1. 람다식

1. 람다식이란

1. 함수(메소드==기능==펑션)형 프로그래밍이란 메소드를 만들고 만든 메소드를 통해서 데이터를 처리하는 기법.
2. 메소드의 처리부에서는 정해진 방식은 존재하지 않고 데이터를 가공하거나 연산 또는 리턴등의 처리를 담당한다. 메소드A와 메소드B의 처리부가 다르면 같은 데이터를 처리해도 다른 결과가 나올 수도 있다. 이러한 현상을 데이터 처리의 다형성이라고 한다.
3. 자바에서는 함수형 프로그래밍을 지원하기 위해서 람다식을 제공.
4. 람다식은 데이터 처리부에서 데이터를 가공하거나 연산, 리턴 하는 함수 역할을 하는 매개변수를 갖는 중괄호 블록이다.
5. Spring에서 지원하는 인터페이스중 무조건 람다식이나 익명객체로 구현되어야 하는 인터페이스들이 있어서 람다식은 필수적으로 사용법을 알아하고 익명객체보다 구현이 간단하기 때문에 람다식을 주로 사용한다.

2. 람다식의 선언

1. 람다식 : (매개변수) -> {처리부(처리내용)} 메소드명(매개변수) { 처리부 } public int add(int a, int b) { return a + b; } 람다식 변환 => (a, b) -> {return a + b;}

3. 매개변수가 없는 람다식

1. 함수형 인터페이스에 추상 메소드가 매개변수가 없는 경우 람다식은 아래와 같이 작성한다.
    - () -> { 실행문; 실행문; }
    - () -> 실행문;

4. 매개변수가 있는 람다식

1. (타입 매개변수) -> { 실행문; 실행문; } (타입 매개변수) -> 실행문;
2. (var 매개변수) -> { 실행문; 실행문; } (var 매개변수) -> 실행문;
3. (매개변수) -> { 실행문; 실행문; } (매개변수) -> 실행문;
4. 매개변수의 개수가 하나일 경우 ()생략 가능 매개변수 -> { 실행문; 실행문; } 매개변수 -> 실행문;

5. 리턴 값이 있는 람다식

1. (매개변수, ...) -> { 실행문; return 값; }
2. (매개변수, ...) -> 값
    - 중괄호가 없을 경우 return구문없이 값만 적어야 리턴함.

6. 메소드의 참조

1. 람다식을 사용해서 메소드를 구현할 때 직접 구현하는 것보다 다른 클래스에 있는 메소드를 참조하여 사용하는 것이 소스코드를 줄일 수 있는 방법중 하나다.
   ex) (int a, int b) -> Math.max(a, b) => action(Math :: max);
2. static 메소드 참조
    - 클래스명 :: 메소드명
3. 일반 메소드 참조
    - 클래스 타입의 참조변수 = 인스턴스화();
    - 참조변수 :: 메소드명

7. 매개변수의 메소드 참조

```
1. (String a, String b) - > {
        syso(a.concat(b))
   } =>
   (String :: concat)
```

8. 생성자 참조

1. (매개변수, ...) -> {return new 클래스(매개변수, ...)}
2. 클래스 :: new
3. 생성자가 여러 개 존재할 때 함수형 인터페이스에 존재하는 메소드와 매개변수의 타입과 개수, 순서가 같은 생성자를 호출
4. 함수형 인터페이스에 존재하는 메소드와 매개변수의 타입과 개수, 순서가 같은 생성자가 존재하지 않으면 에러가 발생.

9. 람다식 사용 목적

1. 인터페이스 구현 클래스를 만들지 않고 구현체를 함수형 프로그래밍으로 구현하기 위한 하나의 수단.(익명객체, 람다식)

2. 람다식은 익명객체보다 코드가 간결하기 때문에 주로 람다식을 많이 사용한다.
3. 스트림처리를 위해서 사용한다. 스트림 메소드들이 매개변수로 함수형 인터페이스를 가지고 있기 때문에 함수형 인터페이스 구현체를 람다식으로 구현한다.
4. 함수형 인터페이스에 있는 메소드를 참조하여 구현하고
5. 매개변수 -> 구현내용
6. 클래스, 객체, 생성자의 클래스 :: 메소드명, new