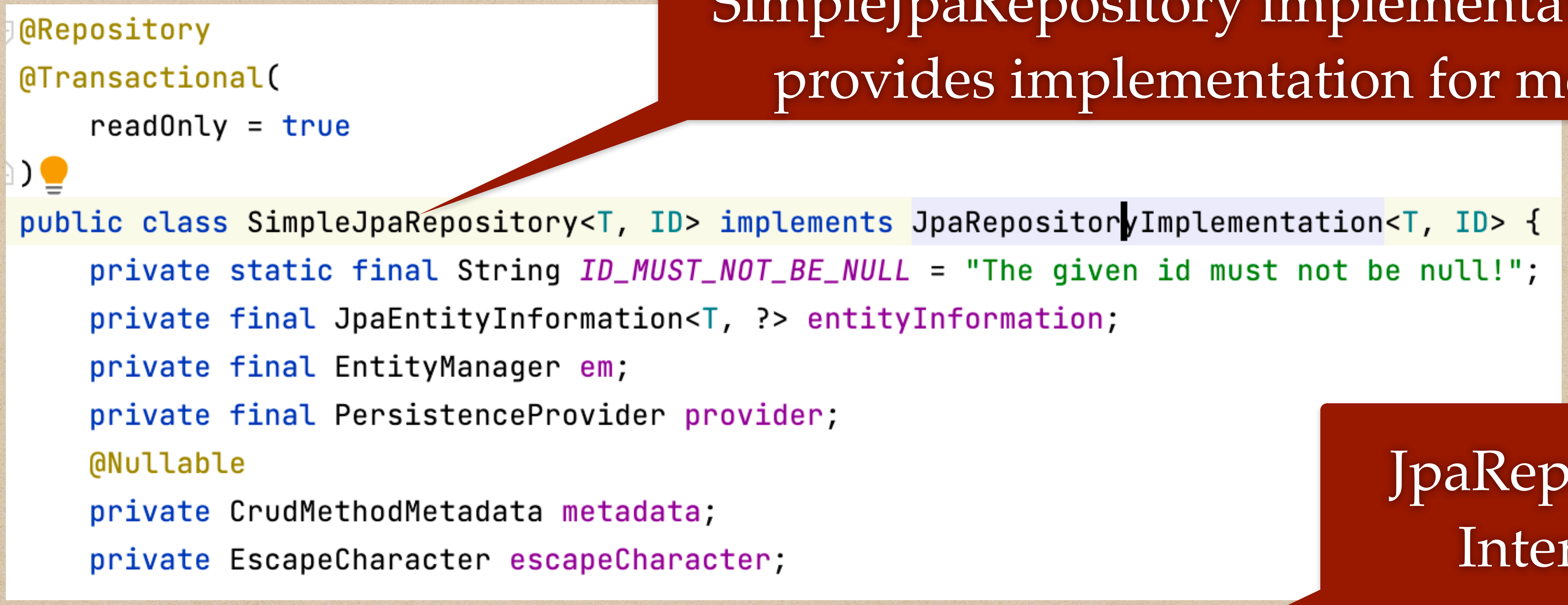
deleteById()

- - - - - -



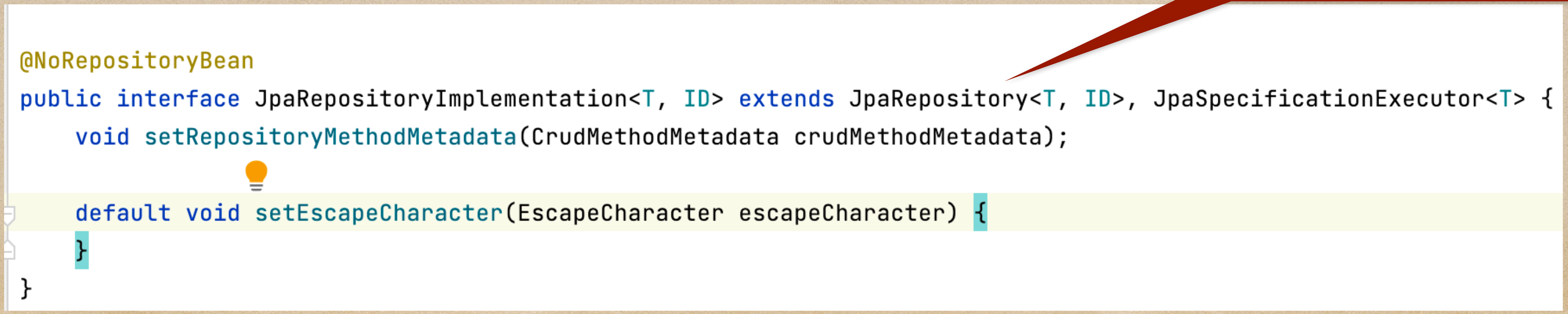
<<Java Interface>>
Repository<T,ID>
org.springframework.data.repository
<<Java Interface>>
CrudRepository<T,ID>
org.springframework.data.repository
save(S):S
saveAll(Iterable<S>):Iterable<S>
findById(ID):Optional<T>
existsById(ID):boolean
findAll():Iterable<T>
findAllById(Iterable<ID>):Iterable<T>
count():long
deleteById(ID):void
delete(T):void
deleteAll(Iterable<? extends T>):void
deleteAll():void
<<Java Interface>>
PagingAndSortingRepository<T,ID>
org.springframework.data.repository
findAll(Sort):Iterable<T>
findAll(Pageable):Page<T>
<<Java Interface>>
JpaRepository<T,ID>
org.springframework.data.jpa.repository
findAll():List<T>
findAll(Sort):List<T>
findAllById(Iterable<ID>):List<T>
saveAll(Iterable<S>):List<S>
flush():void
saveAndFlush(S):S
deleteInBatch(Iterable<T>):void
deleteAllInBatch():void
getOne(ID):T
findAll(Example<S>):List<S>
findAll(Example<S>,Sort):List<S>
findAllById(Iterable):Iterable
saveAll(Iterable):Iterable
findAll(Example):Iterable
<<Java Interface>>
QuerydslPredicateExecutor<T>
org.springframework.data.querydsl
findOne(Predicate):Optional<T>
findAll(Predicate):Iterable<T>
findAll(Predicate,Sort):Iterable<T>
findAll(Predicate,OrderSpecifier[]<?>):Iterable<T>
findAll(OrderSpecifier[]<?>):Iterable<T>
findAll(Predicate,Pageable):Page<T>
count(Predicate):long
exists(Predicate):boolean
<<Java Interface>>
JpaSpecificationExecutor<T>
org.springframework.data.jpa.repository
findOne(Specification<T>):Optional<T>
findAll(Specification<T>):List<T>
findAll(Specification<T>,Pageable):Page<T>
findAll(Specification<T>,Sort):List<T>
count(Specification<T>):long

JpaRepository Implementation



```
@Repository
@Transactional(
    readOnly = true
)
public class SimpleJpaRepository<T, ID> implements JpaRepositoryImplementation<T, ID> {
    private static final String ID_MUST_NOT_BE_NULL = "The given id must not be null!";
    private final JpaEntityInformation<T, ?> entityInformation;
    private final EntityManager em;
    private final PersistenceProvider provider;
    @Nullable
    private CrudMethodMetadata metadata;
    private EscapeCharacter escapeCharacter;
```

SimpleJpaRepository implementation class provides implementation for methods



```
@NoRepositoryBean
public interface JpaRepositoryImplementation<T, ID> extends JpaRepository<T, ID>, JpaSpecificationExecutor<T> {
    void setRepositoryMethodMetadata(CrudMethodMetadata crudMethodMetadata);

    default void setEscapeCharacter(EscapeCharacter escapeCharacter) {
    }
}
```

JpaRepository Interface

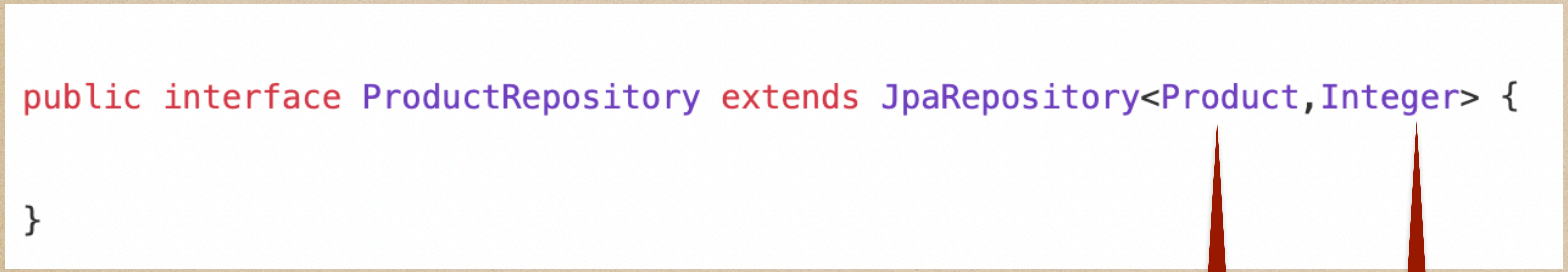
Steps to create and use Spring Data JPA Repository

1. Create a repository interface and extend to JpaRepository interface

2. Add custom query methods to the created repository interface (if we need them)

3. Inject the repository interface to another component and use the implementation that is provided automatically by Spring Data Jpa.

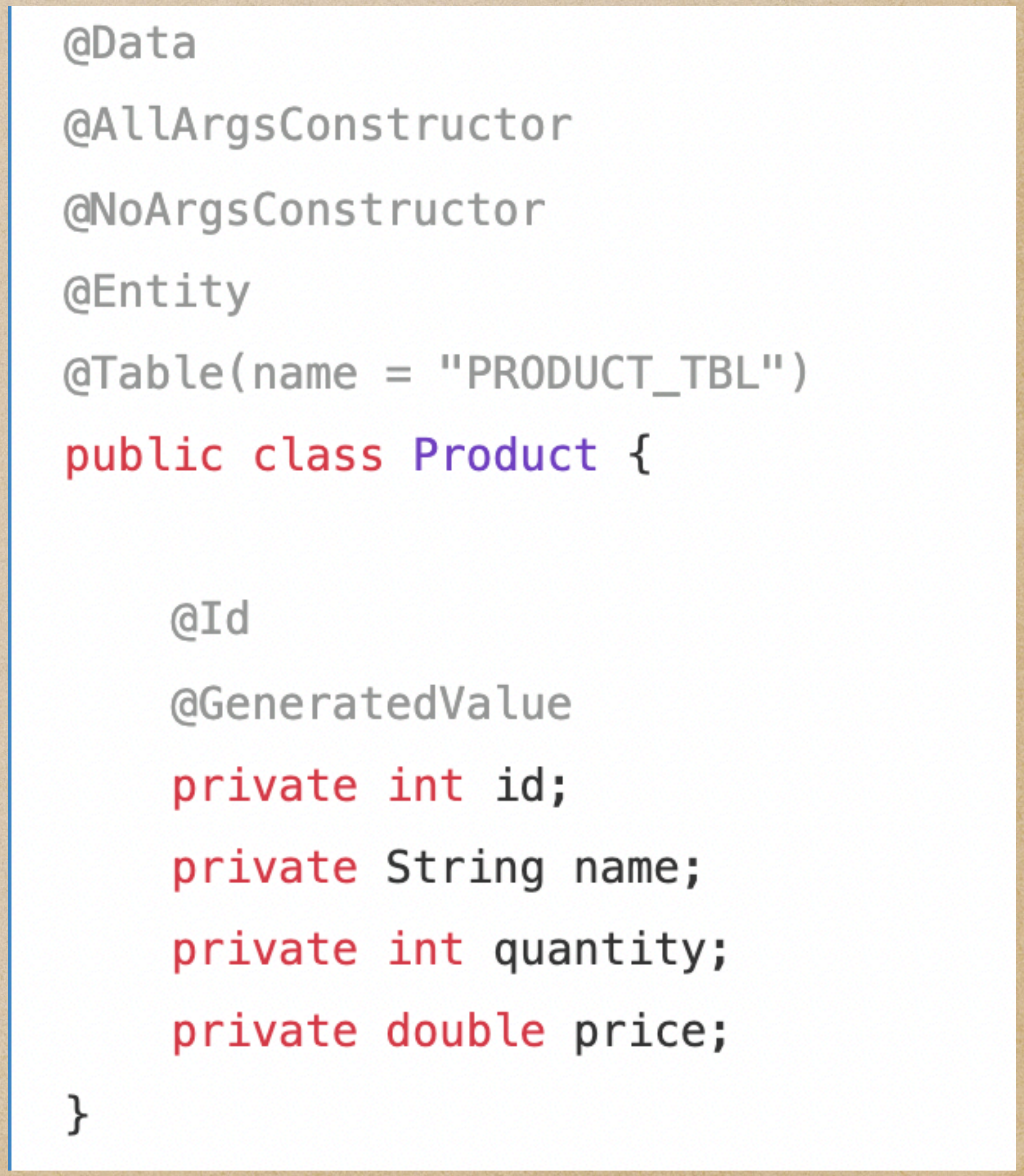
1. Create a repository interface and extend to JpaRepository interface



```java
public interface ProductRepository extends JpaRepository<Product,Integer> {

}
```

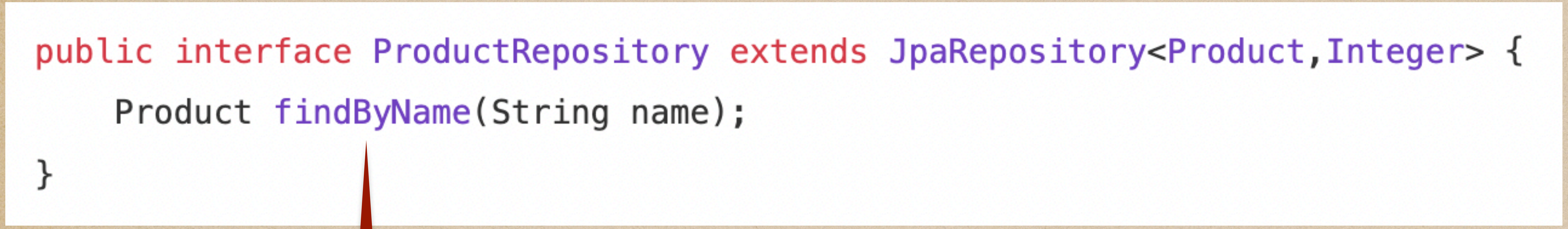
JPA Entity

Primary Key



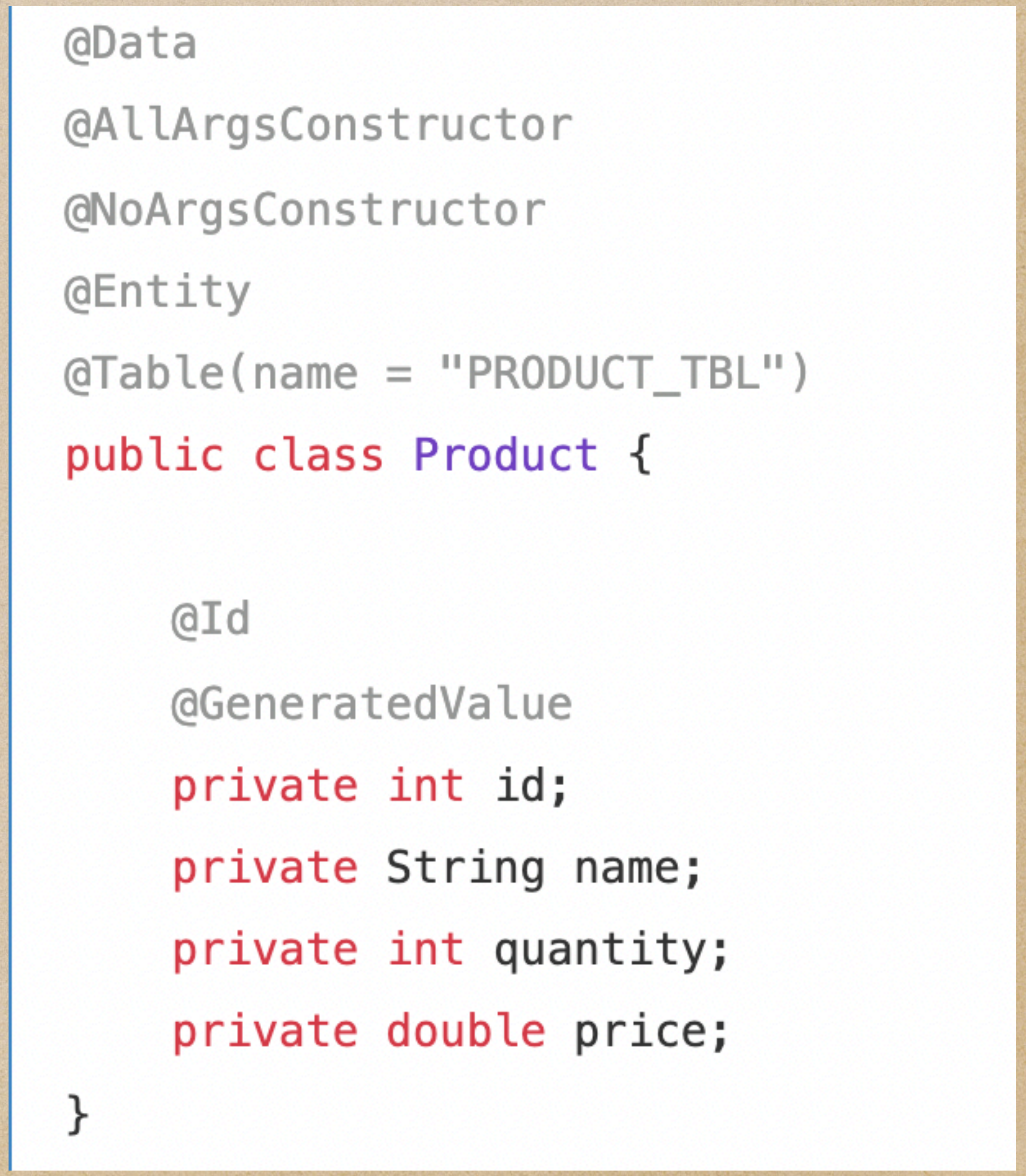
```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "PRODUCT_TBL")
public class Product {

    @Id
    @GeneratedValue
    private int id;
    private String name;
    private int quantity;
    private double price;
}
```

2. Add custom query methods to the created repository interface (if we need them)



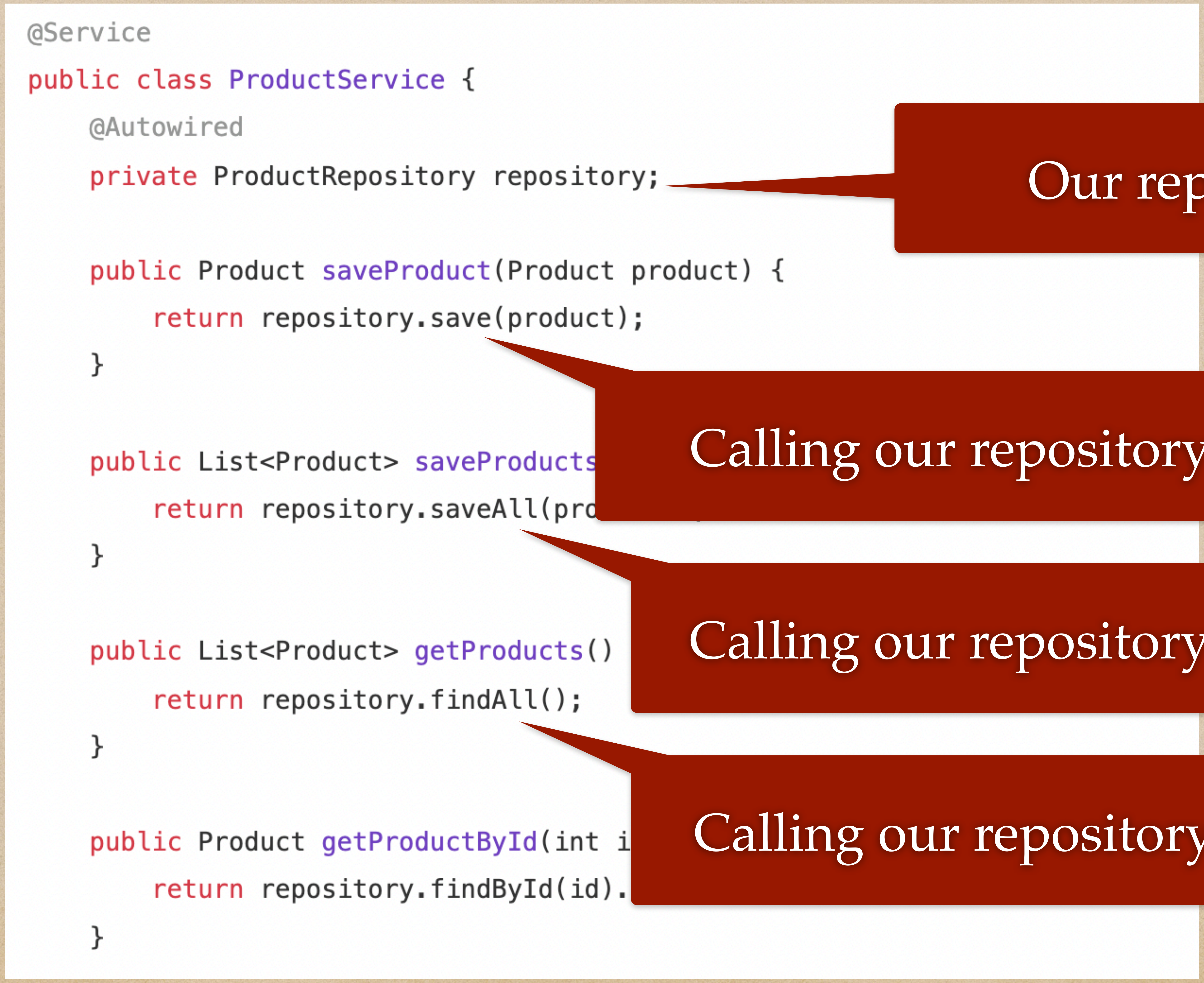
```
public interface ProductRepository extends JpaRepository<Product,Integer> {
    Product findByName(String name);
}
```

Query method or finder method



```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "PRODUCT_TBL")
public class Product {

    @Id
    @GeneratedValue
    private int id;
    private String name;
    private int quantity;
    private double price;
}
```

3. Using Repository interface in our project



```
@Service
public class ProductService {
    @Autowired
    private ProductRepository repository;

    public Product saveProduct(Product product) {
        return repository.save(product);
    }

    public List<Product> saveProducts
        return repository.saveAll(pro
    }

    public List<Product> getProducts()
        return repository.findAll();
    }

    public Product getProductById(int i
        return repository.findById(id).
    }
```

Our repository

Calling our repository save() method

Calling our repository saveAll() method

Calling our repository findAll() method

Minimised boilerplate code

Before Spring Data JPA

```java
public interface EmployeeDAO {

        public List<Employee> findAll();

        public Employee findById(int theId);

        public void sav

        public void del

}
```

```java
@Repository
public class EmployeeDAOJpaImpl implements EmployeeDAO {

        private EntityManager entityManager;

        @Autowired
        public EmployeeDAOJpaImpl(EntityManager theEntityManager) {
                entityManager = theEntityManager;
        }

        @Override
        public List<Employee> findAll() {
                // create a query
                Query theQuery = entityManager.createQuery("from Employee");
                // execute query and get result list
                List<Employee> employees = theQuery.getResultList();
                // return the results
                return employees;
        }

        @Override
        public Employee findById(int theId) {
                // get employee
                Employee theEmployee = entityManager.find(Employee.class, theId);
                // return employee
                return theEmployee;
        }

        @Override
        public void save(Employee theEmployee) {
                // save or update the employee
                Employee dbEmployee = entityManager.merge(theEmployee);
                // update with id from db ... so we can get generated id for save/insert
                theEmployee.setId(dbEmployee.getId());
        }

        @Override
        public void deleteById(int theId) {
                // delete object with primary key
                Query theQuery = entityManager.createQuery("delete from Employee where id=:employeeId");
                theQuery.setParameter("employeeId", theId);
                theQuery.executeUpdate();
        }
}
```

2 Files
30+ lines of code

After Spring Data JPA

```java
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

        // that's it ... no need to write any code LOL!

}
```

1 File
3 lines of code

No need for implementation Class