JAVA 기초입문과정

CHAPTER14 람다식

Contents

람다식

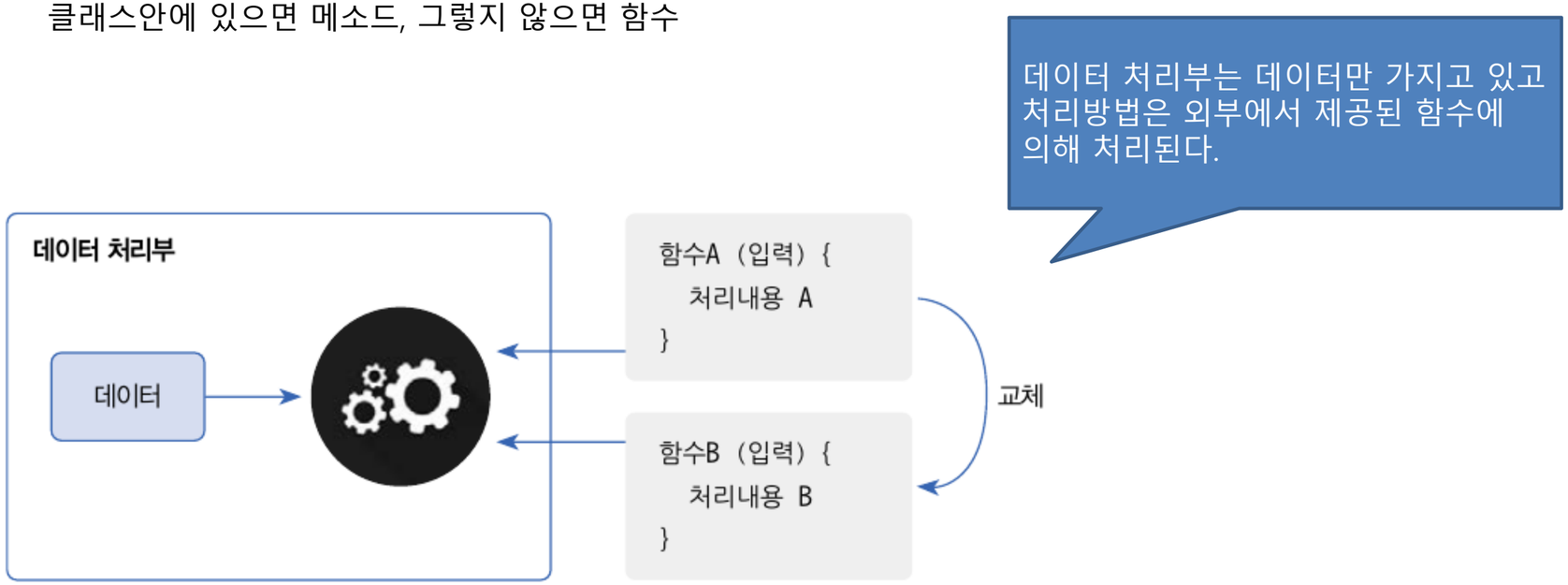

PART
01
“
람다식
”

PART 01

람다식

1. 람다식이란?
   - 메소드를 하나의 식으로 표현한 것.
   - 함수명을 선언 -> 함수형 프로그램

<함수와 메소드의 차이>
클래스안에 있으면 메소드, 그렇지 않으면 함수



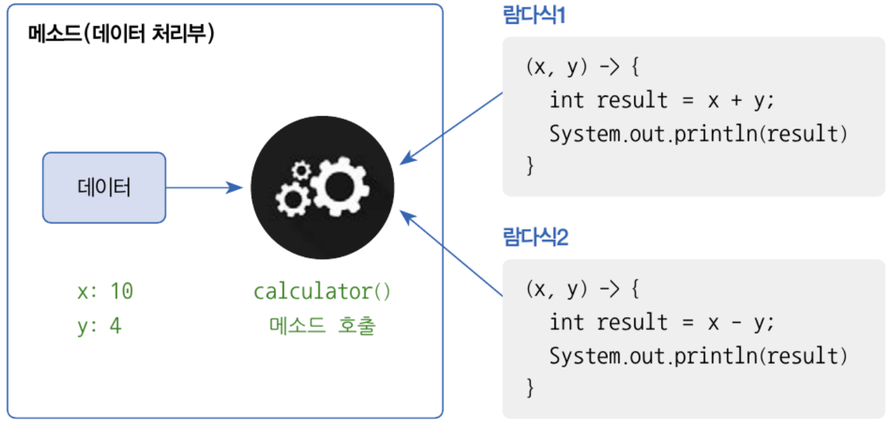
함수A에서 처리한 결과와 함수B에서 처리한 결과가 다를 수 있다.(데이터 처리의 다형성)

람다식 구현 과정

```
public interface Calculable {
  //추상 메소드
  void calculate(int x, int y);
}
```

람다식 표현

```
(x, y) -> { 처리내용 };
```




```
public void action(Calculable calculable) {
  int x = 10;
  int y = 4;
  calculable.calculate(x, y);   //데이터를 제공하고 추상 메소드를 호출
}
```

람다식으로 표현

```
action( (x, y) -> {
  int result = x + y;
  System.out.println(result);
});
```

익명 구현 객체로 표현

```
new Calculable() {
  @Override
  public void calculate(int x, int y) { 처리내용 }
};
```

람다식

인터페이스의 익명 구현 객체를 람다식으로 표현을 하려면 인터페이스가 단 하나의 추상 메소드만 가져야 한다.

```
public interface RemoteControl {
  void turnOn();
  void turnOff();
}
```

인터페이스가 단 하나의 추상 메소드를 가질 때, 이를 함수형 인터페이스 라고 한다.

인터페이스

```
public interface Runnable {
  void run();
}
```

람다식

```
( ) -> { … }
```

@Functional Interface
컴파일 과정에서 추상메소드가 하나인지 검사

인터페이스

```
@FunctionalInterface
public interface Calculable {
  void calculate(int x, int y);
}
```

람다식

```
( x, y ) -> { … }
```

PART 01

람다식

1. 람다식이란?
   - 메소드를 하나의 식으로 표현한 것.
   - 함수명을 선언, 사용하지 않는다.

```
람다식: (매개변수, ...) -> { 처리내용 }
```

2. 매개변수가 없는 람다식

```
()-> {
        실행문;
        실행문;
 }
```

```
()-> 실행문;
```

실행문이 하나일 경우
중괄호 생략 가능!

람다식

3. 매개변수가 있는 람다식

```
(타입 매개변수, ...) -> {
        실행문;
        실행문;
 }
```

```
(타입 매개변수, ...) -> 실행문;
```

```
(var 매개변수, ...) -> {
        실행문;
        실행문;
 }
```

```
(var 매개변수, ...) -> 실행문;
```

```
(매개변수, ...) -> {
        실행문;
        실행문;
 }
```

```
(매개변수, ...) -> 실행문;
```

가장 많이 사용해요!
(타입 생략)

```
매개변수 -> {
        실행문;
        실행문;
 }
```

```
매개변수 -> 실행문;
```

매개변수가 하나일 경우
괄호를 생략할 수 있어요!

람다식

4. 리턴 값이 있는 람다식

```
(매개변수, ...) -> {
        실행문;
        return 값;
 }
```

```
(매개변수, ...) -> return 값
(매개변수, ...) -> 값
```

return 문 하나만 있을 경우 중괄호와 return을 생략할 수 있어요!

람다식

5. 메소드 참조

- 메소드를 참조해 람다식에서 불필요한 매개변수를 제거하는 것이 목적.

```
(left, right) -> Math.max(left, right);
```

단순히 두개의 값을 전달하는 역할만 할때는 메소드 참조를 이용해 이렇게 사용해요!

```
Math :: max
```

- 정적 메소드일 경우

```
클래스 :: 메소드
```

- 인스턴스 메소드일 경우

```
참조변수 :: 메소드
```

- 매개변수의 메소드 참조

```
(a, b) -> { a.instanceMethod(b); }
```

```
클래스 :: instanceMethod
```

a 매개변수의 메소드를 호출해서 b 매개변수를 매개 값으로 사용하는 경우

람다식

6. 생성자 참조

- 람다식이 객체를 생성하고 리턴한다면 생성자 참조로 대치 가능

(a, b) -> { return new 클래스(a, b); }

클래스 :: new

생성자가 오버로딩되어 있을 경우 컴파일러는 동일한 매개변수 타입과 개수를 가지고 있는 생성자를 찾아 실행 해요!

Thank You!