11-1. java.lang 패키지

혼자 공부하는 자바 (신용권 저)

목차

시작하기 전에

[핵심 키워드] : Object 클래스, System 클래스, Class 클래스, String 클래스, Wrapper 클래스, Math 클래스

[핵심 포인트]

java.lang 패키지는 자바 프로그램의 기본적인 클래스를 담은 패키지이다. 그래서 그 클래스와 인터페이스는 import 없이 사용할 수 없다. java.lang 패키지에 속하는 주요 클래스에 대해 알아본다

시작하기 전에

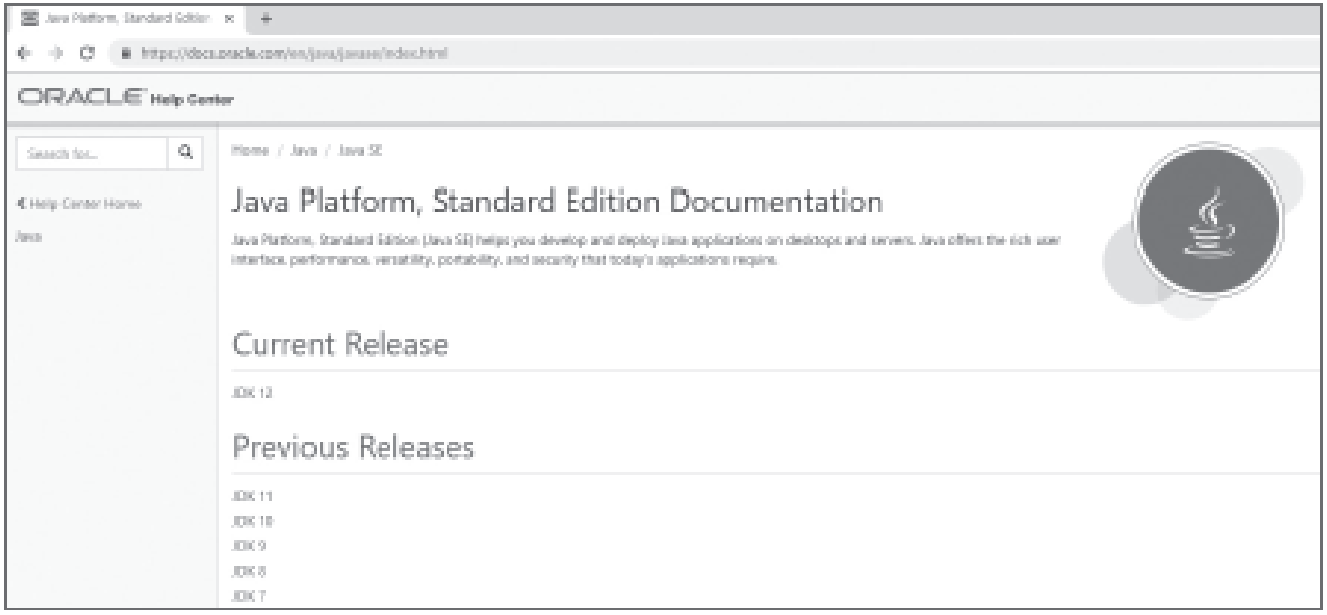
❖ java.lang 패키지의 주요 클래스와 용도

클래스		용도
Object		– 자바 클래스의 최상위 클래스로 사용
System		– 표준 입력 장치(키보드)로부터 데이터를 입력받을 때 사용 – 표준 출력 장치(모니터)로 출력하기 위해 사용 – 자바 가상 기계를 종료할 때 사용 – 쓰레기 수집기를 실행 요청할 때 사용
Class		– 클래스를 메모리로 로딩할 때 사용
String		– 문자열을 저장하고 여러 가지 정보를 얻을 때 사용
Wrapper	Byte, Short, Character Integer, Float, Double Boolean, Long	– 기본 타입의 데이터를 갖는 객체를 만들 때 사용 – 문자열을 기본 타입으로 변환할 때 사용 – 입력값 검사에 사용
Math		– 수학 함수를 이용할 때 사용

자바 API 도큐먼트

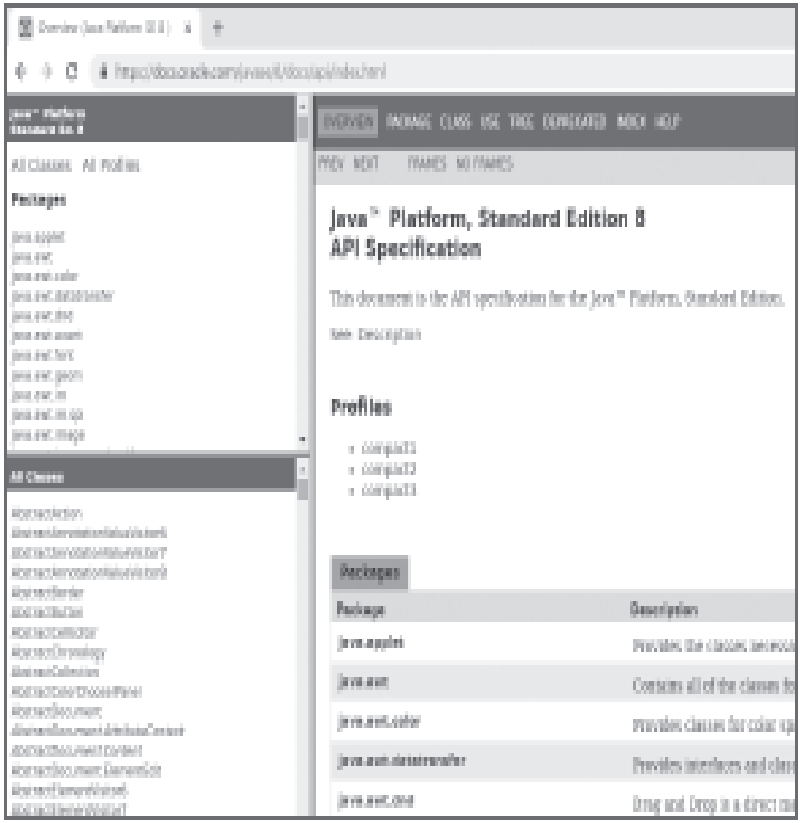
- API (Application Programming Interface)
  - 라이브러리
  - 프로그램 개발에 자주 사용되는 클래스 및 인터페이스 모음
  - API 도큐먼트로 원하는 API 쉽게 찾아 이용할 수 있음

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



자바 API 도큐먼트

JDK 8 API 도큐먼트



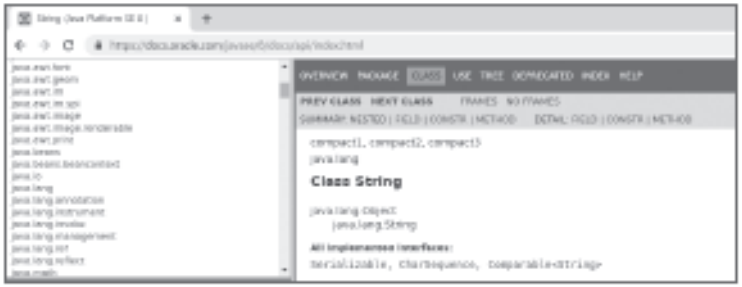
JDK 11 이후 버전 API 도큐먼트

자바 API 도큐먼트

- 각 버전의 API 도큐먼트에서 java.lang 패키지에 포함된 String 클래스 찾기

JDK 8 API 도큐먼트

1. 왼쪽 상단 Packages 목록에서 java.lang 패키지 링크를 찾아 클릭합니다.
2. 왼쪽 하단에 java.lang 패키지의 내용이 나옵니다.
3. Classes 목록에서 String 클래스 링크를 찾아 클릭합니다.
4. 오른쪽에 Class String 페이지가 나옵니다.



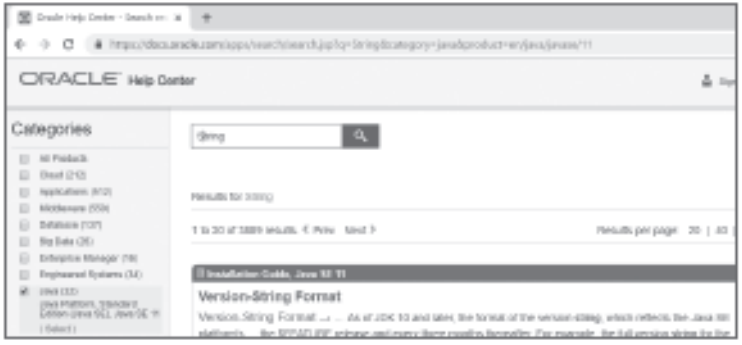
JDK 11 이후 버전 API 도큐먼트

[방법1]

1. All Modules 목록에서 java.base 링크를 찾아 클릭합니다.
2. java.base 모듈 페이지의 Packages 목록에서 java.lang 패키지 링크를 찾아 클릭합니다.
3. java.lang 패키지 페이지의 Class Summary 목록에서 String 클래스 링크를 찾아 클릭합니다.

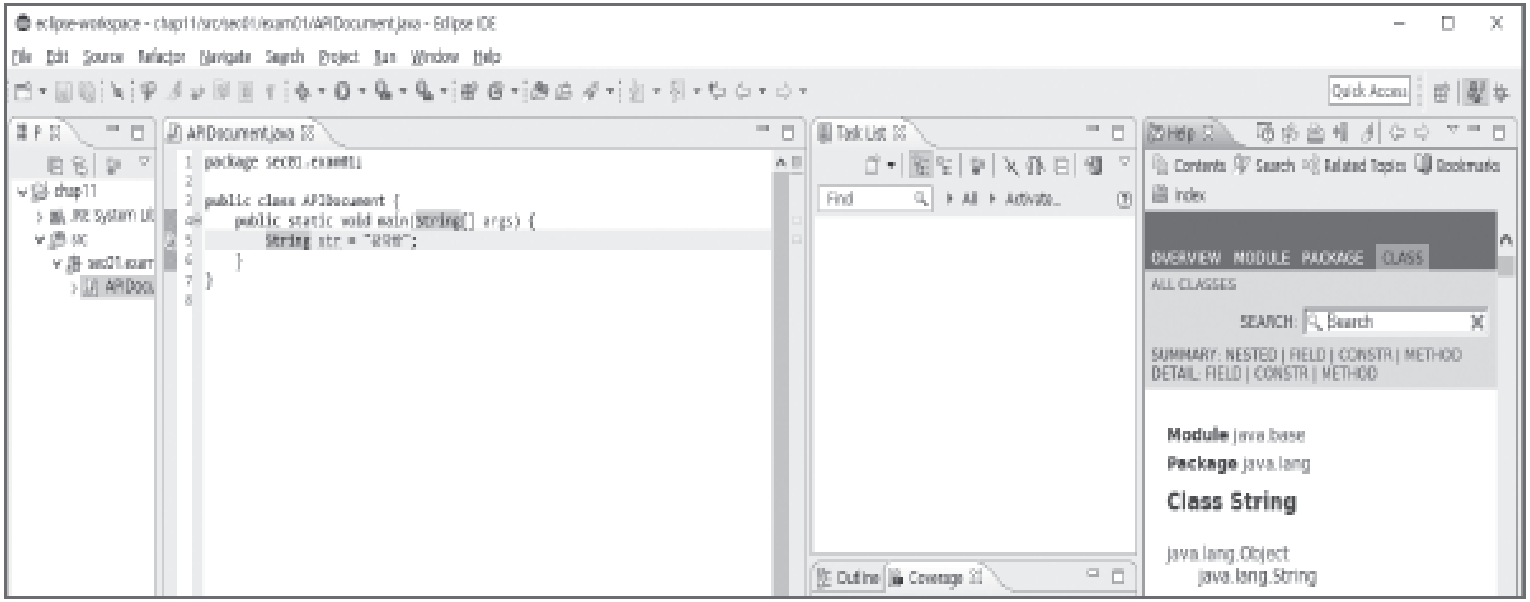
[방법2]

1. 오른쪽 상단의 Search 검색란에 "String"을 입력합니다.
2. 드롭다운 목록에서 java.lang.String 항목을 선택합니다.



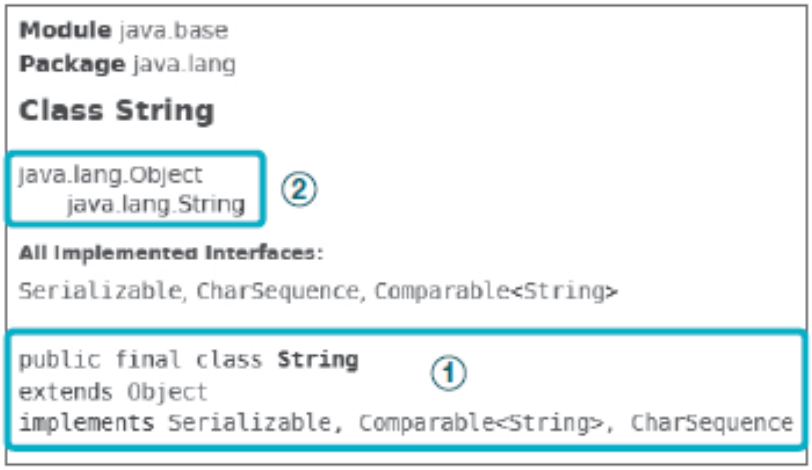
자바 API 도큐먼트

- 이클립스에서는 코드 편집 뷰의 String 클래스 선택 후 F1 키 클릭 – Help 뷰로 이동 – java.lang.String 링크 클릭



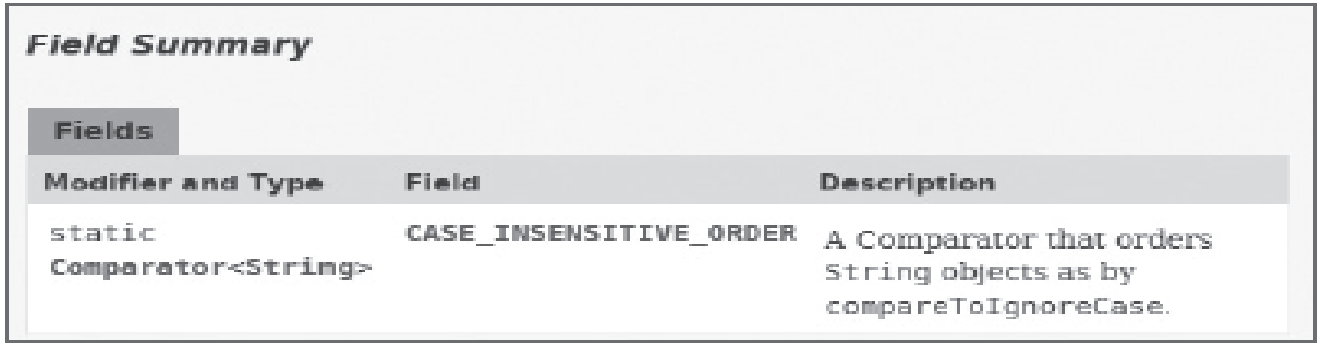
API 도큐먼트에서 클래스 페이지 읽는 방법

- 최상단 SUMMARY: NESTED | FIELD | CONSTR | METHOD
  - SUMMARY : 클래스 내 선언된 멤버 어떤 것들 있는지 알려줌
  - NESTED : 중첩 클래스 혹은 인터페이스 여부



- 그림에서 ① 클래스의 선언부
  - final 혹은 abstract 키워드 있는지 확인
  - extends 뒤 언급된 부모 클래스 확인
  - implements 키워드 뒤의 인터페이스 확인

- 클래스에 선언된 필드 목록
  - SUMMARY: NESTED | FIELD | CONSTR | METHOD 에서 FIELD 클릭하여 필드 목록으로 이동



API 도큐먼트에서 클래스 페이지 읽는 방법

- 클래스에 선언된 생성자 목록
  - SUMMARY: NESTED | FIELD | CONSTR | METHOD 에서 CONSTR 클릭하여 생성자 목록으로 이동
  - 생성자 이름 클릭하면 상세 페이지로 이동

Constructor Summary

Constructors

Constructor	Description
`String()`	Initializes a newly created `String` object so that it represents an empty character sequence.

- 클래스에 선언된 메소드 목록
  - SUMMARY: NESTED | FIELD | CONSTR | METHOD 의 METHOD 클릭하여 메소드 목록으로 이동

Method Summary

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

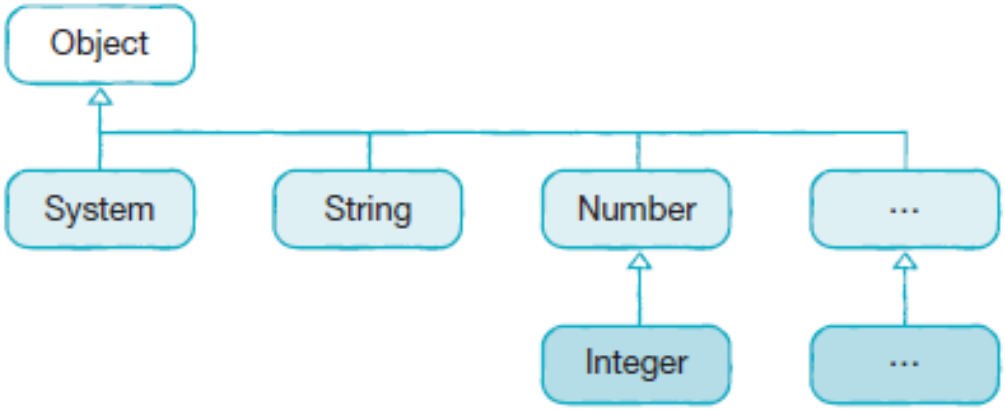
Modifier and Type	Method	Description
`char`	`charAt(int index)`	Returns the `char` value at the specified index.
`IntStream`	`chars()`	Returns a stream of `int` zero-extending the `char` values from this sequence.

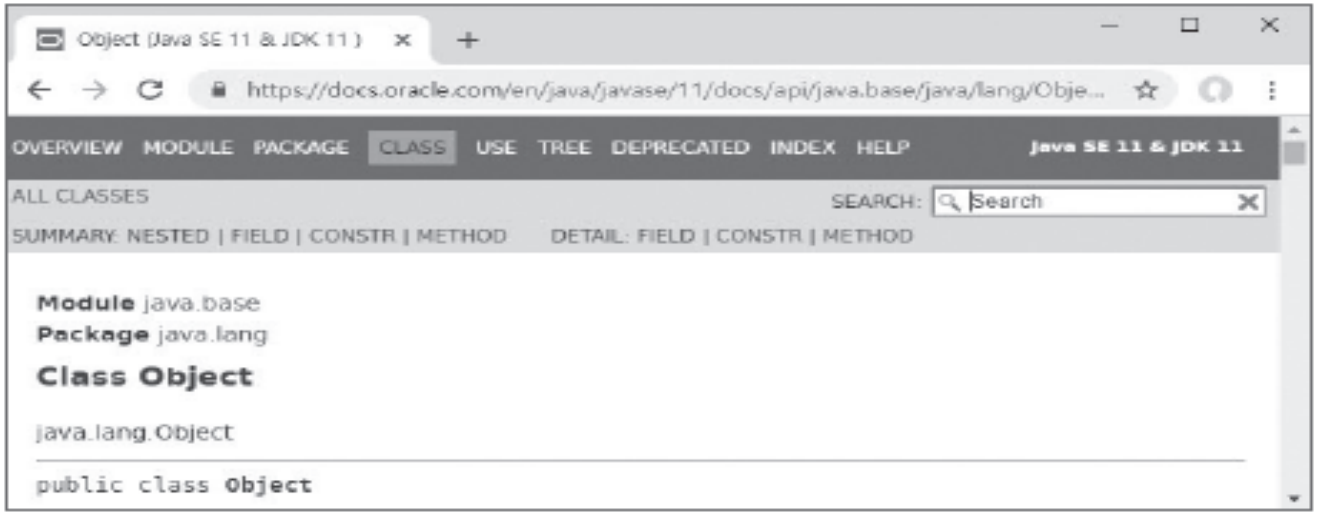
API 도큐먼트에서 클래스 페이지 읽는 방법

- [All Methods] : 전체 메소드 목록
- [Static Methods] : 정적 메소드 목록
- [Instance Methods] : 인스턴스 메소드 목록
- Modifier and Type : static 또는 protected 여부와 리턴 타입 표시
- Method와 Description의 메소드 이름 클릭하면 상세 설명 페이지로 이동

Object 클래스

❖ 모든 클래스는 Object 클래스의 자식이거나 자손 클래스





Object 클래스

❖ 객체 비교 (equals())

- equals() 메소드의 매개 타입은 Object로, 모든 객체가 매개값으로 대입될 수 있음
- Object 클래스의 equals() 메소드는 비교 연산자인 ==와 동일 결과 리턴

```
public boolean equals(Object obj) { … }
```

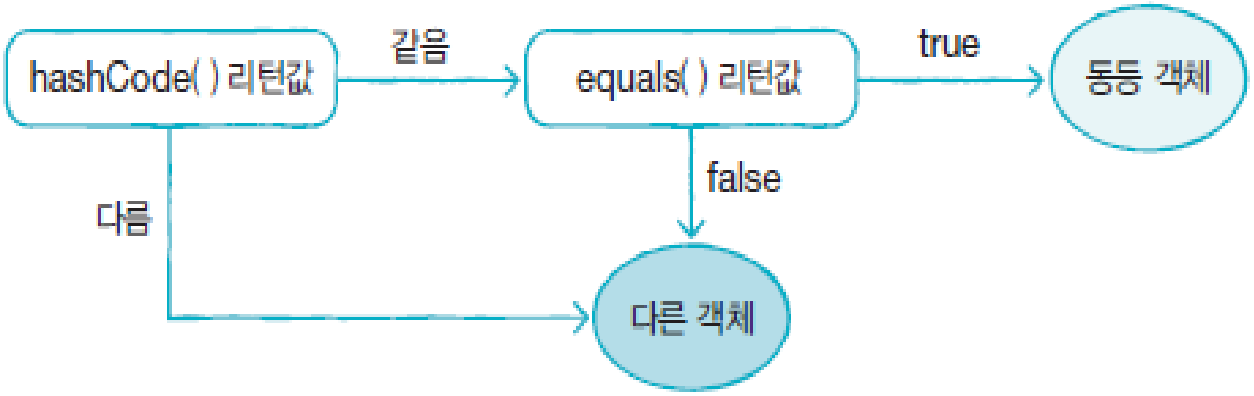
```
Object obj1 = new Object();
Object obj2 = new Object();

boolean result = obj1.equals(obj2);
                 기준 객체     비교 객체
                                          결과가 동일
boolean result = (obj1 == obj2)
```

- equals() 메소드는 두 객체가 논리적으로 동등하면 true를, 그렇지 않으면 false 리턴
- eqauls() 메소드는 매개값이 기준 객체와 동일 타입 객체인지 먼저 확인 필요

Object 클래스

❖ 객체 해시코드 (hashCode())

- 객체를 식별하는 하나의 정수값
- Object 클래스의 객체 해시코드 메소드는 객체 메모리 번지 이용하여 해시코드 만들어 리턴
  - 객체마다 다른 값 가짐
- 두 객체가 동등한지 비교 필요



Object 클래스

❖ 객체 문자 정보 (toString())

- Object 클래스의 toString() 메소드는 객체의 문자 정보 리턴
  - '클래스이름@16진수해시코드'로 구성된 문자 정보

```
Object obj = new Object();
System.out.println( obj.toString() );
```

실행결과

```
java.lang.Object@de6ced
```

- Object의 하위 클래스는 toString() 메소드 재정의하여 간결하고 유익한 정보 리턴

System 클래스

- System 클래스의 모든 필드와 메소드는 정적 필드 및 메소드로 구성

- 프로그램 종료 (exit())
  - exit() 메소드 호출하여 JVM을 강제 종료
  - exit() 메소드가 지정하는 int 매개값을 종료 상태값이라 함

System 클래스

❖ 현재 시각 읽기 (currentTimeMillis(), nanoTime())

- System 클래스의 currentTimeMillis() 및 nanoTime() 메소드로 각기 1/1000초 및 $10/10^9$ 단위 long 값 리턴

```
long time = System.currentTimeMillis();

long time = System.nanoTime();
```

Class 클래스

❖ 자바는 클래스와 인터페이스의 메타 데이터를 Class 클래스로 관리

❖ Class 객체 얻기 (getClass(), forName())

클래스로부터 얻는 방법

```
① Class clazz = 클래스이름.class
② Class clazz = Class.forName("패키지...클래스이름")
```

객체로부터 얻는 방법

```
③ Class clazz = 참조변수.getClass( );
```

- ex) String 클래스

```
① Class clazz = String.class;
② Class clazz = Class.forName("java.lang.String");
   String str = "감자바";
③ Class clazz = str.getClass();
```

Class 클래스

❖ 클래스 경로 활용하여 리소스 절대 경로 얻기

- Class 객체는 해당 클래스의 파일 경로 정보 가지고 있어 이 경로 활용해 다른 리소스 파일의 경로 얻을 수 있음

```
C:\SelfJavaStudy\chap11\bin\sec01
                              ¦ - exam09
                                    ¦ - Car.class
                                    ¦ - photo1.jpg
                                    ¦ - images
                                          ¦ - photo2.jpg
```

```
String photo1Path = clazz.getResource("photo1.jpg").getPath();
String photo2Path = clazz.getResource("images/photo2.jpg").getPath();
```

String 클래스

❖ String 생성자

- 직접 String 객체를 생성

```
//배열 전체를 String 객체로 생성
String str = new String(byte[] bytes);
//지정한 문자셋으로 디코딩
String str = new String(byte[] bytes, String charsetName);

//배열의 offset 인덱스 위치부터 length만큼 String 객체로 생성
String str = new String(byte[] bytes, int offset, int length);
//지정한 문자셋으로 디코딩
String str = new String(byte[] bytes, int offset, int length, String charsetName)
```

String 클래스

❖ String 메소드

리턴 타입	메소드 이름(매개 변수)	설명
char	charAt(int index)	특정 위치의 문자를 리턴합니다.
boolean	equals(Object anObject)	두 문자열을 비교합니다.
byte[]	getBytes()	byte[]로 리턴합니다.
byte[]	getBytes(Charset charset)	주어진 문자셋으로 인코딩한 byte[]로 리턴합니다.
int	indexOf(String str)	문자열 내에서 주어진 문자열의 위치를 리턴합니다.
int	length()	총 문자의 수를 리턴합니다.
String	replace(CharSequence target, CharSequence replacement)	target 부분을 replacement로 대치한 새로운 문자열을 리턴합니다.
String	substring(int beginIndex)	beginIndex 위치에서 끝까지 잘라낸 새로운 문자열을 리턴합니다.
String	substring(int beginIndex, int endIndex)	beginIndex 위치에서 endIndex 전까지 잘라낸 새로운 문자열을 리턴합니다.
String	toLowerCase()	알파벳 소문자로 변환한 새로운 문자열을 리턴합니다.
String	toUpperCase()	알파벳 대문자로 변환한 새로운 문자열을 리턴합니다.
String	trim()	앞뒤 공백을 제거한 새로운 문자열을 리턴합니다.
String	valueOf(int i) valueOf(double d)	기본 타입 값을 문자열로 리턴합니다.

String 클래스

- 문자 추출 (charAt())
  - 매개값으로 주어진 인덱스의 문자를 리턴

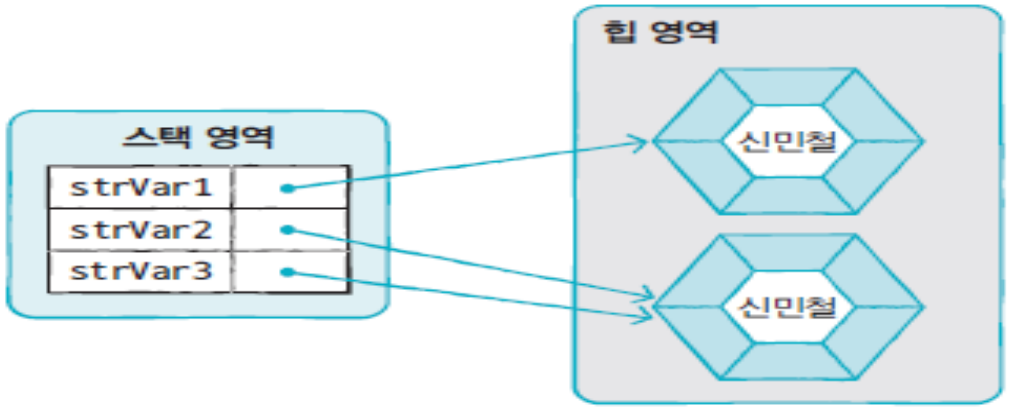
```
String subject = "자바 프로그래밍";
char charValue = subject.charAt(3);
```

  - 위 경우 '프' 리턴

String 클래스

- 문자열 비교 (equals())
  - == 연산자 사용할 경우 원하지 않는 결과 나올 수 있음

```
String strVar1 = new String("신민철");
String strVar2 = "신민철";
String strVar3 = "신민철";
```




```
strVar1 == strVar2   //false
strVar2 == strVar3   //true
```

```
strVar1.equals(strVar2)   //true
strVar2.equals(strVar3)   //true
```

String 클래스

- 바이트 배열로 변환 (getBytes())

```
byte[] bytes = "문자열".getBytes();
byte[] bytes = "문자열".getBytes(Charset charset);
```

  - getBytes()의 메소드는 시스템의 기본 문자셋으로 인코딩된 바이트 배열 리턴
  - getBytes(Charset charset) 메소드는 특정 문자셋으로 인코딩된 바이트 배열 리턴

```
try {
  byte[] bytes1 = "문자열".getBytes("EUC-KR");
  byte[] bytes2 = "문자열".getBytes("UTF-8");
} catch (UnsupportedEncodingException e) {
}
```

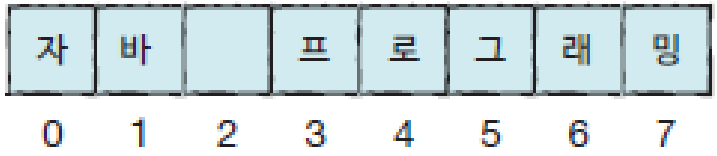
- 바이트 배열을 다시 문자열로 디코딩할 때에는 어떤 문자셋으로 인코딩되었는가에 따라 디코딩 방법 다름

```
String str = new String(byte[] bytes, String charsetName);
```

String 클래스

- 문자열 찾기 (indexOf())
  - 매개값으로 주어진 문자열이 시작되는 인덱스 리턴
  - 주어진 문자열 포함되어 있지 않으면 -1 리턴

```
String subject = "자바 프로그래밍";
int index = subject.indexOf("프로그래밍");
```



  - 위 경우 index 변수에 3이 저장

```
if( 문자열.indexOf("찾는문자열") != -1 ) {
   //포함되어 있는 경우
} else {
   //포함되어 있지 않은 경우
}
```

String 클래스

- 문자열 길이 (length())
  - 문자열의 길이를 리턴

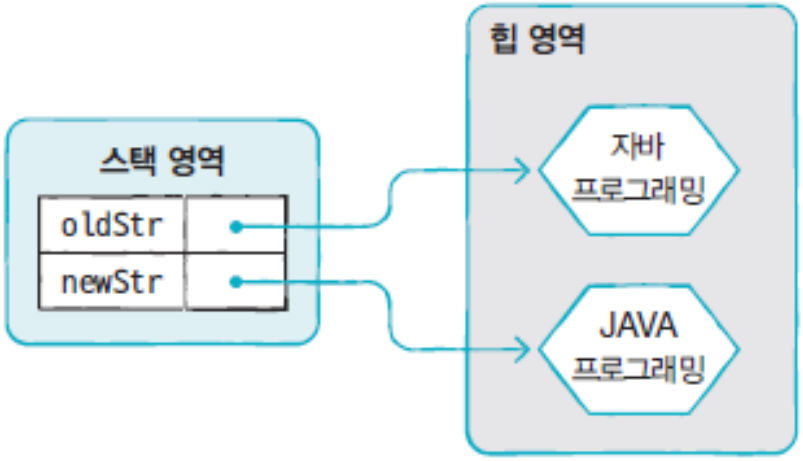
```
String subject = "자바 프로그래밍";
int length = subject.length();
```

  - 위 경우 8 저장

String 클래스

- 문자열 대치 (replace())
  - 첫 번째 매개값인 문자열을 찾아 두 번째 매개값인 문자열로 대치한 새로운 문자열 생성 및 리턴

```
String oldStr = "자바 프로그래밍";
String newStr = oldStr.replace("자바", "JAVA");
```

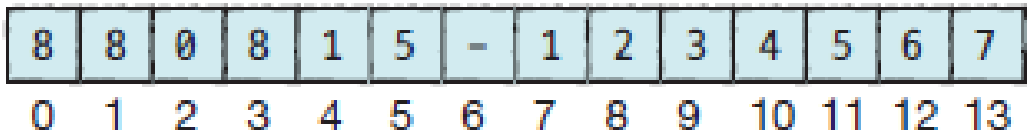


  - 위 경우 newStr 변수는 "JAVA 프로그래밍" 문자열 참조

String 클래스

- 문자열 잘라내기 (substring())
  - 주어진 인덱스에서 문자열을 추출
  - substring(int beginIndex, int endIndex) 는 주어진 시작과 끝 인덱스 사이의 문자열 추출
  - substring(int beginIndex)는 주어진 인덱스부터 끝까지 문자열 추출

```
String ssn = "880815-1234567";
String firstNum = ssn.substring(0, 6);
String secondNum = ssn.substring(7);
```

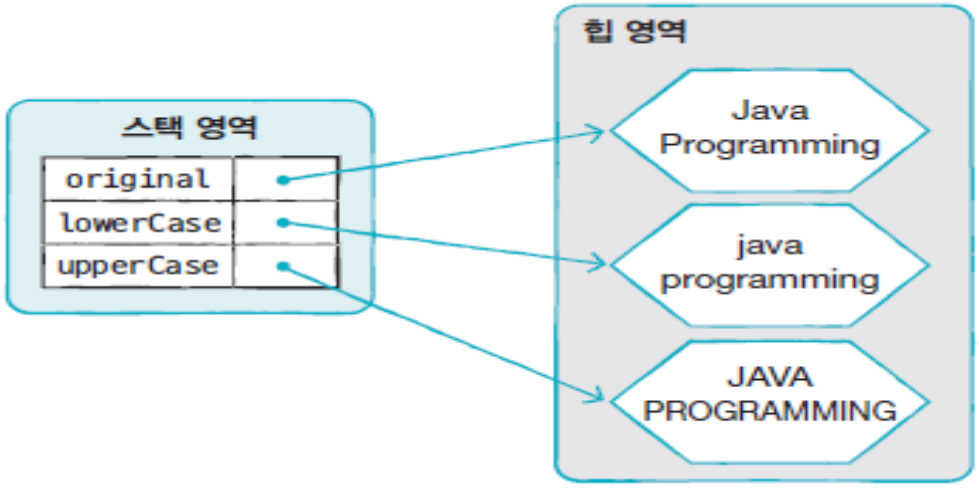


  - 위 경우 firstNum 변수값 880815, secondNum 변수값 1234567

String 클래스

- 알파벳 소, 대문자 변경 (toLowerCase(), toUpperCase())

```
String original = "Java Programming";
String lowerCase = original.toLowerCase();
String upperCase = original.toUpperCase();
```

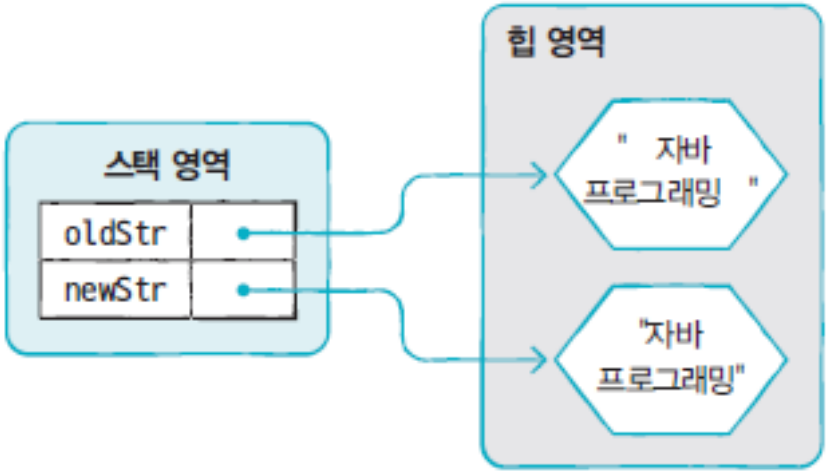


- lowerCase 변수는 새로 생성된 "java programming" 문자열 참조
- upperCase 변수는 새로 생성된 "JAVA PROGRAMMING" 문자열 참조
- 원래 original 변수의 "Java Programming" 문자열이 변경된 것 아님

String 클래스

- 문자열 앞뒤 공백 잘라내기 (trim())
  - 문자열의 앞뒤 공백 제거한 새로운 문자열 생성 및 리턴

```
String oldStr = "   자바 프로그래밍   ";
String newStr = oldStr.trim();
```



  - 위 경우 newStr 변수는 앞뒤 공백 제거되고 새로 생성된 "자바 프로그래밍" 문자열 참조

String 클래스

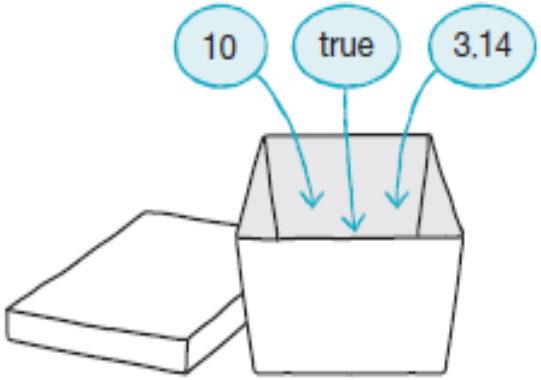
- 문자열 변환 (valueOf())
  - 기본 타입의 값을 문자열로 변환

```
static String valueOf(boolean b)
static String valueOf(char c)
static String valueOf(int i)
static String valueOf(long l)
static String valueOf(double d)
static String valueOf(float f)
```

Wrapper(포장) 클래스

❖ 포장 객체

- 기본 타입의 값을 내부에 두고 포장
- 포장하고 있는 기본 타입 값은 외부에서 변경할 수 없음
- byte, char, short, int, long, float, double, boolean 기본 타입 값 갖는 객체



기본 타입	포장 클래스
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

Wrapper(포장) 클래스

❖ Boxing과 Unboxing

- 박싱 : 기본 타입의 값을 포장 객체로 만드는 과정
- 언박싱 : 포장 객체에서 기본 타입의 값을 얻어내는 과정

기본 타입의 값을 줄 경우	문자열을 줄 경우
Byte obj = new Byte(10);	Byte obj = new Byte("10");
Character obj = new Character('가');	없음
Short obj = new Short(100);	Short obj = new Short("100");
Integer obj = new Integer(1000);	Integer obj = new Integer("1000");
Long obj = new Long(10000);	Long obj = new Long("10000");
Float obj = new Float(2.5F);	Float obj = new Float("2.5F");
Double obj = new Double(3.5);	Double obj = new Double("3.5");
Boolean obj = new Boolean(true);	Boolean obj = new Boolean("true");

Wrapper(포장) 클래스

- 생성자 이용하지 않고 각 포장 클래스마다 가진 정적 valueOf() 메소드 활용

```
Integer obj = Integer.valueOf(1000);
Integer obj = Integer.valueOf("1000");
```

- '기본 타입 이름+Value()' 메소드 호출하여 언박싱

기본 타입의 값을 이용			
byte	num	=	obj.byteValue();
char	ch	=	obj.charValue();
short	num	=	obj.shortValue();
int	num	=	obj.intValue();
long	num	=	obj.longValue();
float	num	=	obj.floatValue();
double	num	=	obj.doubleValue();
boolean	bool	=	obj.booleanValue();

Wrapper(포장) 클래스

❖ 자동 박싱과 언박싱

- 포장 클래스 타입에 기본값이 대입될 경우 자동 박싱 발생

```
Integer obj = 100;   //자동 박싱
```

- 기본 타입에 포장 객체가 대입되는 경우 및 연산에서 자동 언박싱 발생

```
Integer obj = new Integer(200);
int value1 = obj;          //자동 언박싱
int value2 = obj + 100;    //자동 언박싱
```

Wrapper(포장) 클래스

❖ 문자열을 기본 타입 값으로 변환

- 포장 클래스로 문자열을 기본 타입 값으로 변환
- 'parse+기본 타입 이름' 정적 메소드

기본 타입의 값을 이용			
byte	num	=	Byte.parseByte("10");
short	num	=	Short.parseShort("100");
int	num	=	Integer.parseInt("1000");
long	num	=	Long.parseLong("10000");
float	num	=	Float.parseFloat("2.5F");
double	num	=	Double.parseDouble("3.5");
boolean	bool	=	Boolean.parseBoolean("true");

- 예시 – 문자열을 기본 타입 값으로 변환

```
01  package sec01.exam24;
02
03  public class StringToPrimitiveValueExample {
04    public static void main(String[] args) {
05      int value1 = Integer.parseInt("10");
06      double value2 = Double.parseDouble("3.14");
07      boolean value3 = Boolean.parseBoolean("true");
08
09      System.out.println("value1: " + value1);
10      System.out.println("value2: " + value2);
11      System.out.println("value3: " + value3);
12    }
13  }
```

실행결과

```
value1: 10
value2: 3.14
value3: true
```

Wrapper(포장) 클래스

❖ 포장 값 비교

- 포장 객체는 내부 값 비교하기 위해 == 및 != 연산자 사용하지 않는 것이 좋음
  - 아래 연산 결과 false

```
Integer obj1 = 300;
Integer obj2 = 300;
System.out.println(obj1 == obj2);
```

- 박싱된 값이 아래 표에 나와 있는 범위의 값이 아닌 경우 언박싱한 값 얻어 비교해야 함

타입	값의 범위
boolean	true, false
char	\u0000 ~ \u007f
byte, short, int	-128 ~ 127

Math 클래스

❖ Math 클래스

- 수학 계산에 사용

메소드	설명	예제 코드	리턴값
int abs(int a) double abs(double a)	절대값	int v1 = Math.abs(−5); double v2 = Math.abs(−3.14);	v1 = 5 v2 = 3.14
double ceil(double a)	올림값	double v3 = Math.ceil(5.3); double v4 = Math.ceil(−5.3);	v3 = 6.0 v4 = −5.0
double floor(double a)	버림값	double v5 = Math.floor(5.3); double v6 = Math.floor(−5.3);	v5 = 5.0 v6 = −6.0
int max(int a, int b) double max(double a, double b)	최대값	int v7 = Math.max(5, 9); double v8 = Math.max(5.3, 2.5);	v7 = 9 v8 = 5.3
int min(int a, int b) double min(double a, double b)	최소값	int v9 = Math.min(5, 9); double v10 = Math.min(5.3, 2.5);	v9 = 5 v10 = 2.5
double random( )	랜덤값	double v11 = Math.random( );	0.0 <= v11 < 1.0
double rint(double a)	가까운 정수의 실수값	double v12 = Math.rint(5.3); double v13 = Math.rint(5.7);	v12 = 5.0 v13 = 6.0
long round(double a)	반올림값	long v14 = Math.round(5.3); long v15 = Math.round(5.7);	v14 = 5 v15 = 6

Math 클래스

- math.random() 메소드는 0.0 이상 1.0 미만 범위에 속하는 하나의 double 타입 값 리턴

```
0.0  <=  Math.random()  <  1.0
```

  - 예시 – 1부터 10Rkwldml 정수 난수 얻기

```
0.0 * 10   <=   Math.random() * 10   <   1.0 * 10
   ↑                                        ↑
 (0.0)                                   (10.0)
```

```
(int) (0.0 * 10)   <=   (int) (Math.random() * 10)   <   (int) (1.0 * 10)
       ↑                          ↑                             ↑
      (0)                  (0, 1,2,3,4,5,6,7,8,9)              (10)
```

```
(int) (0.0 * 10) + 1   <=   (int) (Math.random() * 10) + 1   <   (int) (1.0 * 10) + 1
       ↑                             ↑                                 ↑
      (1)                   (1,2,3,4,5,6,7,8,9,10)                    (11)
```

```
int num = (int) (Math.random() * 10) + 1;
```

키워드로 끝내는 핵심 포인트

- Object 클래스 : 자바의 최상위 부모 클래스, Object 클래스의 메소드는 모든 자바 객체에서 사용 가능
- System 클래스 : 운영체제의 일부 기능 이용할 수 있음. 정적 필드와 정적 메소드로 구성
- Class 클래스 : 클래스와 인터페이스의 메타 데이터가 Class 클래스로 관리됨.
- String 클래스 : String 클래스의 다양한 생성자 이용하여 직접 String 객체를 생성 가능.
- Wrapper 클래스 : 기본 타입의 값 갖는 객체를 포장 객체라 함. 기본 타입의 값을 포장 객체로 만드는 것을 박싱, 반대의 과정을 언박싱이라 함.
- Math 클래스 : 수학 계산에 사용할 수 있는 메소드 제공하며, Math 클래스가 제공하는 메소드는 모두 정적 메소드이므로 Math 클래스로 바로 사용 가능.

11-2. java.util 패키지

혼자 공부하는 자바 (신용권 저)

목차

- 시작하기 전에
- Date 클래스
- Calendar 클래스
- 키워드로 끝내는 핵심 포인트
- 확인문제

시작하기 전에

[핵심 키워드] : Date 클래스, Calendar 클래스

[핵심 포인트]

java.util 패키지는 프로그램 개발에서 자주 사용되는 자료구조일 뿐만 아니라, 날짜 정보를 제공하는 유용한 API를 포함하고 있다. 날짜 정보를 제공하는 API에 대해 알아본다.

시작하기 전에

클래스	용도
Date	날짜와 시간 정보를 저장하는 클래스
Calendar	운영체제의 날짜와 시간을 얻을 때 사용

Date 클래스

❖ Date 클래스

- 날짜를 표현하는 클래스
- Date는 객체 간 날짜 정보 주고받을 때 매개 변수나 리턴 타입으로 주로 사용

```
Date now = new Date();
```

- 원하는 날짜 형식의 문자열 얻기 위해 java.text 패키지의 SimpleDataFormat 클래스와 함께 사용

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy년 MM월 dd일 hh시 mm분 ss초");
```

- format() 메소드 호출

```
String strNow = sdf.format(now);
```

Calendar 클래스

❖ Calendar 클래스

- 추상 클래스이므로 new 연산자 사용하여 인스턴스 생성 불가
- getInstance() 메소드 이용하여 현재 운영체제에 설정된 시간대 기준으로 한 Calendar 하위 객체 얻을 수 있음

```
Calendar now = Calendar.getInstance();
```

```
int year   = now.get(Calendar.YEAR);          //연도를 리턴
int month  = now.get(Calendar.MONTH) + 1;     //월을 리턴
int day    = now.get(Calendar.DAY_OF_MONTH);  //일을 리턴
int week   = now.get(Calendar.DAY_OF_WEEK);   //요일을 리턴
int amPm   = now.get(Calendar.AM_PM);         //오전/오후를 리턴
int hour   = now.get(Calendar.HOUR);          //시를 리턴
int minute = now.get(Calendar.MINUTE);        //분을 리턴
int second = now.get(Calendar.SECOND);        //초를 리턴
```

키워드로 끝내는 핵심 포인트

- Date 클래스 : 날짜를 표현하는 클래스, 객체 간 날짜 정보 주고받을 때 매개 변수나 리턴 타입으로 주로 사용.

- Calendar 클래스 : 달력을 표현하는 클래스. 추상 클래스이므로 new 연산자 사용한 인스턴스 생성이 불가능. Calendar 클래스의 정적 메소드인 getInstance() 메소드 이용하여 현재 운영체제에 설정된 시간대를 기준으로 한 Calendar 하위 객체 얻을 수 있음

Thank You!

혼자 공부하는 자바 (신용권 저)

한빛미디어 Hanbit Media, Inc.