정규표현식과 Pattern 클래스

❖ 정규 표현식(Regular Expression) 작성 방법

- 문자열이 정해져 있는 형식으로 구성되어 있는지 검증할 때 사용
  - Ex) 이메일, 전화번호, 비밀번호 등

❖ 정규 표현식(Regular Expression) 작성 방법

ㅇ 문자 또는 숫자 기호와 반복 기호가 결합된 문자열

기호	설명		
[ ]	한 개의 문자	[abc]	a, b, c 중 하나의 문자
		[^abc]	a, b, c 이외의 하나의 문자
		[a-zA-Z]	a~z, A~Z 중 하나의 문자
₩d	한 개의 숫자, [0-9]와 동일		
₩s	공백		
₩w	한 개의 알파벳 또는 한 개의 숫자, [a-zA-Z_0-9]와 동일		
?	없음 또는 한 개		
*	없음 또는 한 개 이상		
+	한 개 이상		
{n}	정확히 n 개		
{n,}	최소한 n 개		
{n, m}	n 개에서부터 m 개까지		
( )	그룹핑		

정규표현식과 Pattern 클래스

❖ Pattern 클래스

- ㅇ 정규 표현식으로 문자열을 검증하는 역할
  - ▪ 결과는 `boolean` 타입 !!!

```
boolean result = Pattern.matches("정규식", "입력된 문자열");
```

정규표현식과 Pattern 클래스

❖ 문자열 검증하기 : StringBuilderExample.java

```
import java.util.regex.Pattern;

public class PatternExample {
    public static void main(String[] args) {
        String regExp = "(02|010)-\\d{3,4}-\\d{4}";
        String data = "010-123-4567";
        boolean result = Pattern.matches(regExp, data);
        if (result) {
            System.out.println("정규식과 일치합니다.");
        } else {
            System.out.println("정규식과 일치하지 않습니다.");
        }

        regExp = "\\w+@\\w+\\.\\w+(\\.\\w+)?";
        data = "angel@navercom";
        result = Pattern.matches(regExp, data);
        if (result) {
            System.out.println("정규식과 일치합니다.");
        } else {
            System.out.println("정규식과 일치하지 않습니다.");
        }
    }
}
```