자바 패턴 매칭 라이브러리

JPatternMatch

Scala의 match / TypeScript의 ts-pattern과 같은
패턴 매칭 기능을 자바에서 구현했습니다.

* Pattern Matching at Scala

: 스칼라에서의 패턴 매칭

```scala
def example(a: Int) {
    a match {
        case 1 => print("One")
        case 2 => print("Two")
        case _ => print("Other")
    }
}
```

```scala
def example2(a: Any) {
    a match {
        case target:Int if (target > 10) => print("over ten")
        case 2 => print("Two")
        case target : String => println(a + " is String" )
        case _ => printf("Others")
    }
}
```

(1) Java 16 instanceof 이전

: 자바 16에 포함된 패턴 매칭 기능 1.8 내 사용

Java 16

```java
// Java 16 instanceof 연산자에 대한 pattern matching
    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Student student) {
            if (this.studentNumber == student.studentNumber) {
                return true;
            } else {
                return false;
            }
        }
        return false;
    }
```

JPattern Match

asTypeOf

- 자바 13의 패턴 매칭 `instanceOf`를 모방한 함수입니다.
- 타입 체크 / 반환을 동시에 하는 패턴 매칭의 기능을 구현합니다.

```java
// Null check and excetue : Runnable
asTypeOf(1, Integer.class, () -> System.out.println("expression lambda"));

// Null Check and Return Result : Function
Integer result = asTypeOf(input, String.class, Integer::parseInt);
```

- `Runnable`의 경우 확인 객체, 확인 클래스 타입, 실행 함수의 순서로 구성됩니다.
- `Function`의 경우 확인 객체, 확인 클래스 타입, 반환 객체 타입의 순서로 구성됩니다.

(2) ts-pattern to Java 1.8

: ts-pattern 내부 패턴 매칭과 유사한 기능 1.8 내 사용

JPattern Match

`of(instance)`

```
// initialize JPatternMatch instance
JPatternMatch.of(instance)
```

- 비교할 instance를 토대로 객체를 생성합니다.
- instance에는 `null` 을 주입할 수 없습니다.
- 생성된 객체는 초기화된 `JPatternMatch` 클래스를 반환합니다.

`.match()`

```
// Start matching
JPatternMatch.of(instance)
             .match()
```

- 생성된 JPatternMatch 인스턴스에서 matching 시작을 선언합니다.

ts-pattern

```
const sanitize = (name: string) =>
  match(name)
    .with('text', 'span', 'p', () => 'text')
    .with('btn', 'button', () => 'button')
    .otherwise(() => name);
```

(2) ts-pattern to Java 1.8

: ts-pattern 내부 패턴 매칭과 유사한 기능 1.8 내 사용

JPattern Match

`.with(instance, lambda expression)`

```java
// Set Compare Instance
JPatternMatch.of(instance)
            .with(testInstance, () -> System.out.println("첫 번째 매치 실행"))
```

- 주입 객체와 비교 대상 객체를 비교합니다.
- 비교 대상 객체의 기준은 메모리 주소가 아닌 `객체의 상태 값`입니다.
- returnObject의 경우 return을 통해 반환 값을 지정해야 합니다.

ts-pattern

`.with`

```
match(...)
  .with(pattern, [...patterns], handler)
```

```ts
const sanitize = (name: string) =>
  match(name)
    .with('text', 'span', 'p', () => 'text')
    .with('btn', 'button', () => 'button')
    .otherwise(() => name);
```

(2) ts-pattern to Java 1.8

: ts-pattern 내부 패턴 매칭과 유사한 기능 1.8 내 사용

JPattern Match

`.returnObject(Class)`

```
// Set Return Type
Instance instance = JPatternMatch.of(instance)
                                 .match()
                                 .returnObject(instance.class)
```

- 수행 함수의 결과를 반환받고 싶은 경우, 반환 객체의 타입을 지정합니다.

ts-pattern

`.returnType`

```
match(...)
  .returnType<string>()
  .with(..., () => "has to be a string")
  .with(..., () => "Oops".length)
  //               ~~~~~~~~~~~~~ ❌ `number` isn't a string!
```

The `.returnType<SomeType>()` method allows you to control the return type of all of your branches of code. It accepts a single type parameter that will be used as the return type of your `match` expression. All code branches must return values assignable to this type.

(2) ts-pattern to Java 1.8

: ts-pattern 내부 패턴 매칭과 유사한 기능 1.8 내 사용

JPattern Match

`otherwise(lambda expression)`

```
// Define default action
JPatternMatch.of(instance)
             .match()
             .otherwise(() -> System.out.println("기본 매치 실행"));
```

- 매칭되는 객체가 없을 시 해당 함수를 수행합니다.
- `Runnable`의 경우 함수 수행 후 객체가 만료됩니다.
- `Function`의 경우 함수 수행 후 객체를 반환받습니다.
- 함수 수행 시 `객체를 변경하는 것은 가능`하나, `기본 타입(Primitive Type)` 변경은 불가능합니다.

ts-pattern

`.otherwise`

```
match(...)
  .with(...)
  .otherwise(defaultHandler)
```

Runs the pattern-matching expression with a default handler which will be called if no previous `.with()` clause match the input value, and returns the result.

(2) ts-pattern to Java 1.8

: ts-pattern 내부 패턴 매칭과 유사한 기능 1.8 내 사용

JPattern Match

`exhaustive()`

```
// Define exhaustive match
JPatternMatch.of(instance)
             .match()
             .exhaustive(); // PatternMatchException!
```

- 매칭되는 객체가 있을 경우 해당 매칭 함수를 수행하거나 결과를 반환합니다.
- 매칭되는 객체가 없을 시 `PatternMatchException` 을 발생시킵니다.

ts-pattern

`.exhaustive`

```
match(...)
  .with(...)
  .exhaustive()
```