JAVA 기초입문과정

CHAPTER11 JAVA.base 모듈

Contents

API 도큐먼트

JAVA.base 모델

리플렉션

어노테이션


PART
01
“
API 도큐먼트
”

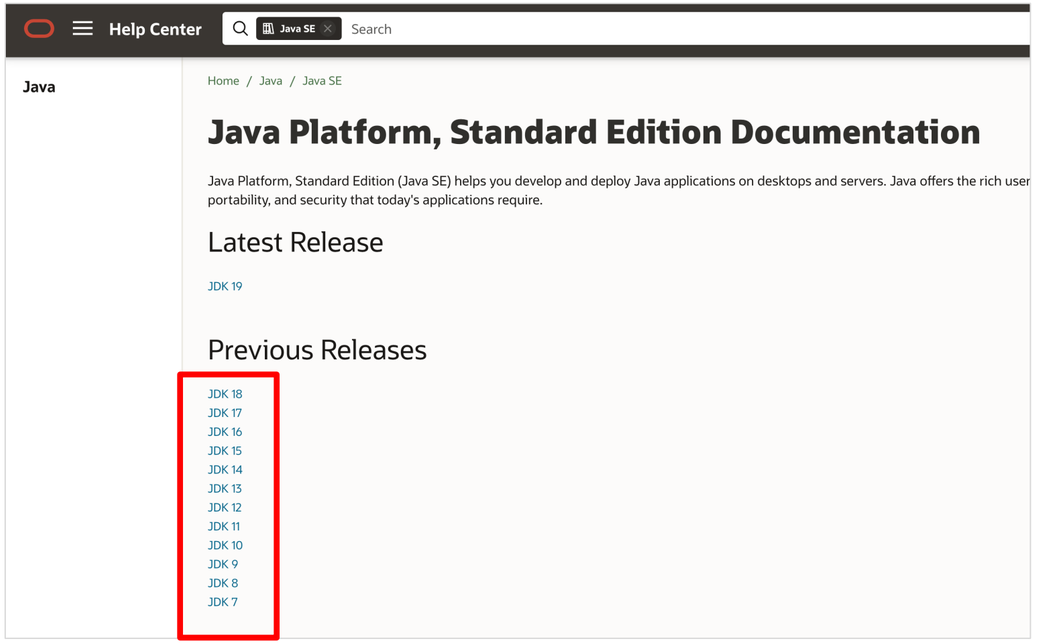
PART 01

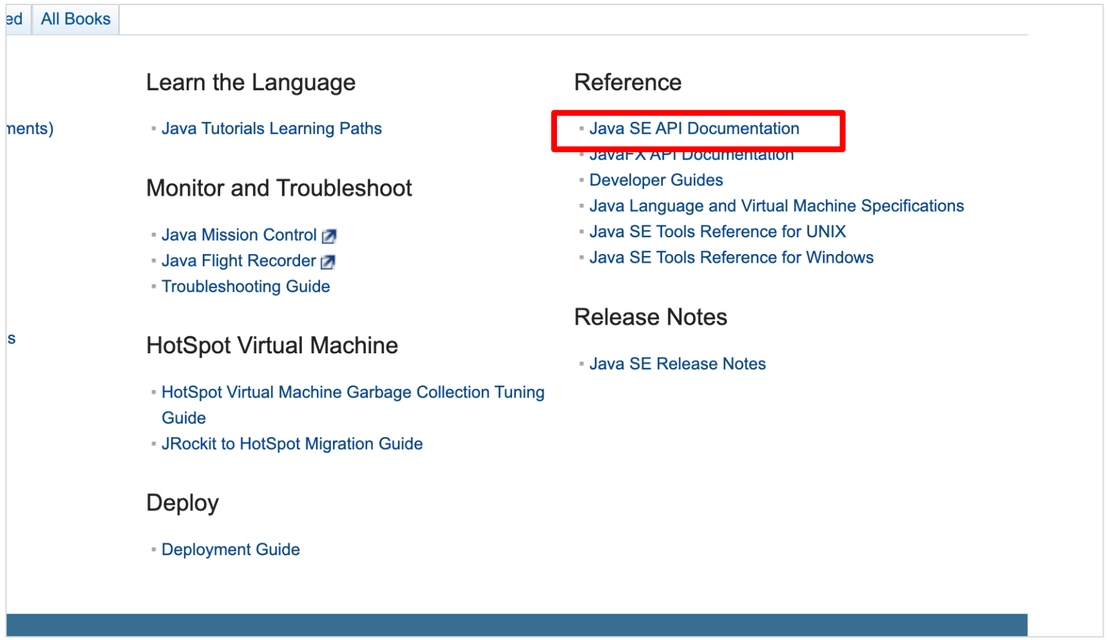
API 도큐먼트

1. API 도큐먼트 보는 법

- 자바 라이브러리를 정리해 둔 페이지
- 필요할 때 찾아볼 것

https://docs.oracle.com/en/java/javase/index.html

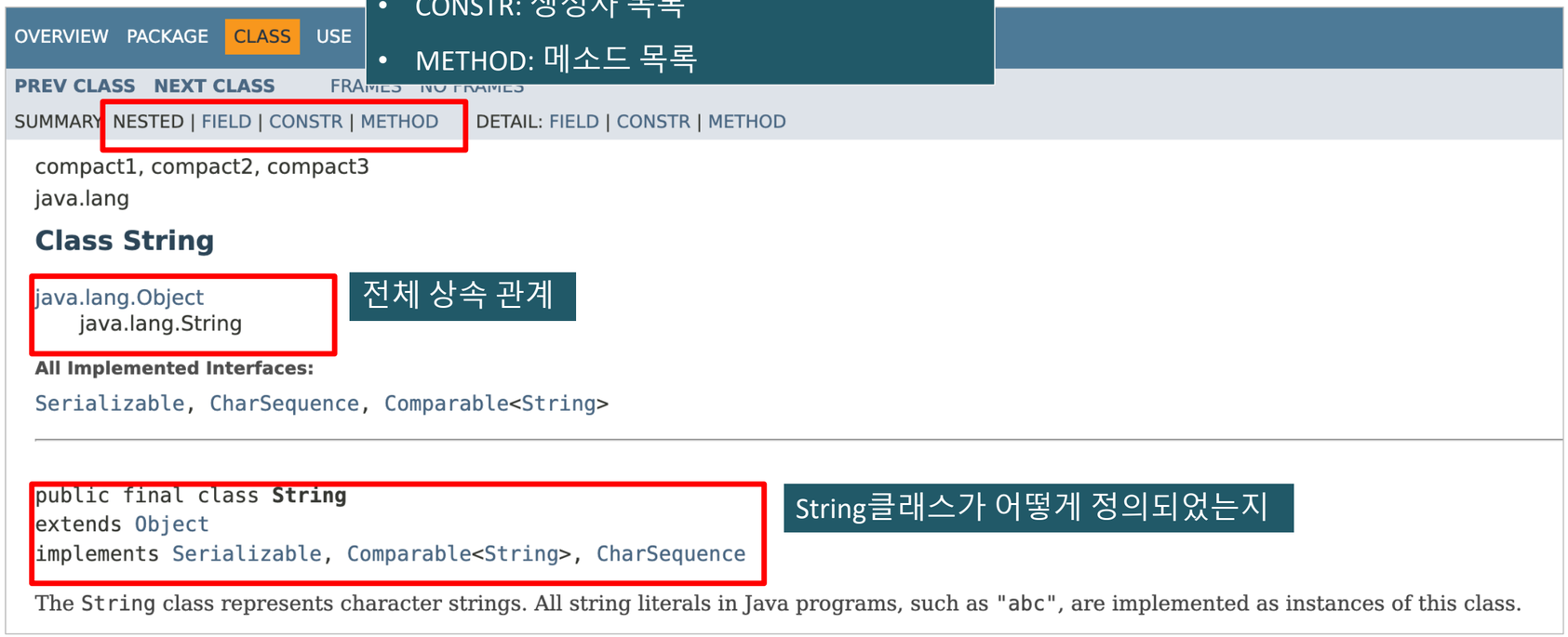




API 도큐먼트

1. API 도큐먼트 보는 법

- NESTED: 중첩 클래스/중첩 인터페이스 목록
- FIELD: 필드 목록
- CONSTR: 생성자 목록
- METHOD: 메소드 목록



PART 01

API 도큐먼트

1. API 도큐먼트 보는 법

- All Method: 모든 메소드 목록
- Static Method: 정적 메소드 목록
- Instance Method: 인스턴스 메소드 목록
- Concreate Method: 완전한 실행부를 갖춘 메소드
- Deprecated Method: 향후 제거될 메소드

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods** | **Deprecated Methods**

Modifier and Type	Method and Description
char	**charAt**(int index) Returns the char value at the specified index.
int	**codePointAt**(int index) Returns the character (Unicode code point) at the specified index.
int	**codePointBefore**(int index) Returns the character (Unicode code point) before the specified index.

Concrete 메소드란? 추상클래스 내부에서 완전한 실행부가 있는 메소드를 말한다.

```
1 abstract class A{
2 void concreteMethod(){
3 System.out.println(“this is a concrete method”);
4 }
5 abstract void abstractMethod();
6 }
```


PART
02
“
JAVA_base 모듈
”

JAVA.base모듈

- java.base: 모든 모듈이 의존하는 기본 모듈. requires하지 않아도 사용할 수 있음.

패키지	용도
java.lang	자바 언어의 기본 클래스를 제공
java.util	자료 구조와 관련된 컬렉션 클래스 제공
java.text	날짜 및 숫자를 원하는 형태의 문자열로 만들어 주는 클래스 제공
java.time	날짜 및 시간을 조작하거나 연산하는 클래스 제공
java.io	입출력 스트림 클래스 제공
java.net	네트워크 통신과 관련된 클래스 제공
java.nio	데이터 저장을 위한 Buffer 및 새로운 입출력 클래스 제공

<java.lang 패키지에 포함된 주요 클래스와 용도>

클래스		용도
Object		• 자바 클래스의 최상위 클래스
System		• 데이터 입출력 • 프로세스 종료 • 진행 시간 읽기, 시스템 속성 읽기 등
문자열 관련	String	• 문자열 저장, 조작
	StringBuilder	• 효율적인 문자열 조작 기능이 필요할 때 사용
	java.util.StringTokenizer	• 구분자로 연결된 문자열을 분리할때 사용
포장 관련	Byte, Short, Character, Interger, Float, Double, Boolean	• 기본 타입의 값을 포장 • 문자열을 기본타입으로 변환할 때 사용
Math		• 수학 계산
Class		• 클래스의 메타정보(이름, 구성 멤버) 등을 조사할 때 사용

PART 02

DecimalFormat

- Decimalformat: 숫자를 형식화된 문자열로 변환하는 기능

기호	의미	패턴 예	1234567.89 -> 변환결과
0	10진수	0 0.0 0000000000.00000	1234568 1234567.9 0001234567.89000
#	10진수	# #.# ##########.#####	1234568 1234567.9 1234567.89
.	소수점	#.0	1234567.9
-	음수기호	+#.0 -#.0	+1234567.9 -1234567.9
,	단위 구분	#,###,0	1.2E6
E	지수 문자	0.0E0	+ -
;	양수와 음수의 패턴을 모두 기술할 경우 패턴 구분자	+#,### ; -#,###	+1,234,568(양수일 때) -1,234,568(음수일 때)
%	%문자	#.# %	123456789 %
\u00A4	통화기호	\u00A4 #,###	₩1,234,568

PART 02

SimpleDateFormat

- SimpleDateformat: 날짜를 형식화된 문자열로 변환하는 기능

패터문자	의미
y	년
M	월
d	일
D	월 구분이 없는 일(1~365)
E	요일
a	오전/오후
w	년의 몇 번째 주
W	월의 몇 번째 주
H	시(0~23)
h	시(1~12)
K	시(0~11)
k	시(1~24)
m	분
s	초
S	밀리세커드(1/1000초)

정규 표현식

- 문자열이 정해진 형식으로 구성되어 있는지 검증되야 하는 경우 정규표현식 이용

표현 및 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	한개의 숫자, [0-9]와 동일		
\	공백		
\	한 개의 알파벳 또는 한 개의 숫자, [a-zA-Z_0-9]와 동일		
\.	.		
.	모든 문자 중 한 개의 문자		
?	없음 또는 한 개		
*	없음 또는 한 개 이상		
+	한 개 이상		
{n}	정확히 n개		
{n,}	최소한 n개		
{n, m}	n개부터 m개 까지		
a | b	a 또는 b		
{}	그룹핑		

PART
03

“

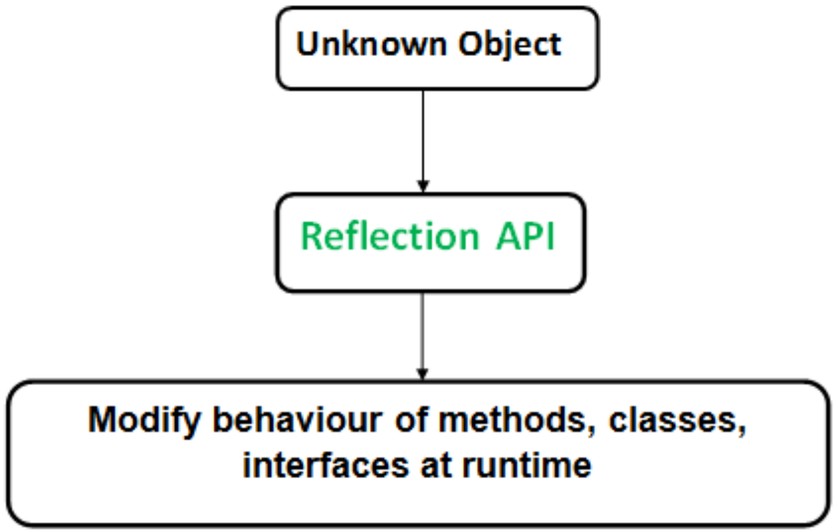
리플렉션

”

리플렉션

- 리플렉션: 메타 정보(패키지 정보, 타입 정보, 멤버-생성자, 필드, 메소드)를 프로그램에서 읽고 수정하는 행위

- 클래스 정보를 일목요연하게 볼수 있다.
- 어노테이션하고도 관련있다.




```
① Class clazz = 클래스이름.class;
② Class clazz = Class.forName("패키지…클래스이름");
③ Class clazz = 객체참조변수.getClass();
```

② 클래스로부터 얻는 방법

③ 객체로부터 얻는 방법

리플렉션

- 패키지 타입, 정보 얻기

메소드	용도
Package getPackage( )	패키지 정보 읽기
String getSimpleName( )	패키지를 제외한 타입 이름
String getName( )	패키지를 포함한 전체 타입 이름

- 멤버 정보 얻기

메소드	용도
Constructor[ ] getDeclaredConstructors( )	생성자 정보 읽기
Field[ ] getDeclaredFields( )	필드 정보 읽기
Method[ ] getDeclaredMethods( )	메소드 정보 읽기

- 리소스 경로 얻기

메소드	용도
URL getResource(String name)	리소스 파일의 URL 리턴
InputStream getResourceAsStream(String name)	리소스 파일의 InputStream 리턴

```
C:\app\bin
        ¦ - Car.class
        ¦ - photo1.jpg
        ¦ - images
              ¦ - photo2.jpg
```


PART
04
“
어노테이션
”

PART 04

어노테이션

1. 어노테이션이란? 코드에서 @으로 작성되는 요소

1. 어노테이션의 용도
   - 컴파일 시 사용하는 정보 전달
   - 빌드 툴이 코드를 자동으로 생성할 때 사용하는 정보 전달
   - 실행 시 특정 기능을 처리할 때 사용하는 정보 전달

빌드란? 소스코드를 실행 가능한 소프트웨어 산출물로 만드는 과정
*빌드툴: gradle, maven 등..

```
@AnnotationName(prop1= "값");
@AnnotationName(prop1= "값", prop2=3);
```

2. 어노테이션 타입 정의

```
public @interface AnnotationName {
}
```

3-1. 어노테이션 적용 대상
   - 어떤 대상에 설정 정보를 적용할 것인지 명시

ElementType 열거 상수	적용 요소
TYPE	클래스, 인터페이스, 열거 타입
ANNOTATION_TYPE	어노테이션
FIELD	필드
CONSTRUCTOR	생성자
METHOD	메소드
LOCAL_VARIABLE	로컬 변수
PAKAGE	패키지

어노테이션

3-2. 어노테이션 유지 정책

- @AnnotaionName을 언제까지 유지할 것인지 지정해야 한다.

Retention Policy 열거 상수	어노테이션 적용 시점	어노테이션 제거 시점
SOURCE	컴파일할 때 적용	컴파일된 후에 제거됨
CLASS	메모리로 로딩할 때 적용	메모리로 로딩된 후에 제거됨
RUNTIME	실행할 때 적용	계속 유지됨

바이트 코드 파일(.class) 파일이 메모리(메소드 영역)으로 로딩 된 후에 사라짐

3-3. 어노테이션 설정 정보 이용

- 어노테이션의 정보를 다음 메소드를 통해 얻어낼 수 있다.

리턴 타입	메소드명(매개변수)	설명
boolean	isAnnotationPresent(AnnotationName.class)	지정한 어노테이션이 적용되었는지 여부
Annotation	getAnnotation(AnnotationName.class)	지정한 어노테이션이 적용되어 있으면 어노테이션을 리턴하고, 그렇지 않으면 null을 리턴
Annotation[]	getDeclaredAnnotations()	적용된 모든 어노테이션을 리턴

Thank You!