SUMMARY

Java1

Java(CT)

1. 이식성이 높은 언어이다. -> JRE

    * byte code (JVM 구동명령어(java.exe)에 의해 JVM에서 해석되고 해당 운영체제에 맞게 기계어로 번역

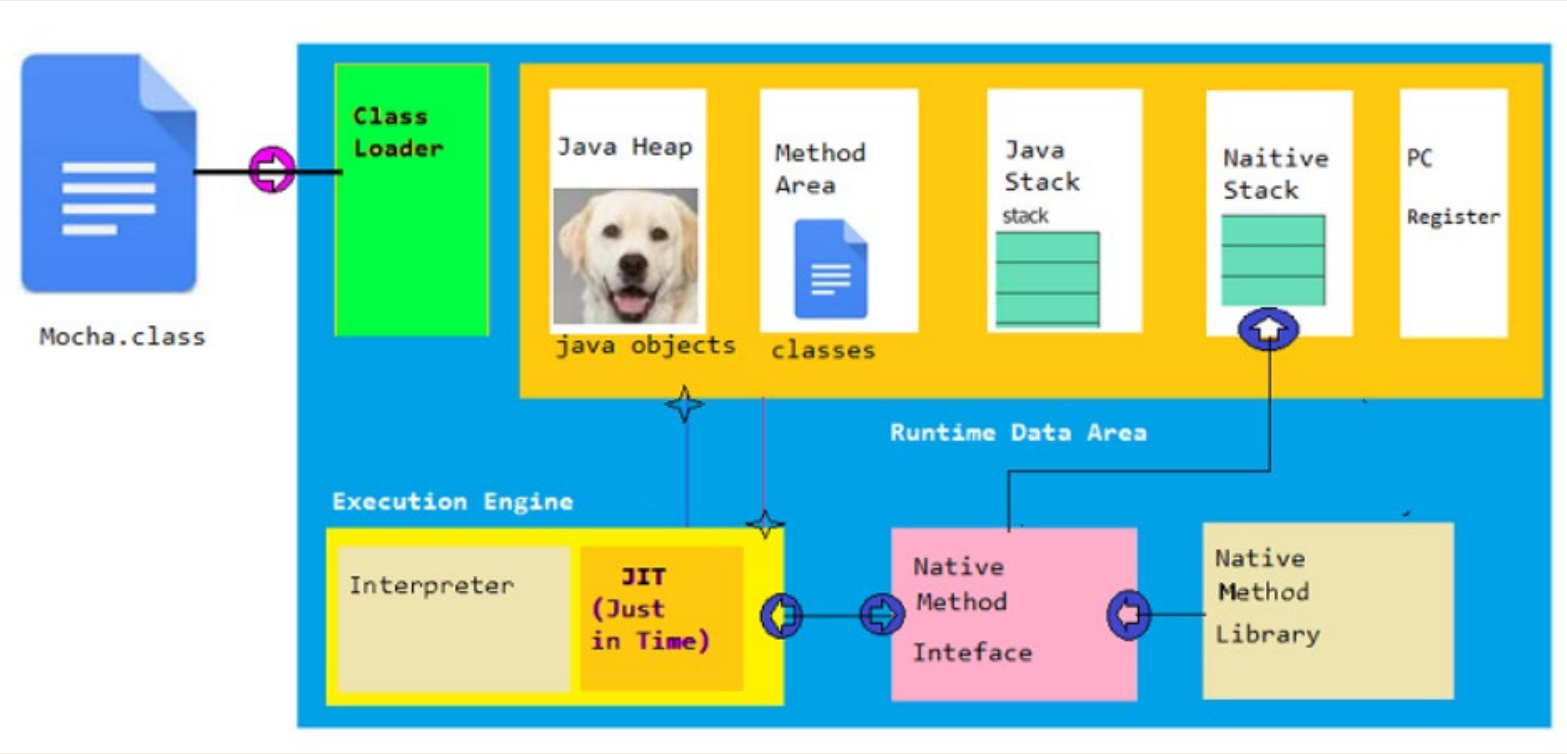
2. 객체 지향 언어이다. -> OOP -> 캡슐화(access modifiers), 상속, 다형성(field, parameter)
3. 함수적 스타일 코딩을 지원한다. -> lambda
4. 메모리를 자동으로 관리한다. -> garbage Collector
5. 다양한 애플리케이션을 개발할 수 있다. -> JVM(OS에 종속적), JRE, Java SE, Java EE
    - JRE = JVM + 표준 클래스 라이브러리(Java SE)
    - JDK = JRE + 개발에 필요한 도구
    - JVM 내부의 bin director는 compile인 javac.exe와 JVM구동명령어인 java.exe가 포함되어 있다.
6. 멀티 스레드를 쉽게 구현할 수 있다. -> thread
7. 동적로딩을 지원한다. -> reflection
8. 막강한 오픈소스 라이브러리가 풍부하다. -> API

JVM Architecture

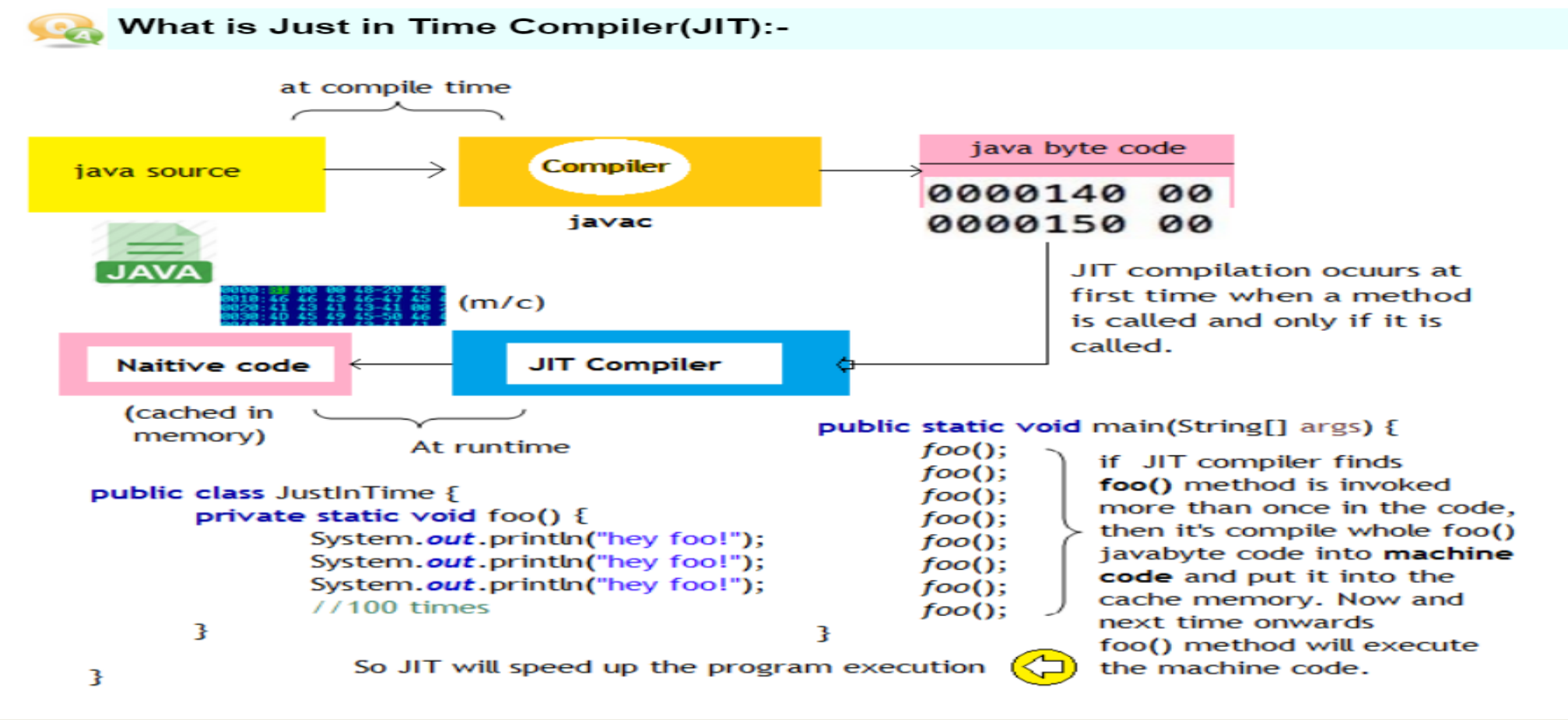
1. https://docs.google.com/document/d/1dsinB-qW-ChjL07msZeytYuqbS-G4h5doCMhztytH6s/edit?usp=sharing
2. http://www.gpsprogramys.co.in/java/jvm-architecture
3. https://hongsii.github.io/2018/12/20/jvm-memory-structure/
4. https://jdm.kr/blog/188

* 복습 과정 1, (2, 3, 4) -> 같은 내용이지만 빈약한 개념들 채워줌

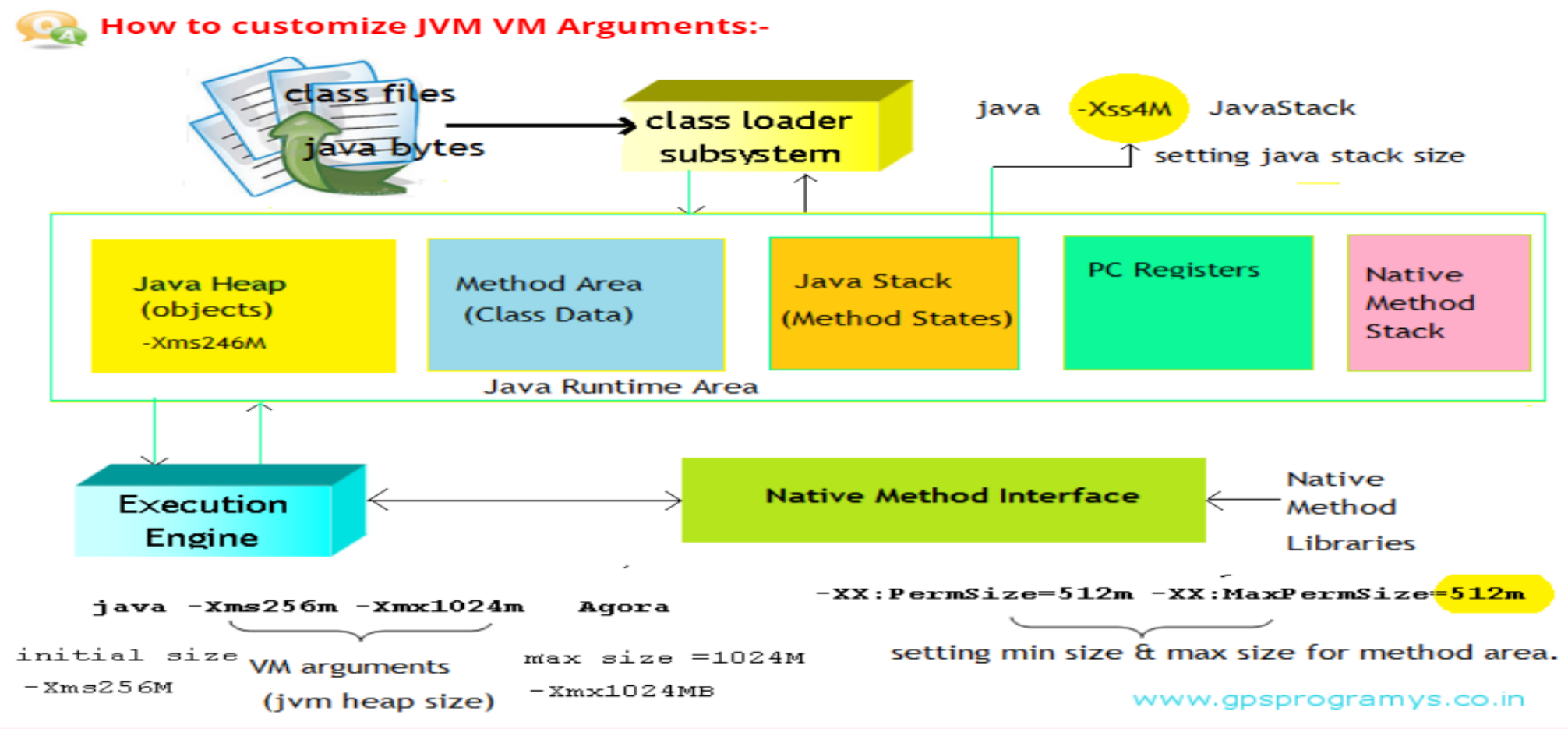
JVM Architecture



JVM Architecture



JVM Architecture



Reflection

- https://brunch.co.kr/@kd4/8 -> intro
- https://gyrfalcon.tistory.com/entry/Java-Reflection -> 사용법 * newInstance()
- https://kaspyx.tistory.com/80 -> 예제 * getMethod() vs getDeclaredMethods()

  - getMethods() , getFields() 등등을 사용하면 함수이나 변수이름을 몰라도 불러올 수있으며, 이 함수는 public으로 선언한 메소드(함수)나 필드(변수)를 가져온다.

  - getMethod()는 문자열을 가지고 단일 함수를 찾아주지만 getDeclaredMethods()는 private 포함 Class 내부에서 선언한 함수만 찾아준다.

- https://jdm.kr/blog/68 -> 예제
- http://www.nextree.co.kr/p3643/ -> 사례 (객체 처리)
- http://tutorials.jenkov.com/java-reflection/index.html -> tutorial

Annotation

- https://kang594.blog.me/39704853 -> Intro
- http://tutorials.jenkov.com/java-reflection/annotations.html -> 사용법
- https://m.blog.naver.com/PostView.nhn?blogId=javaking75&logNo=220727816394&proxyReferer=https%3A%2F%2Fwww.google.com%2F -> 예제
- http://www.nextree.co.kr/p5864/ -> 사례

lombok

- https://goddaehee.tistory.com/95 -> intro

getProperty() 시스템 프로퍼티 읽기
getenv() 환경변수 읽기

1. System.getProperty("xxx")

https://hoonihoon.tistory.com/entry/Java-SystemgetProperty

- JVM내 자바 속성 값을 가져오는 함수.

2. System.getenv("xxx")

http://mwultong.blogspot.com/2007/01/java-os-get-print-environment-variable.html

- OS내 환경 변수를 가져오는 함수.

https://www.baeldung.com/java-system-get-property-vs-system-getenv

Exception

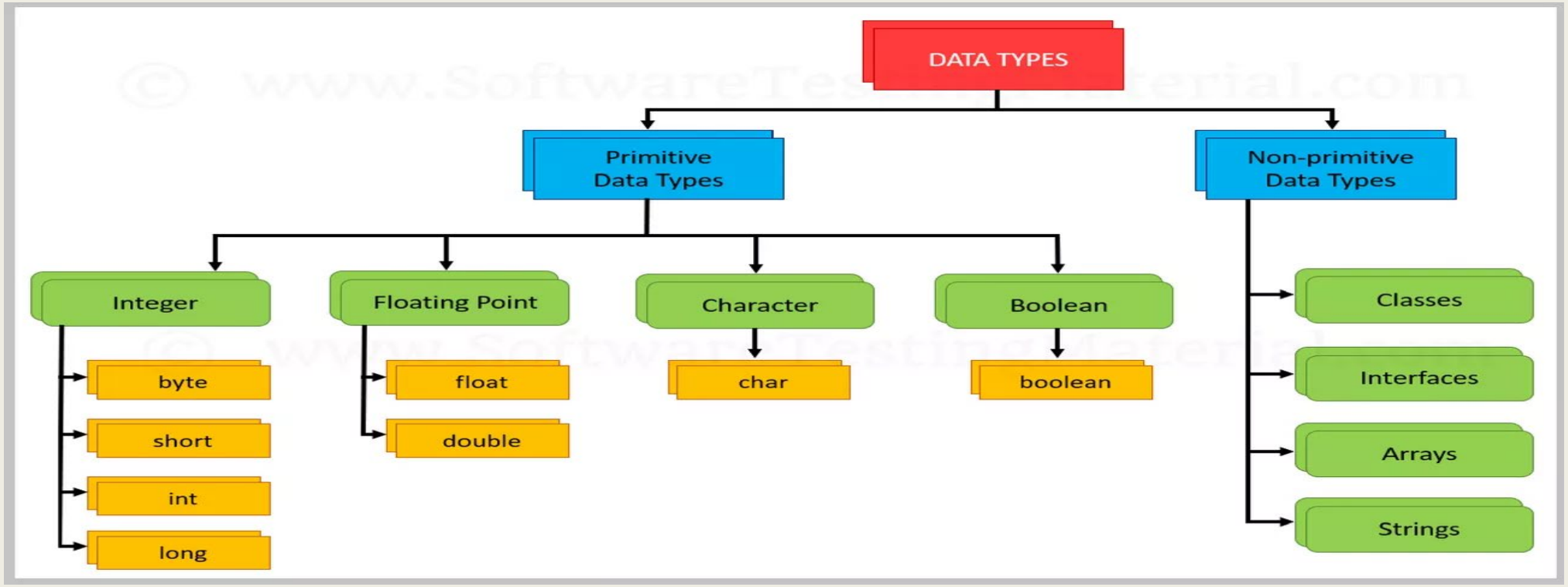
```
static void method(boolean b) {
    try {
        System.out.println(1);
        if (b) throw new ArithmeticException();
        System.out.println(2);
    } catch (RuntimeException r) {
        System.out.println(3);
        return;
    } catch (Exception e) {
        System.out.println(4);
        return;
    } finally {
        System.out.println(5);
    }
    System.out.println(6);
}

public static void main(String[] args) {
    method(true);
    method(false);
}
```

<term

```
1
3
5
1
2
5
6
```

Data Type


DATA TYPES
Primitive Data Types
Non-primitive Data Types
Integer
Floating Point
Character
Boolean
byte
short
int
long
float
double
char
boolean
Classes
Interfaces
Arrays
Strings

Naming convention

■ Camel Case

- ○ Second and subsequent words are capitalized, to make word boundaries easier to see. (Presumably, it struck someone at some point that the capital letters strewn throughout the variable name vaguely resemble camel humps.)
  - ➢ Example: numberOfCollegeGraduates

■ Pascal Case

- ○ Identical to Camel Case, except the first word is also capitalized.
  - ➢ Example: NumberOfCollegeGraduates

■ Snake Case

- ○ Words are separated by underscores.
  - ➢ Example: number_of_college_graduates

예약어

분류	예약어
기본 데이터 타입	boolean, byte, char, short, int, long, float, double
접근 지정자	private, protected, public
클래스와 관련된 것	class, abstracted, interface, extends, implements, enum
객체와 관련된 것	new, instanceof, this, super, null
메소드와 관련된 것	void, return
제어문과 관련된 것	if, else, switch, case, default, for, do, while, break, continue
논리값	true, false
예외 처리와 관련된 것	try, catch, finally, throw, throws
기타	transient, volatile, package, import, synchronized, native, final, static, strictfp, assert

Operator (float, double)

- Promotion, casting
- 부동소수점 타입(float, double)은 0.1을 정확하게 표현할 수 없다.
- 0.1f != 0.1d
- 지수, 가수 범위 숙지
- 8bit , 23bit (float) / 11bit, 52bit (double)

switch

- switch -> 정수 타입(byte, char, short, int, long), String(Java7)

arraycopy

- shallow copy
- Deep copy
- https://m.blog.naver.com/pgh7092/221076215577

clone

- Shallow clone
- Deep clone

1. https://taetaetae.github.io/2018/08/21/how-to-use-cloneUtils/

-> Instance value is object in class(Mutable vs Immutable)

2. https://library1008.tistory.com/47

-> container(List, Set, Map)

Java Access Modifiers

Most Restrictive ← → Least Restrictive

Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

Same rules apply for inner classes too, they are also treated as outer class properties

Java Access Modifiers

1. Class -> default, public [static* I final]

    * 바깥의 class는 static class가 될 수 없다. ==> (틀린 말이다. static 키워드를 단지 안 붙일 뿐이다.)

    * 바깥의 class는 static 클래스라서 static을 붙이면 에러가 난다! 즉 static을 붙일 필요가 없어서 안 붙이는 것이지 static 클래스가 아니거나 불가능해서 static을 안 붙이는 것이 아니다.

    * So... Nested Class???? -> https://m.blog.naver.com/iq_up/220013622883

2. Construct -> public, protected, default, private

    -> why not use the static or final?

3. Field -> public, protected, private [static I final]

4. Method -> public, protected, private [static I final]

Construct is method???

https://javacan.tistory.com/entry/37

->A: No

* this(), super() -> Instance의 예약어

리펙토링

- https://blog.asamaru.net/2012/09/24/dont-afraid-code-refactoring/ ->intro

객체지향 설계

- http://woowabros.github.io/study/2016/07/07/think_object_oriented.html

Abstract vs Interface

- https://postitforhooney.tistory.com/entry/Java-Interface%EC%99%80-Abstract-class%EC%9D%98-%EC%B0%A8%EC%9D%B4%EC%99%80-%EA%B0%81%EA%B0%81%EC%9D%98-%ED%8A%B9%EC%A7%95-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0

Nested class

1. https://terranboy1.github.io/2019/01/12/0-3-java-innerclass/
2. https://doublesprogramming.tistory.com/158

   -> SUMMARY

- Using -> GUI , AWT , Swing
- 개념만 알자 (key point: Instance member, class member, static , final)

singleton

- http://blog.naver.com/PostView.nhn?blogId=heartflow89&logNo=221001179016&parentCategoryNo=&categoryNo=&viewDate=&isShowPopularPosts=false&from=postView -> intro
- https://blog.seotory.com/post/2016/03/java-singleton-pattern -> 예제
  *enum class
- http://woowabros.github.io/tools/2017/07/10/java-enum-uses.html -> 사례

Java.utill

- https://m.blog.naver.com/PostView.nhn?blogId=mals93&logNo=220722541094&proxyReferer=https%3A%2F%2Fwww.google.com%2F

Comparable vs Comparator

- https://gmlwjd9405.github.io/2018/09/06/java-comparable-and-comparator.html -> Intro
- https://cwondev.tistory.com/15 -> 예제
- Comparable - 기본 정렬기준을 구현하는데 사용.
- Comparator - 기본 정렬기준 외에 다른 기준으로 정렬 하고자 할 때 사용.

결론

- Comparable 구현 후 내부의 compareTo 메소드를 오버라이드해 정의해야 하는데, 이 정의 결과에 따라 정렬 값이 나온다.
- 또한, 오브젝트의 다른 값으로 비교를 원한다면 compareTo를 하나하나 바꿔줄 필요 없이, Comparator를 이용하면 된다.

- Collections.sort() , Arrays.sort() 등 ~~.sort()는 배열이나 리스트를 정렬할 때 Comparator를 지정하지 않았을 경우
- Comprarable을 구현한 클래스의 객체에 구현된 내용에 따라 정렬!!

Equlas() vs deepEquals()

- Object.equals(Object a, Object b)는 두 객체의 동등을 비교하는데 다음과 같은 결과를 리턴합니다. 특이한 점은 a와 b가 모두 null일 경우 true를 리턴 한다는 점 입니다.
  A 와 b 가 null이 아닌 경우에는 a.equals(b)의 결과를 리턴 합니다.

a	b	Objects.equals(a,b)
not null	not null	a.equals(b) 의 리턴값
null	not null	false
not null	null	false
null	null	true

Equlas() vs deepEquals()

- Object.deepEqua니(Object a, Object b)역시 두 객체의 동등을 비교하는데, a와 b가 서로 다른 배열일 경우, 항목 값까지 모두 같다면 true를 리턴 합니다. 이것은 Arrays.deepEquals(Object[] a , Object[] b)와 동일합니다.

a	b	Objects.deepEquals(a,b)
not null (not array)	not null (not array)	a.equals(b) 의 리턴값
not null (array)	not null (array)	Array.deepEquals(a,b)
not null	null	false
null	not null	false
null	null	true

Equlas() vs deepEquals()

- 객체이면?
- deepEquals는 객체를 하나 씩 비교를 하므로 equal을 오버라딩하여 비교 함수를 재 정의 하여 만들면 된다.

Equlas() vs deepEquals()

```java
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        String  a= "dfd";
        String b = "dfd";
//      String d = "fff";
        String c = new String("dfd");
        String[] df = new String[]{a, a, a};
        String[] dd = new String[] {c, c, c};
        System.out.println(Arrays.deepEquals(df, dd));

        Test t = new Test();
        test tll = t.new test();
        test tll3 = t.new test();
        tll.a = 10;
        tll3.a = 10;
        test hj[] = {tll, tll};
        test hh[] = {tll3, tll3};
        System.out.println(Arrays.deepEquals(hj, hh));
        System.out.println(hh[0]);

    }


class test{
    int a;

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return String.valueOf(a);
    }

    @Override
    public boolean equals(Object obj) {
        // TODO Auto-generated method stub
        if(obj instanceof test) {
            test a = (test)obj;
            if(a.a == this.a)return true;
        }
        return false;
    }
}
```

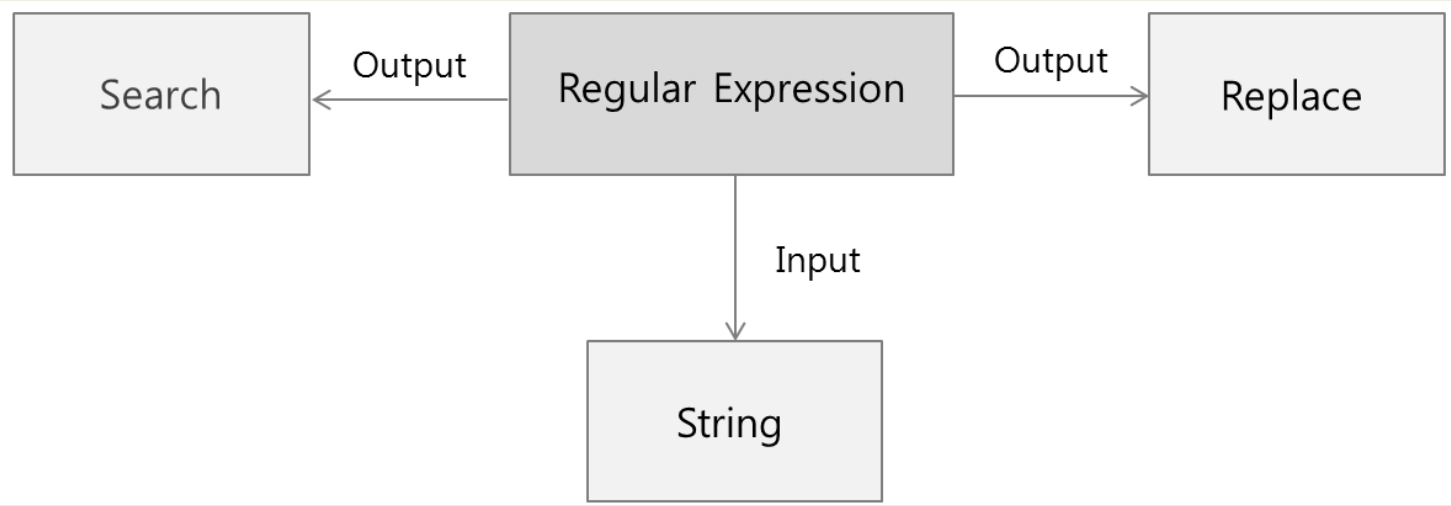
```
<terminate
true
true
10
```

isNull() vs nonNull() vs requireNonNull()

- Objects.isNull(Object obj)는 매개값이 null 일 경우 true를 리턴 한다.
- 반대로 nonNull(Object obj)는 매개값이 not null 일 경우 true를 리턴 합니다.
- 그리고 requireNonNull()는 다음 세 가지로 오버로딩 되어 있습니다.
- 세 가지 오버로딩 모두 첫 번째 매개값이 not null 이면 첫 번째 매개값을 리턴하고,
  null 이면 모두 NullPointerException을 발생시킵니다.
  두 번째 매개값은 NullPointerException의 예외 메시지를 제공합니다.

리턴 타입	메소드(매개 변수)	설명
T	requireNonNull(T obj)	not null -> obj null -> NullPointerException
T	requrieNonNull(T obj, String message)	not null -> obj null ->NullPointerException(message)
T	requrieNonNull(T obj, Supplier<String>msgSupplier)	not null -> obj null – NullPointerException(msgSupplier.get())

- **What is RE?**
  - *a special text string for describing a search pattern*
  - *searching, replacing, and parsing text with complex patterns of characters*



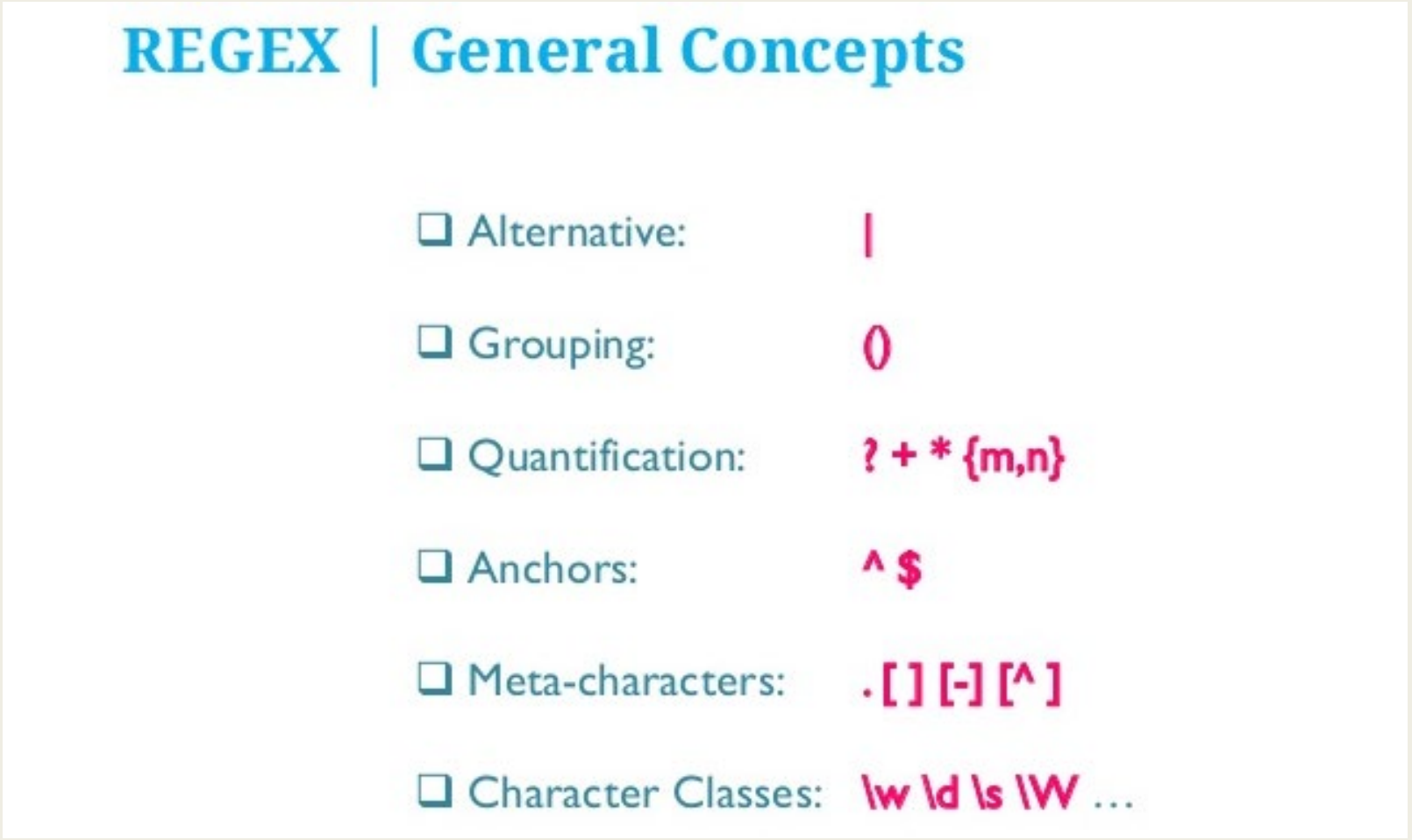
■ **Why use RE?**

○ *Processes large amounts of text over and over again / Extremely fast*

○ *Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation*

■ **But, There is a learning curve**

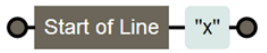
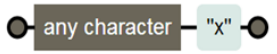
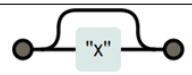
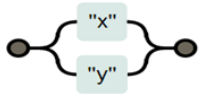
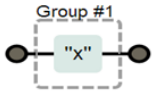
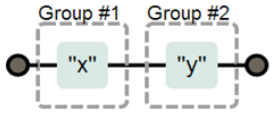
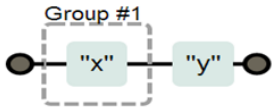
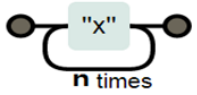
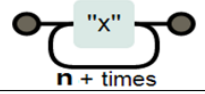
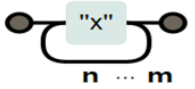
■ **General concepts**



■ Meta characters

○ . ^ $ * + ? { } [ ] \ | ( )

- .(dot)
  - *a.b*
  - *a+모든문자+b*
  - *aab, a0b, ~~abc~~*
- *(asterisk)
  - *ab*c*
  - *a+b(0번 이상 반복)+c*
  - *ac, abc, abbbbbbbc*
- +(plus)
  - *ab+c*
  - *a+b(1번 이상 반복)+c*
  - *~~ac~~, abc, abbbbbbbc*
- {n | min, | min, max}(curly braces)
  - *{0,}=*, {1,}=+*
  - *ab{1}c, ab{2,6}c*
  - *a+b(1번 반복, 2번-6번)+c*
  - *abc, abbc, abbbbbbc*
- ?(question)
  - *ab?c, b{0,1}*
  - *a+b(1개 있거나, 없거나)+c*
  - *ac, abc, ~~abbbc~~*

정규표현식	표현	설명
^x		문자열이 x로 시작합니다.
x$		문자열이 x로 끝납니다.
.x		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
x+		x가 1번이상 반복합니다.
x?		x가 존재하거나 존재하지 않습니다.
x*		x가 0번이상 반복합니다.
x|y		x 또는 y를 찾습니다. (or연산자를 의미합니다.)
(x)		( )안의 내용을 캡쳐하며, 그룹화 합니다.
(x)(y)		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡쳐합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
(x)(?:y)		캡쳐하지 않는 그룹을 생성할 경우 ?:를 사용합니다. 결과값 배열에 캡쳐하지 않는 그룹은 들어가지 않습니다.
x{n}		x를 n번 반복한 문자를 찾습니다.
x{n,}		x를 n번이상 반복한 문자를 찾습니다.
x{n,m}		x를 n번이상 m번이하 반복한 문자를 찾습니다.

■ Character classes

○ *[], [-], [^], \d, \D, \s, \S, \w, \W*

- ➢ [](square braket)
  - *[abc]*
  - *a, b or c*
- ➢ -
  - *[a-z]*
  - *a부터 z까지*
- ➢ ^
  - *[^a-z]*
  - *a부터 z까지를 제외*

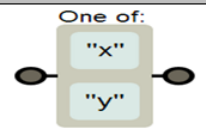
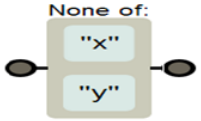
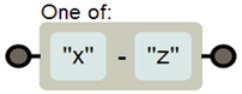
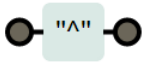
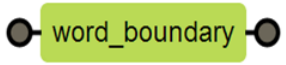
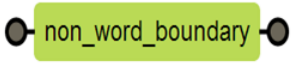
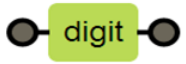
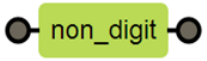
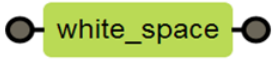
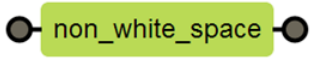
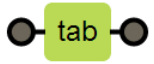
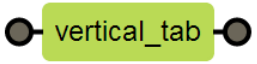
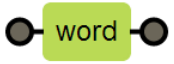
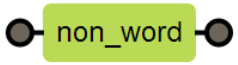
■ Anchors

○ *^, $, \b, \B*

- ➢ ^ - Starting position
- ➢ $ - Ending position

■ Grouping

- ➢ ()(parenthesis)
  - *(abc)*

정규표현식	표현	설명
[xy]		x,y중 하나를 찾습니다.
[^xy]		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
[x-z]		x~z 사이의 문자중 하나를 찾습니다.
₩^		^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
₩b		문자와 공백사이의 문자를 찾습니다.
₩B		문자와 공백사이가 아닌 값을 찾습니다.
₩d		숫자를 찾습니다.
₩D		숫자가 아닌 값을 찾습니다.
₩s		공백문자를 찾습니다.
₩S		공백이 아닌 문자를 찾습니다.
₩t		Tab 문자를 찾습니다.
₩v		Vertical Tab 문자를 찾습니다.
₩w		알파벳 + 숫자 + _ 를 찾습니다.
₩W		알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

■ Match method

- ○ *Greedy vs Lazy(non greedy)*
  - ➢ Greedy – tries to find the last possible match
  - ➢ Lazy – tries to find the first possible match

Greedy quantifier	Lazy quantifier	Description
*	*?	Match zero or more times.
+	+?	Match one or more times.
?	??	Match zero or one time.
{n}	{n}?	Match exactly n times.
{n,}	{n,}?	Match at least n times.
{n,m}	{n,m}?	Match from n to m times.

> Add a ? to a quantifier to make it ungreedy i.e lazy.

Example:
test string : *stackoverflow*
greedy reg expression : `s.*o` output: **stackoverflo**w
lazy reg expression : `s.*?o` output: **stacko**verflow

■ 예제

○ *https://www.regexr.com/*

○ *https://www.regexper.com/*

○ *Alternative*

- cat|mat → ‘car’ or ‘mat’
- regular|expression → ‘regular’ or ‘expression’

○ *Grouping*

- gr(e|a)y → ‘grey’ or ‘gray’
- py(pi|tho(n|nic) → ‘pypi’ or ‘python’ or ‘pythonic’

○ *Quantification*

- colou?r → ‘color’ or ‘colour’
- 81*5 → ‘85’ or ‘815’ or ‘8111115’
- 81+5 → ‘815’ or ‘8111115’
- go{2,3}gle → ‘google’ or ‘gooogle’
- go{2,} → ‘goo’ or ‘gooooooooooooo’

○ *Anchors – match the starting or ending position*

- ^obje → ‘object’ or ‘object-oriented’
- ^2018 → ‘2018’ or ‘2018-07-10’
- gram$ → ‘program’ or ‘kilogram’

○ *Meta characters – match a single character*

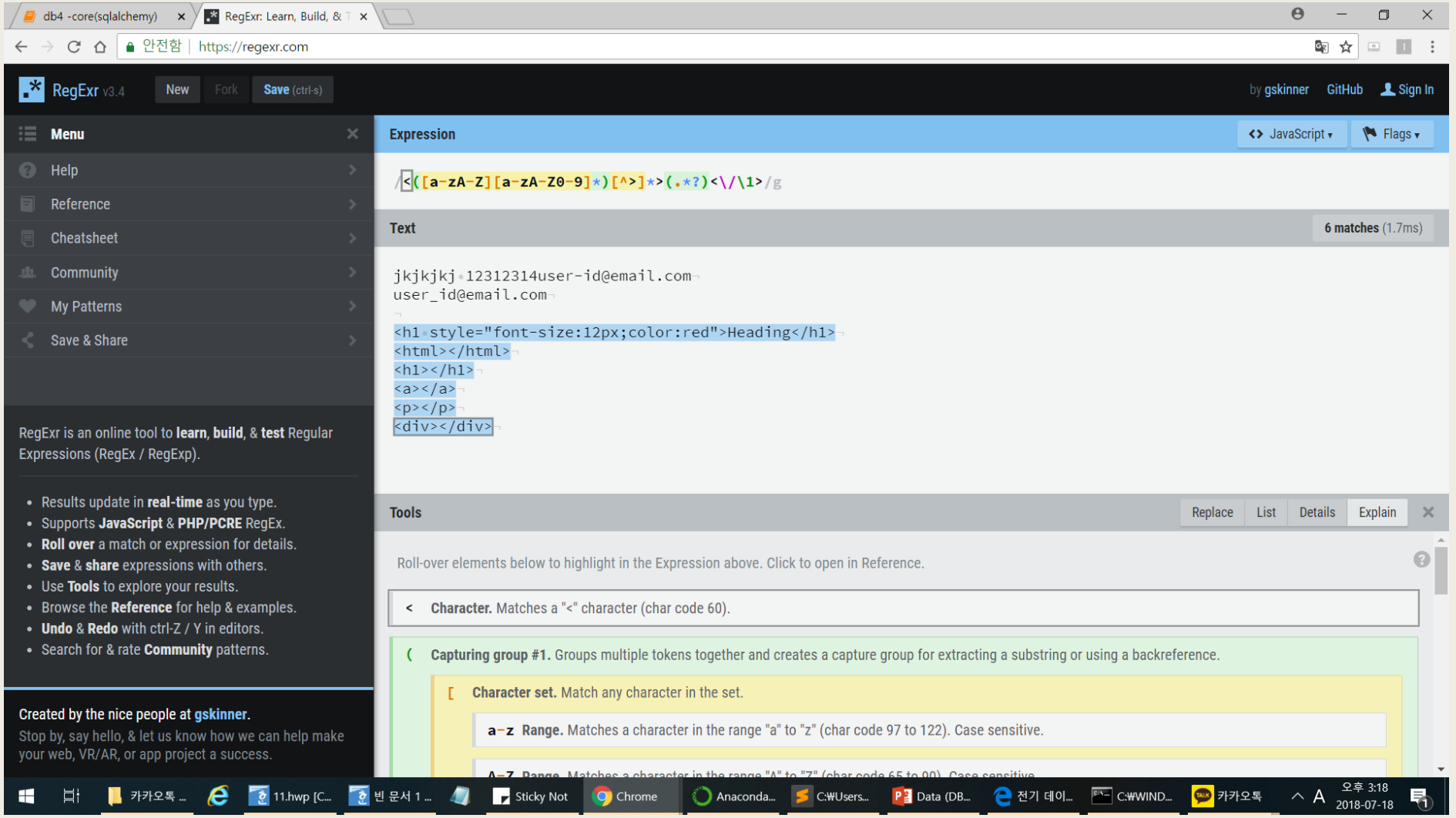
- bat. → ‘bat’ or ‘bats’ or ‘bata’
- [xyz] → ‘x’ or ‘y’ or ‘z’
- [aeiou] → any vowel
- [0123456789] → any digit
- [a-c] → ‘a’ or ‘b’ or ‘c’
- [a-zA-Z] → all letters(uppercase, lowercase)
- [0-9] → all digits
- [^aeiou] → any non-vowel
- [^0-9] → any non digit
- [^xyz] → any character, but not ‘x’ or ‘y’ or ‘z’

○ *Character classes*

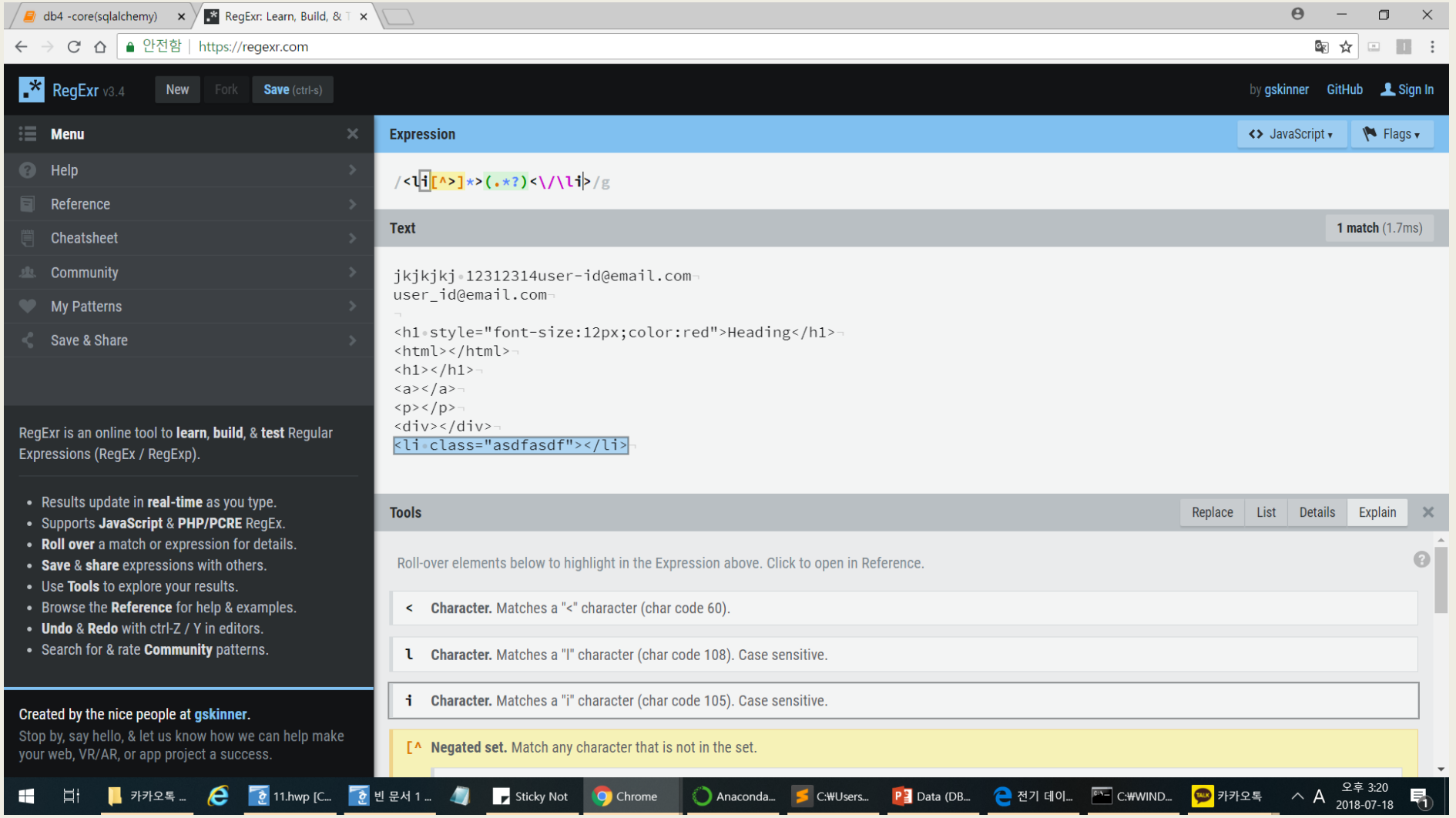
- \d – digits / \D – non digits
- \s – a single white space character / \S – non white space
- \w – alphanumeric character [a-ZA-Z0-9_] / \W – non alphanumeric
- \b – word boundaries / \B – non word boundaries

- *'Hello World'*
  - (H . .) . (o . .)
    - *'Hell', 'o W'*
  - l+
    - *He'll'o, Wor'l'd*
  - H.?e
    - *'He'*
  - l.+?o
    - *'llo'*
  - el*o
    - *'ello'*
  - l{1,2}
    - *He'l'lo, He'll'o, Wor'l'd*
  - [aeiou]+
    - *'e', 'o', 'o'*
  - (Hello|Hi|Pogo)
    - *'Hello'*
  - llo\b
    - *'llo'*
  - \w
    - *'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd'*
  - \W
    - ' '

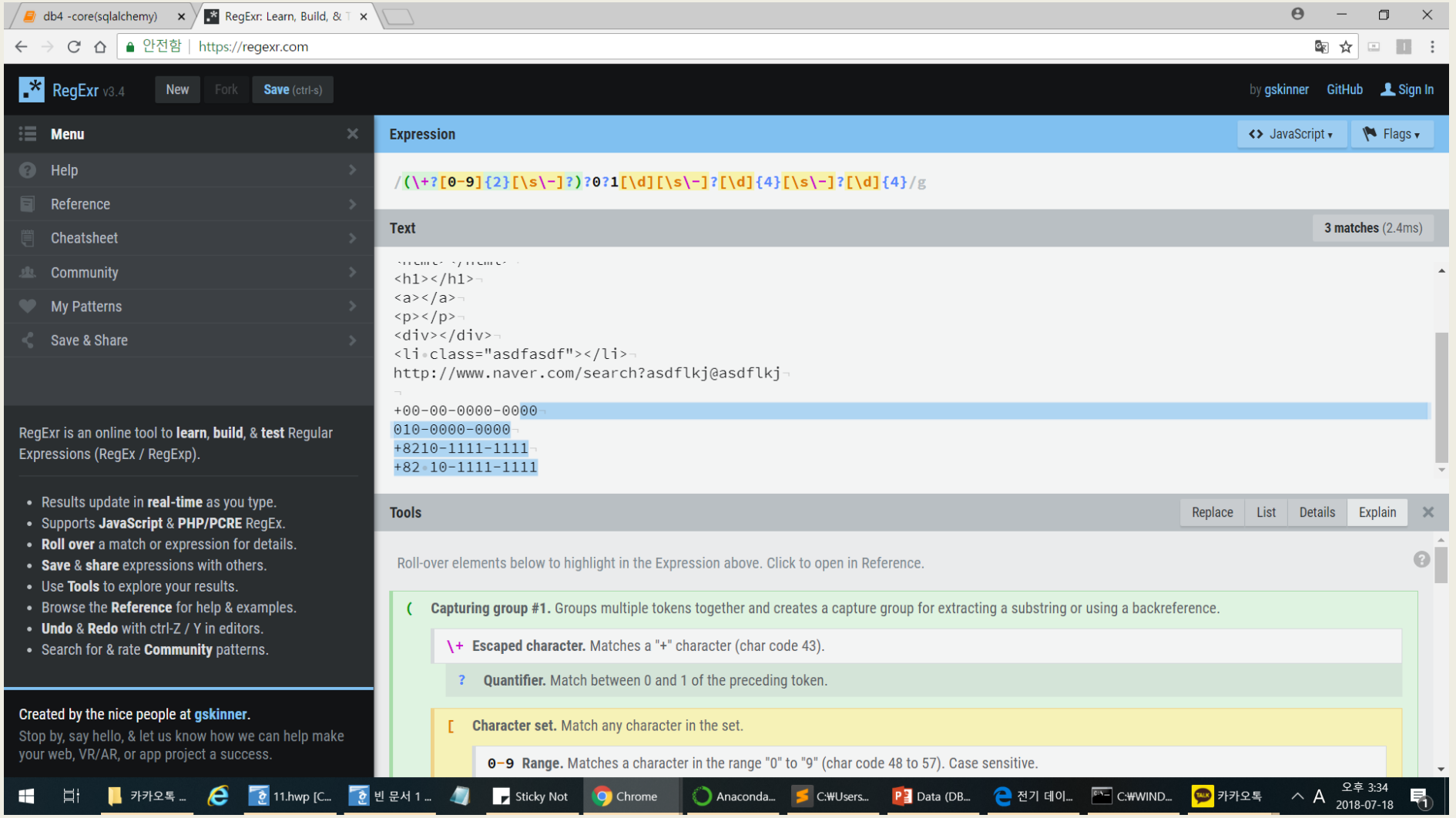
- 'In Hello World'
  - \s.*\s
    - ' Hello '
- 'Hello World'
  - \S.*\S
    - 'Hello World'
  - ^He
    - 'He'
  - rld$
    - 'rld'
  - el*o
    - 'ello'
  - [^Hlo]
    - 'e', ' ', 'W', r', 'd'


db4 -core(sqlalchemy)
RegExr: Learn, Build, &
안전함 | https://regexr.com
RegExr v3.4
New
Fork
Save (ctrl-s)
by gskinner
GitHub
Sign In
Menu
Help
Reference
Cheatsheet
Community
My Patterns
Save & Share
RegExr is an online tool to learn, build, & test Regular Expressions (RegEx / RegExp).
Results update in real-time as you type.
Supports JavaScript & PHP/PCRE RegEx.
Roll over a match or expression for details.
Save & share expressions with others.
Use Tools to explore your results.
Browse the Reference for help & examples.
Undo & Redo with ctrl-Z / Y in editors.
Search for & rate Community patterns.
Created by the nice people at gskinner.
Stop by, say hello, & let us know how we can help make your web, VR/AR, or app project a success.
Expression
JavaScript
Flags
/<([a-zA-Z][a-zA-Z0-9]*)[^>]*>(.*?)<\/\1>/g
Text
6 matches (1.7ms)
jkjkjkj 12312314user-id@email.com
user_id@email.com
<h1 style="font-size:12px;color:red">Heading</h1>
<html></html>
<h1></h1>
<a>

Tools
Replace
List
Details
Explain
Roll-over elements below to highlight in the Expression above. Click to open in Reference.
< Character. Matches a "<" character (char code 60).
(Capturing group #1. Groups multiple tokens together and creates a capture group for extracting a substring or using a backreference.
[Character set. Match any character in the set.
a-z Range. Matches a character in the range "a" to "z" (char code 97 to 122). Case sensitive.
카카오톡 ...
11.hwp [C...
빈 문서 1 ...
Sticky Not
Chrome
Anaconda...
C:₩Users...
Data (DB...
전기 데이...
C:₩WIND...
카카오톡
오후 3:18
2018-07-18


db4 -core(sqlalchemy)
RegExr: Learn, Build, &
안전함 | https://regexr.com
RegExr v3.4
New
Fork
Save (ctrl-s)
by gskinner
GitHub
Sign In
Menu
Help
Reference
Cheatsheet
Community
My Patterns
Save & Share
RegExr is an online tool to learn, build, & test Regular Expressions (RegEx / RegExp).
Results update in real-time as you type.
Supports JavaScript & PHP/PCRE RegEx.
Roll over a match or expression for details.
Save & share expressions with others.
Use Tools to explore your results.
Browse the Reference for help & examples.
Undo & Redo with ctrl-Z / Y in editors.
Search for & rate Community patterns.
Created by the nice people at gskinner.
Stop by, say hello, & let us know how we can help make your web, VR/AR, or app project a success.
Expression
JavaScript
Flags
/<li[^>]*>(.*?)<\/li>/g
Text
1 match (1.7ms)
jkjkjkj 12312314user-id@email.com
user_id@email.com
<h1 style="font-size:12px;color:red">Heading</h1>
<html></html>
<h1></h1>
<a>

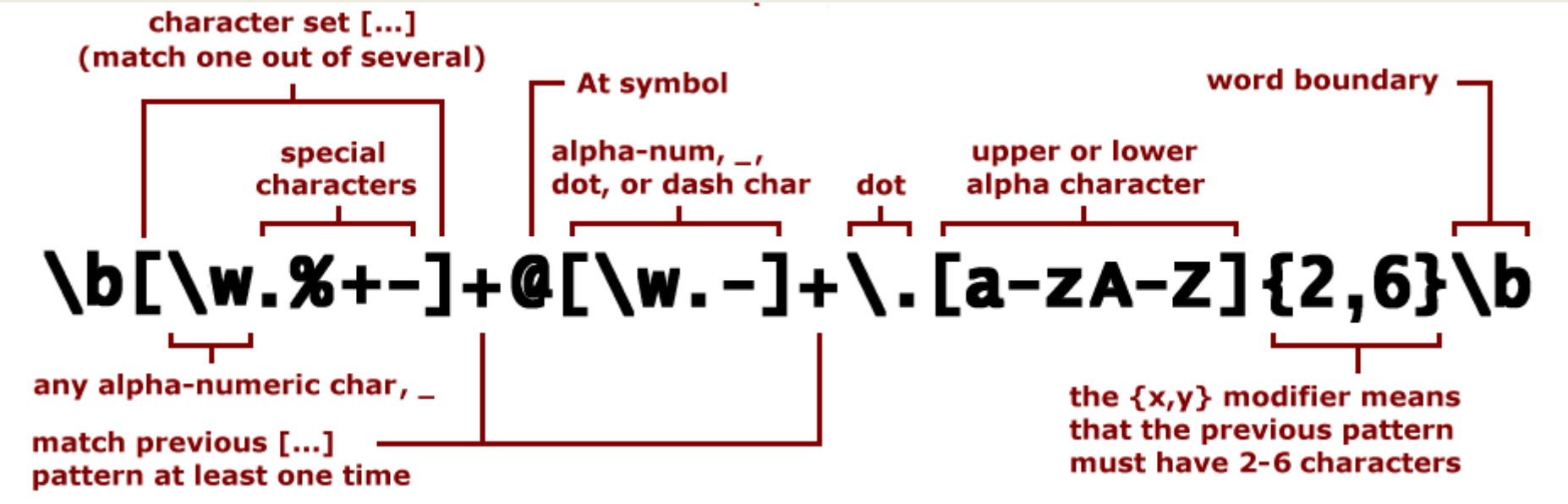
<li class="asdfasdf">
Tools
Replace
List
Details
Explain
Roll-over elements below to highlight in the Expression above. Click to open in Reference.
< Character. Matches a "<" character (char code 60).
l Character. Matches a "l" character (char code 108). Case sensitive.
i Character. Matches a "i" character (char code 105). Case sensitive.
[^ Negated set. Match any character that is not in the set.
카카오톡 ...
11.hwp [C...
빈 문서 1 ...
Sticky Not
Chrome
Anaconda...
C:₩Users...
Data (DB...
전기 데이...
C:₩WIND...
카카오톡
오후 3:20
2018-07-18


db4 -core(sqlalchemy)
RegExr: Learn, Build, &
https://regexr.com
RegExr v3.4
New
Fork
Save (ctrl-s)
by gskinner
GitHub
Sign In
Menu
Help
Reference
Cheatsheet
Community
My Patterns
Save & Share
RegExr is an online tool to learn, build, & test Regular Expressions (RegEx / RegExp).
Results update in real-time as you type.
Supports JavaScript & PHP/PCRE RegEx.
Roll over a match or expression for details.
Save & share expressions with others.
Use Tools to explore your results.
Browse the Reference for help & examples.
Undo & Redo with ctrl-Z / Y in editors.
Search for & rate Community patterns.
Created by the nice people at gskinner.
Stop by, say hello, & let us know how we can help make your web, VR/AR, or app project a success.
Expression
JavaScript
Flags
/(\+?[0-9]{2}[\s\-]?)?0?1[\d][\s\-]?[\d]{4}[\s\-]?[\d]{4}/g
Text
3 matches (2.4ms)
<h1></h1>
<a>

<li class="asdfasdf">
http://www.naver.com/search?asdflkj@asdflkj
+00-00-0000-0000
010-0000-0000
+8210-1111-1111
+82 10-1111-1111
Tools
Replace
List
Details
Explain
Roll-over elements below to highlight in the Expression above. Click to open in Reference.
(Capturing group #1. Groups multiple tokens together and creates a capture group for extracting a substring or using a backreference.
\+ Escaped character. Matches a "+" character (char code 43).
? Quantifier. Match between 0 and 1 of the preceding token.
[Character set. Match any character in the set.
0-9 Range. Matches a character in the range "0" to "9" (char code 48 to 57). Case sensitive.
Sticky Not
Chrome
Anaconda...
Data (DB...
오후 3:34
2018-07-18

예제

- username@domain.com



예제

<h1 style="color:red">Heading</h1>

<([A-Z][A-Z0-9]*)[^>]*>(.*?)<\/\1>

+00-00-0000-0000

+00 00 0000 0000

000-0000-0000

000 0000 0000

([\+]?[0-9]{2}[\s\-]?)?0?1[0-9]{1}[\s\-]?[0-9]{4}[\s\-]?[0-9]{4}