Spring Data Jpa Specification interface

It allows developers to create type safe and dynamic queries using jpa criteria api. It enables us to build queries on various criteria which can be combined and used in various scenarios providing maintainability and flexibility in constructing queries.

It is a functional interface with single instance method and many default and static methods.

```
@Nullable
Predicate toPredicate(Root<T> root, CriteriaQuery<?>  query, CriteriaBuilder criteriaBuilder);
```

T -> entity type on which query is built.
Root -> represents the root entity of the query from which we can navigate to other related entities and attributes.
CriteriaQuery -> represents the overall query that is being built. It is used to modify the query structure or add additional constraints.
CriteriaBuilder -> object that acts as a factory for creating query elements like predicates, extensions, and orderings.

The method returns a Predicate which is a boolean expression representing the query condition.

To use it first of all we have to extend the repository with JpaSpecificationExcecutor interface (provides methods to perform queries using specifications) along with jpa repository.

We can also create custom specification instances by implementing the specification or by using lambda expression. Specification cab be combined using and or or method. We also have not method.

```
class AuthorRepository{

public static Specification<Author> hasAge(int age) {
	return (Root<Author> root, CriteriaQuery<?> q, CriteriaBuilder cb) -> {
		if (age < 0) return null;
		return cb.equal(root.get(‘age’), age);
	}
}

public static Specification<Author> firstNameContains(String firstName) {
	return (Root<Author> root, CriteriaQuery<?> q, CriteriaBuilder cb) -> {
		if (firstName == null) return null;
		return cb.like(root.get(‘firstName’), “%” + firstName + “%”);
```

```
        }
}

}
```

Now we can use this as:

```
Specification<Author> spec =
Specification.where(AuthorSpecification.hasAge(34)).and(AuthorSpecification.f
irstNameContains(firstName));

List<Author> authors = repository.findAll(spec);
```