

# 일반적으로 통용되는 명명 규칙을 따르라

# 명명 규칙

- 자바의 명명 규칙은 크게 **철자**와 **문법** 두 범주로 나뉜다

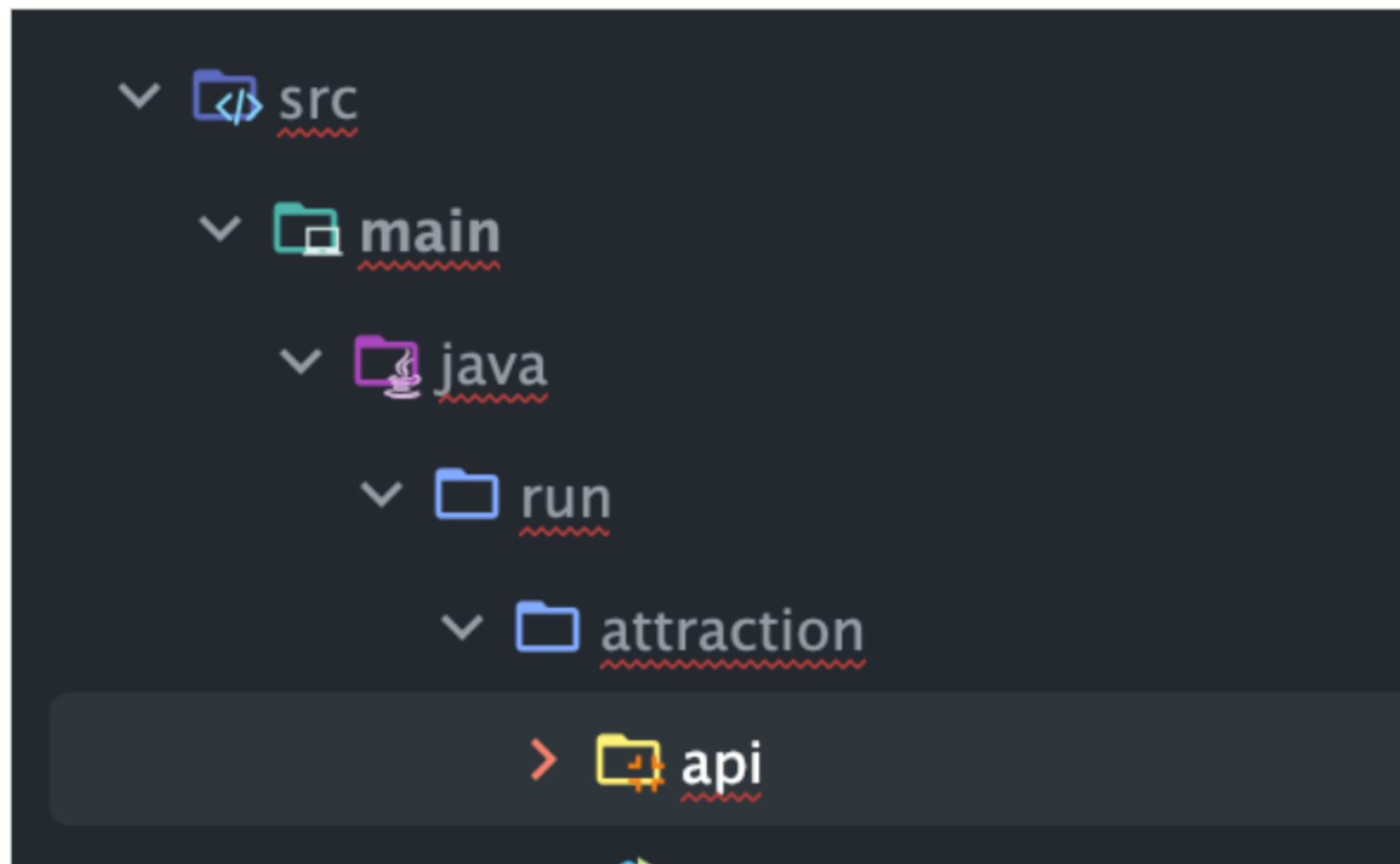
## 절차 규칙

- 패키지
- 클래스와 인터페이스
- 메서드와 필드 이름

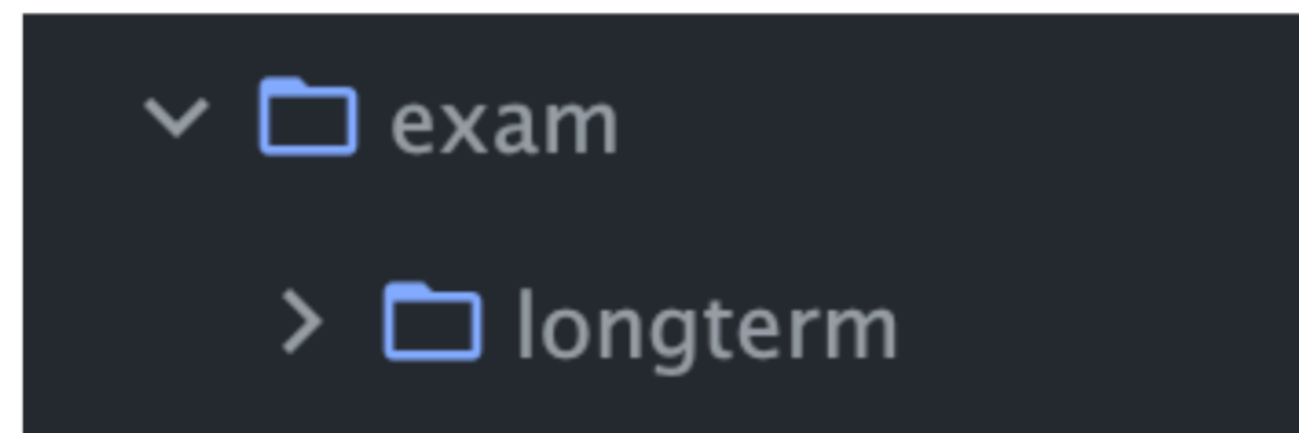
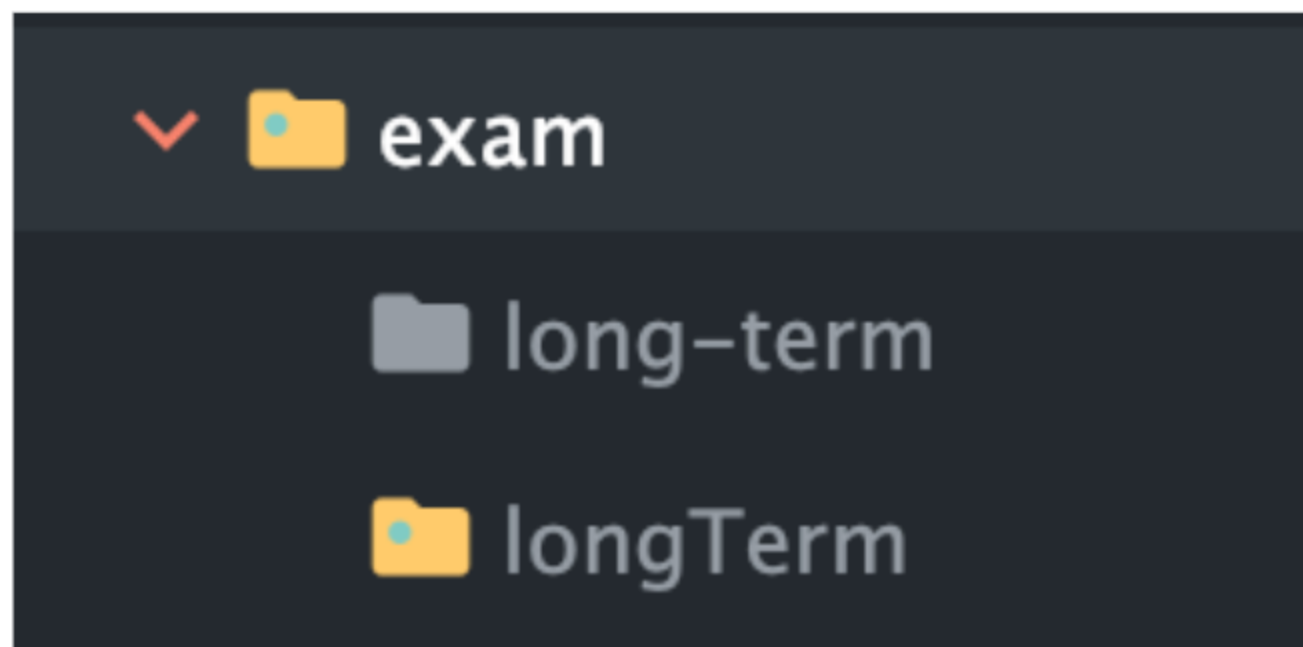
## 문법 규칙

- 클래스와 인터페이스
- 메서드

**package** run.attraction.api



조직 바깥에서 사용될 패키지라면 조직의 인터넷 도메인 이름을 역순으로 사용한다



요소들은 모두 소문자 알파벳 혹은 숫자

```
package exam.longterm;
```

```
import org.springframework.util.Assert;
```

2 related problems

```
public class Test {
```

```
    public static void main(String[] args) {
```

```
        Assert.hasText(text: "test", message: "spring-framework, springFramework");
```

```
    }
```

```
}
```

**합성어의 경우:** 각 단어의 첫 글자만 따서 써도 좋다(awt)

```
import java.util.  
ServiceLoader<? java.util.  
2 related problems  
public class T  
    public stati  
}  
}
```

- concurrent.
- function.
- jar.
- logging.
- prefs.
- random.
- regex.
- spi.
- stream.
- zip.

\*  
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards Next Tip

각 요소는 일반적으로 8자 이하의 짧은 단어로 하되, 통용되는 약어를 추천한다(utilities -> util)

# 절차 규칙

- 여러 단어의 첫 글자만 딴 약자나 널리 통용되는 약어(max, min)을 제외하고는 단어를 줄여 쓰지 않도록 한다

```
Http|
⌘ HttpServletRequest jakarta.servlet.http
⬢ HttpServlet jakarta.servlet.http
⚡ HttpStatus org.springframework.http
⬢ HttpHeaders org.springframework.http
⬢ HttpHeaders java.net.http
⌘ HttpServletResponse jakarta.servlet.http
⌘ HttpSession jakarta.servlet.http
⬢ HttpClient java.net.http
⚡ HttpURLConnectionException java.net.http
⬢ HttpCookie java.net
⬢ HttpRequest java.net.http
⌘ HttpServletResponse java.net.http
Press ^Space to see non-imported classes Next Tip
```

하나 이상의 단어로 이뤄지며, 각 단어는 대문자로 시작한다



```
URL
* URL java.net
URL javax.print.DocFlavor
URLDecoder java.net
URLEncoder java.net
* URLEncoder org.apache.catalina.util
URLClassLoader java.net
URLConnection java.net
* URLPermission java.net
URLConnectionHandler java.net
* URLStreamHandlerFactory java.net
URLConnectionHandlerProvider java.net.spi
```

```
public static void main(String[] args) {
    HTML
    * HTML javax.swing.text.html
    * HTMLDocument javax.swing.text.html
    * HTMLToolkit javax.swing.text.html
    * HTMLWriter javax.swing.text.html
    * HTMLAnchorElement org.w3c.dom.html
    * HTMLFrameHyperlinkEvent javax.swing.text.html
    * HTMLAppletElement org.w3c.dom.html
    * HTMLAreaElement org.w3c.dom.html
    * HTMLBaseElement org.w3c.dom.html
    * HTMLBaseFontElement org.w3c.dom.html
    * HTMLBodyElement org.w3c.dom.html
    * HTMLDocument org.w3c.dom.html
    Press ← to insert, → to replace Next Tip
```

약어의 경우 첫 글자를 대문자로만 하는 편이 일반적이다(HttpUrl, Db)



상수 필드를 구성하는 단어는 모두 대문자로 쓰며, 단어 사이를 밑줄



```
public class ConstantsExample {  
    // 기본 타입 상수  
    public static final int MAX_COUNT = 100;  
    public static final double PI = 3.14159;  
  
    // 불변 참조 타입 상수  
    public static final String DEFAULT_USER_NAME = "Guest";  
    public static final List<String> DEFAULT_ROLES = List.of("USER", "GUEST");  
}
```

상수 필드는 `static final` 필드의 타입이 기본 타입이나 불변 참조 타입인 경우를 의미한다



이렇게 쓰지 마세요.

```
public class ConstantsExample {  
    public static final List<String> DEFAULT_ROLES = Arrays.asList("USER", "GUEST");  
  
    public static void main(String[] args) {  
        DEFAULT_ROLES.sort(Comparator.naturalOrder());  
        DEFAULT_ROLES.forEach(System.out::println);  
    }  
}
```

상수 필드는 static final 필드의 타입이 기본 타입이나 불변 참조 타입인 경우를 의미한다

# 절차 규칙


- 지역변수엔 약어를 써도 좋다



```
public void test(int indexLastNumber) {  
    for (int i = 0; i < indexLastNumber; i++) {  
    }  
    for (int index = 0; index < indexLastNumber; index++) {  
    }  
}
```

# 절차 규칙

- 입력 매개변수도 지역변수의 하나다.



```
public void test(int indexLastNumber) {  
    for (int i = 0; i < indexLastNumber; i++) {  
    }  
    for (int index = 0; index < indexLastNumber; index++) {  
    }  
}
```

입력 매개변수는 메서드 설명 문서에까지 등장하는 만큼 일반 지역변수보다는 신경을 써야 한다.

# 절차 규칙

- 타입 매개변수의 이름은 보통 한 문자로 표현한다
- 임의의 타입엔 T, 컬렉션 원소의 타입은 E, 맵의 키와 값에는 K와 V, 예외에는 X, 메서드의 반환 타입에는 R
- 그 외의 임의 타입의 시퀀스에는 T, U, V 혹은 T1, T2, T3를 사용한다

# 절차 규칙

메서드와 필드 이름

| 식별자 타입     | 예  |
|------------|--|
| 패키지와 모듈    | org.junit.jupiter.api, com.google.common.collect |
| 클래스와 인터페이스 | Stream, FutureTask, LinkedHashMap, HttpClient    |
| 메서드와 필드    | remove, groupingBy, getCrc                       |
| 상수 필드      | MIN_VALUE, NEGATIVE_INFINITY                     |
| 지역변수       | i, denom, houseNum                               |
| 타입 매개변수    | T, E, K, V, X, R, U, V, T1, T2                   |

## **문법 규칙**

**절차 규칙과 비교하면 더 유연하고 논란도 많다.**



# 문법 규칙

- 객체를 생성할 수 있는 클래스(열거 타입 포함)의 이름은 보통 **단수 명사나 명사구**를 사용한다  
(Thread, PriorityQueue, ChessPiece)
- 객체를 **생성할 수 없는** 클래스의 이름은 보통 **복수형 명사**로 짓는다.  
(Collectors, Collections 등)
- 인터페이스 이름은 클래스와 똑같이 짓거나(Collection, Comparator),  
**able** 혹은 **ible**로 끝나는 형용사로 짓는다(Runnable, Iterable, Accessible)
- 애너테이션은 그 활용이 **다양한 관계로 지배적인 규칙이 없이 명사, 동사, 전치사, 형용사가  
두루 쓰인다**

# 문법 규칙

- 어떤 동작을 수행하는 메서드의 이름은 목적어를 포함한 동사구로 짓는다  
(append, drawImage)
- boolean값을 반환하는 메서드라면 보통 is나 드물게 has로 시작하고  
명사나 명사구, 혹은 형용사로 기능하는 아무 단어나 구로 끝나도록 짓는다  
(isDigit, isEmpty, hasSiblings)
- 반환 타입이 boolean이 아니거나 해당 인스턴스의 속성을 반환하는 메서드의 이름은 보통 명사, 명사구, 혹은 get으로 시작하는 동사구로 짓는다.
  - ObjectMapper의 기본 설정으로는 public 필드 또는 public 형태의 getX로 시작하는 메소드 직렬화 처리

# 문법 규칙

- 객체 타입을 바꿔서 다른 타입의 또 다른 객체를 반환하는 인스턴스 메서드의 이름은 보통 **toType** 형태로 짓는다(toString, toArray)
- 객체의 내용을 다른 뷰로 보여주는 메서드의 이름은 **asType** 형태로 짓는다(asList)

## 주요 차이점 요약

| 메서드 명명 | toType (예: toString , toArray )                  | asType (예: asList )                          |
|--------|--|--|
| 목적     | 객체를 다른 타입으로 변환하여 새로운 객체를 반환                      | 객체의 <b>**다른 뷰(view)**</b> 를 반환               |
| 특징     | 새로운 객체를 생성하여 반환 (데이터 변환)                         | 원본 데이터를 다른 형식으로 참조 (데이터 변환 없음)               |
| 예시     | <code>toString()</code> , <code>toArray()</code> | <code>asList()</code> , <code>asMap()</code> |

- toType** : 객체 변환을 통해 새로운 메모리 공간을 사용하는 경우에 주로 사용됩니다.
- asType** : 데이터 표현 방식을 변경하여 원본 객체에 대한 다른 뷰를 제공하는 경우 사용됩니다.

# 문법 규칙

- 정적 팩터리의 이름은 다양하지만  
from, of, valueOf, instance, getInstance, newInstance, getType, newType을 흔히 사용한다

