아이템 43
'람다보다는 메서드 레퍼런스를 사용하라'

'람다보다는 메서드 레퍼런스를 사용하라'

참고

- Effective Java Item 43
- Oracle java tutorial - generic function type

왜 람다 대신 메서드 레퍼런스를 사용하는가?

간결함에 대해서..

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

메서드 레퍼런스 5가지 유형

	메서드 참조 유형	예	같은 기능을 하는 람다
1	정적	Integer::parseInt	str -> Integer.parseInt(str)
2	한정적 (인스턴스)	Instant.now()::isAfter	Instant then = Instant.now(); t -> then.isAfter(t)
3	비한정적 (인스턴스)	String::toLowerCase	str -> str.toLowerCase()
4	클래스 생성자	TreeMap<K,V>::new	() -> new TreeMap<K,V>()
5	배열 생성자	int[]::new	len -> new int[len]

정적 메서드를 가리키는 메서드 참조

파라미터의 의미가 중요하지 않은 경우 또는
파라미터가 많은 경우 생략하여 간결한 코드를 작성하기 위해 또는
메서드 명 자체가 기능에 대한 유익한 의미를 나타내는 경우 메서드 레퍼런스를 사용한다.

```java
@DisplayName("메서드 참조 유형 - 정적 메서드 테스트")
@Test
void testCase1() {
    List<String> items = Arrays.asList("1", "2");

    Integer[] integers = items.stream()
//            .map(s -> Integer.parseInt(s))
            .map(Integer::parseInt)
            .toArray(Integer[]::new);

    assertThat(integers).containsExactly(1, 2);
}
```

```java
@DisplayName("메서드 참조 유형 - 정적 메서드 테스트2")
@Test
void testCase2() {
    List<String> items = Arrays.asList("1", "2");

    int[] ints = items.stream()
            .mapToInt(value -> new Integer(value))
//            .mapToInt(Integer::new)
            .toArray();

    assertThat(ints).containsExactly(1, 2);
}
```

매개 변수의 이름 자체가 프로그래머에게 좋은 가이드가 되는 경우 람다를 사용한다.
이때 람다는 코드의 양이 늘어나지만 유지보수의 관점에서 좋다.

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

정적 메서드를 가리키는 메서드 참조

```java
@DisplayName("Function.identity() 예제")
@Test
void testCase8() {

    List<DataContainer> dataContainers = Arrays.asList(
            new DataContainer(DataId.ID_1, data: "data1"),
            new DataContainer(DataId.ID_2, data: "data2"),
            new DataContainer(DataId.ID_3, data: "data3")
    );

    Map<DataId, DataContainer> collect = dataContainers.stream()
            .collect(
                    toMap(
                            DataContainer::getId,
                            java.util.function.Function.identity()
//                              (x -> x)
                    )
            );

    System.out.println(collect);
}
```

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

한정적 (인스턴스)를 가리키는 메서드 레퍼런스

```java
@DisplayName("메서드 참조 유형 - 한정적 인스턴스 테스트1")
@Test
void testCase3() {
    Instant now = Instant.now();

    boolean after = now.isAfter(Instant.now());

    assertThat(after).isFalse();
}
```

참조되는 메서드가 받는 인수

함수 객체가 받는 인수

```java
@DisplayName("메서드 참조 유형 - 한정적 인스턴스 테스트2")
@Test
void testCase4() throws InterruptedException {
    List<Instant> instantList = Arrays.asList(Instant.now(), Instant.now());

    Thread.sleep( millis: 100);

    Instant now = Instant.now();

    // 과거 데이터와 현재 데이터의 값 비교
    instantList.forEach(item ->
            assertThat(item.isBefore(now)).isTrue()
    );
}
```

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

비한정적 (인스턴스)를 가리키는 메서드 레퍼런스

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

```java
@DisplayName("메서드 참조 유형 - 비한정적 인스턴스 테스트1")
@Test
void testCase5() {
    String[] lowerStr = {"hello", "world"};

    String[] strings = Arrays.stream(lowerStr)
            .map(str -> str.toUpperCase())
            .toArray(String[]::new);

    assertAll(
            () -> assertThat(strings[0]).isEqualTo(lowerStr[0].toUpperCase()),
            () -> assertThat(strings[1]).isEqualTo(lowerStr[1].toUpperCase())
    );
}
```

함수 객체를 적용하는 시점

수신 객체

클래스 생성자를 가리키는 메서드 레퍼런스

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

```java
@DisplayName("메서드 참조 유형 - 클래스 생성 테스트")
@Test
void testCase6() {

    Supplier<TreeMap> treeMapSupplier = () -> new TreeMap<>();
    Factory<TreeMap> treeMapFactory = () -> new TreeMap<>();

    assertThat(treeMapSupplier).isInstanceOf(Supplier.class);
    assertThat(treeMapFactory).isInstanceOf(Factory.class);
}
```

클래스 생성자를 가리키는 메서드 참조

배열 생성자를 가리키는 메서드 레퍼런스

- Lambda vs Method Reference

1.익명 클래스 vs 람다

2.람다 vs 메서드 레퍼런스

3.메서드 레퍼런스의 5가지 유형

```java
@DisplayName("메서드 참조 유형 - 배열 생성 테스트")
@Test
void testCase7() {
    Function<Integer, int[]> intSupplier = len -> new int[len];

    int[] apply = intSupplier.apply( t: 3);

    assertThat(apply).hasSize(3);
}
```

배열 생성자를 가리키는 메서드 참조

정리

정리

- 메서드 참조가 짧고 메서드 기능에 대한 명시가 필요한 경우 메서드 참조를 사용한다.
- 인수의 의미를 명시하는 것이 더 중요한 경우 람다를 사용한다.
- 람다로 불가능하고 메서드 참조로 가능한 유일한 경우는 제네릭 함수 타입(generic function type)이다.