JAVA PROGRAMMING

정규 표현식

Regular Expression

정규 표현식(Regular Expression)이란?

- **특정한 규칙(패턴)을 가진 문자열의 집합을 표현하는데 사용하는 표현식 언어이다.**

- 정규 표현식은 주로 텍스트(문자열) 많이 처리하는 프로그래밍 언어(C, Perl, Java, JavaScript 등)에서 문자열의 검색과 치환을 위해 주로 사용
  - 특수기호나 문자(메타문자)를 이용하여 일정한 텍스트의 패턴 정의
  - 예) 특정 문자열 안에 hello라는 문자열이 들어 있는지, 혹은 특정 숫자가 들어 있는지 등의 검색

- 자바의 경우 JDK1.4부터 정규 표현식을 지원하기 위한 API 제공

정규 표현식 문법

- '.' 특수문자
  - 임의의 한 문자를 의미
  - '.'가 위치한 곳에는 반드시 임의의 한글자가 위치하여야 한다는 의미

패턴	일치하는 문자열
a.b	acb, azb 등
ab.	abc, abz 등
.bc	abc, zbc 등

- '*' 특수문자
  - 바로 앞의 문자가 없거나 하나 이상 반복한다는 의미

패턴	일치하는 문자열
Hello*	Hell, Hello, Helloo, Hellooo 등
Ab*c	Abc, Abbc, Abbbc 등
*d	표현 불가(*앞에는 반드시 한 글자 이상의 단어가 와야 한다)

정규 표현식 문법

- **'+' 특수문자**
  - '*'과 흡사하지만, '+'는 반드시 하나 이상의 문자가 반복한다는 의미

패턴	일치하는 문자열
Hello+	Hello, Helloo, Hellooo 등
Ab+c	Abc, Abbc, Abbbc 등
+d	표현 불가

- **'?' 특수문자**
  - '?' 바로 앞의 문자가 없거나, 하나임을 의미

패턴	일치하는 문자열
A?c	c, Ac 중에서 하나
Hello?	Hell, Hello 중에서 하나
Try?	Tr, Try 중에서 하나

정규 표현식 문법

- **'^' 특수문자**
  - 문장의 처음을 나타내며, '^'가 있는 단어로 문장이 시작됨을 의미

패턴	일치하는 문자열
^Hello	Hello World, Hello Java 등
^The	The Pen, The Book 등

- **'$' 특수문자**
  - 문장의 끝을 나타내며, '$'가 있는 단어로 문장이 끝남을 의미

패턴	일치하는 문자열
World$	Hello Java World
Java$	Start Java

정규 표현식 문법

▪ '[ ]' 특수문자(조건)

– 괄호 안의 문자 중 일치하는 것을 검색할 할 경우 사용

패턴	일치하는 문자열
[abc]	a, b, c, ab, abc 등(문자열에 'a', 'b', 'c' 등이 있어야 한다)
[a-z]	알파벳 소문자가 포함된 모든 문자열(범위)
[A-Z]	알파벳 대문자가 포함된 모든 문자열(범위)
[0-9]	숫자가 포함된 모든 문자열
^[a-zA-Z0-9]	영문대소문자 또는 숫자로 시작되는 모든 문자열 검색

▪ '[ ]' 안에서의 '^' 특수문자(부정)

– '[ ]' 특수문자 안에 있는 문자를 포함하고 있지 않는 모든 문자열을 찾고자 할 경우

패턴	일치하는 문자열
[^abc]de	dde, fde, zde 등
a[^0-9]c	abc, acc, adc 등

정규 표현식 문법

- **'{ }' 특수문자**
  - **'{ }' 특수문자 앞의 문자가 반복되는 횟수를 의미**

패턴	일치하는 문자열
Hel{2}o	Hello
Gu{5}ggle	Guuuuuggle
Gu{3,}ggle	Guuuggle, Guuuuggle, Guuuuuggle 등(3개 이상)
Gu{2,4}ggle	Guuggle, Guuuggle, Guuuuggle(2개이상 4개이하)

- **'( )' 특수문자**
  - **'( )' 안의 문자열을 하나의 문자로 취급**

패턴	일치하는 문자열
(Hello){3}	HelloHelloHello
(Hello)*	Hello, HelloHello 등
Gu(gg){2}le	Guggggle

정규 표현식 문법

- '|' 특수문자
  - OR 연산을 수행

패턴	일치하는 문자열
Hi|Hello	Hi 나 또는 Hello가 포함된 문자열
Man|Woman	Man, Woman, ManWoman, SuperMan 등

정규 표현식 문법

- **문자 클래스**
  - '[]' 특수문자 안에서 자주 사용되는 패턴들을 미리 키워드로 정의하여 놓은 문자
  - 해당 문자 클래스를 자바에서는 다르게 사용

패턴	문자클래스	자바 사용 예
[a-zA-Z] (모든영문자)	[[:alpha:]]	₩p{Alpha}
[0-9] (숫자)	[[:digit:]]	₩p{Digit}
[a-zA-Z0-9] (영문자와 숫자)	[[:alnum:]]	₩p{Alnum}
공백	[[:space:]]	₩p{Space}

- **특수 문자 사용**
  - 메타문자(., *, + 등)을 정규표현식의 패턴에서 사용하려면 해당 특수문자 앞에 '₩' 사용
  - '₩'를 패턴에서 사용하려면 ₩₩ 로 사용

정규 표현식 문법

▪ 기타 표현

패턴		설 명
\w	[a-zA-Z0-9]	알파벳이나 숫자
\W	[^a-zA-Z0-9]	알파벳이나 숫자를 제외한 문자
\d	[0-9]	숫자[0-9]와 동일
\D	[^0-9]	숫자를 제외한 모든 문자
^[0-9]*$	\p{Digit}	숫자만
^[a-zA-Z]*$	\p{Alpha}	영문자만
^[가-힣]*$		
^[a-zA-Z0-9]*$	\p{Alnum}	영어/숫자만
[[:space:]]	\p{Space}	공백

정규 표현식 문법

▪ 표현 예

구 분	패 턴
이메일	^[a-zA-Z0-9]+@[a-zA-Z0-9]+(.[₩w]+) ^[_0-9a-zA-Z-]+@[0-9a-zA-Z-]+(.[_0-9a-zA-Z]+)*$
휴대폰	^01(?:0|1|[6-9]) - (₩d{3,4}) - ₩d{4}$
일반전화	^₩d{2,3} - ₩d{3,4} - ₩d{4}&
주민등록번호	₩d{6} ₩- [1-4]₩d{6}
IP주소	([0-9]{1,3})₩.{([0-9]{1,3})₩.{([0-9]{1,3})₩.{([0-9]{1,3})
인터넷주소	([₩₩p{Alnum}]+)://([a-z0-9.₩₩-&%=?:@~₩₩_]+)"

자바 API를 이용한 정규 표현식 처리

- **java.util.regex 패키지**
- **Pattern 클래스**
  - **주어진 정규 표현식으로부터 패턴 생성**

static Pattern compile(String regex)	주어진 정규 표현식으로부터 패턴 생성
static Matcher matcher (CharSequence input)	패턴을 검색하는 Matcher 객체 반환
String pattern()	정규표현식을 String으로 반환
String split(CharSequence input)	패턴에 따라 분리

- **Matcher 클래스**
  - **문자열이 주어진 패턴과 일치하는지 확인**
  - **Matcher matcher = pattern.patcher("문자열");**

자바 API를 이용한 정규 표현식 처리

- **String 클래스의 정규 표현식 지원 메소드**

메소드	기 능
boolean matches(String regex)	문자열 안에 주어진 패턴이 존재 하는 지의 여부 반환
String replaceAll(String regex, String replace)	패턴과 일치하는 모든 부분을 replaceStr로 치환
String replaceFirst(String regex, String replaceStr)	패턴과 일치하는 첫 부분을 replaceStr로 치환
String[] splite(String regex)	패턴과 일치하는 구분자를 기준으로 분할하여 배열로 반환