제 31 강

기본 API클래스 알아보기

교재:p178~182

목차

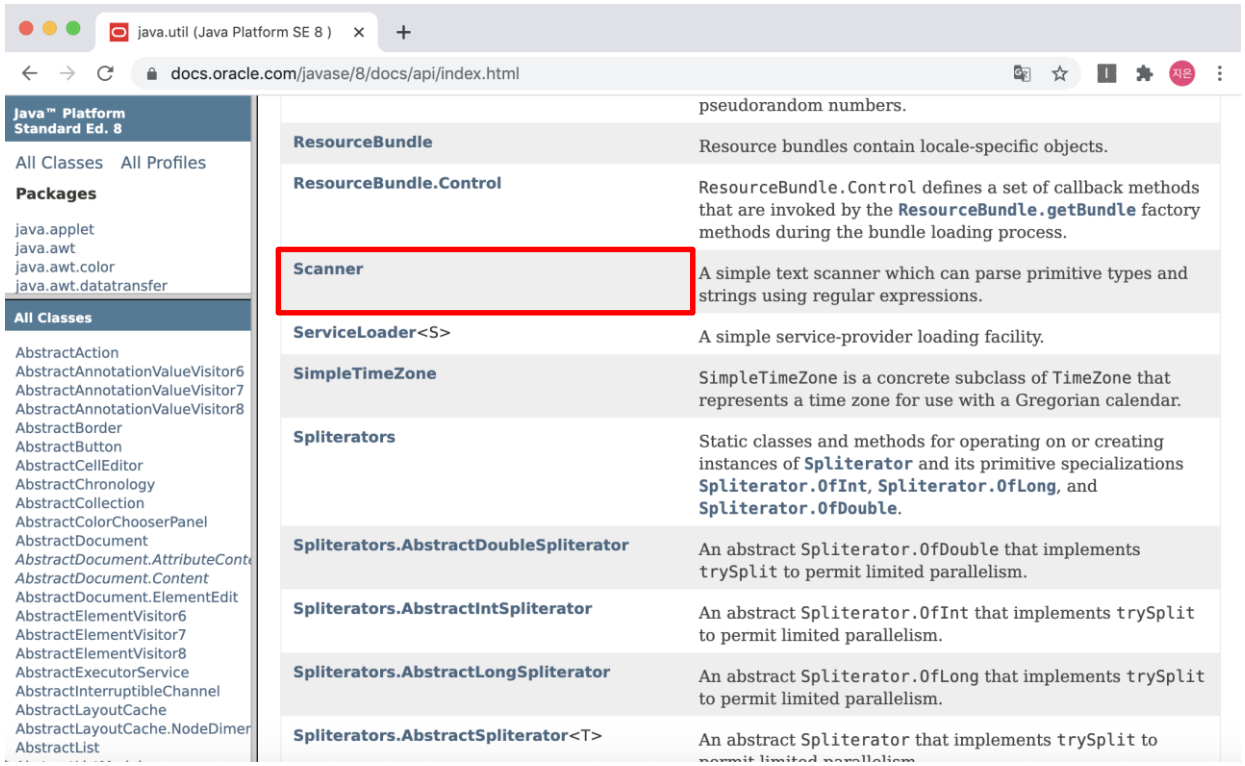
1. API란?

자바API?

API: Application Programming Interface

자바에서 개발자들을 위해 기본적으로 제공하는 클래스

https://docs.oracle.com/javase/8/docs/api/index.html



2. Object 클래스

Object 클래스?

모든 클래스의 조상 클래스

모든 클래스는 Object 클래스의 필드를 상속받는다.

Modifier and Type	메서드	설명
protected Object	clone()	객체의 복사본 반환
boolean	equals(Object obj)	obj와 같은 객체인지 확인 후 boolean 값으로 반환
protected void	finalize( )	객체가 소멸 시 호출
Class <?>	getClass( )	클래스 정보 담긴 Class 객체 반환
int	hashCode( )	해시코드 반환
void	notify( )	대기 중인 스레드 하나 깨움
void	notifyAll()	대기 중인 스레드 모두 깨움
String	toString( )	객체 정보를 문자열로 반환
void	wait( )	현재 스레드 멈추고 대기
void	wait(long timeout)	timeout만큼 스레드 멈추고 대기
void	wait(long timeout, int nanos)	실제로 일어난 시간 만큼 스레드 멈추고 대기

3. 문자열 클래스

문자열 클래스?

문자열에 대한 다양한 처리를 위한 다양한 메서드가 정의된 클래스

종류

1) String 클래스
2) StringBuffer 클래스
3) StringTokenizer 클래스

3. 문자열 클래스

1) String 클래스

String 클래스는 객체 생성 시, 보통 클래스와는 다르게, 기본 자료형 선언처럼 사용한다.

ex) String a="Java";

메서드	의미
concat(문자열)	문자열을 연결
substring(인덱스,잘라낼 길이)	문자열 자르기
length( )	문자열의 길이 반환
toUpperCase( )	문자열을 대문자로 변환
toLowerCase( )	문자열을 소문자로 변환
charAt(인덱스)	인덱스에 해당하는 글자 반환
indexOf(문자열)	문자열의 위치 반환
equals(문자열)	문자열이 같은지 boolean 값 반환
trim( )	문자열 앞,뒤 공백 제거
replace(기존문자열,대체문자열)	기존문자열을 대체 문자열로 변경
repalceAll(변환할문자열,변환된문자열)	문자열 내의 특정부분을 다른 문자열로 변경

3. 문자열 클래스

<실습> StringEx1.java

```
public class StringEx {
    public static void main(String[] args) {
        String str1 = "Hello World";
        //concat, substring,length
        System.out.println(str1.concat(" Bye World"));
        System.out.println(str1);
        System.out.println(str1.substring(6,10));
        System.out.println(str1.length());
        // toUpper, toLower
        System.out.println(str1.toUpperCase());
        System.out.println(str1.toLowerCase());
    }
}
```

<실행결과>v

```
Hello World Bye World
Hello World
Worl
11
HELLO WORLD
hello world
```

3. 문자열 클래스

<실습> StringEx2.java

```
public class StringEx2 {
    public static void main(String[] args) {
        String str1="Hello World";
        //charAt, indexOf,equals
        System.out.println(str1.charAt(6));
        System.out.println(str1.indexOf("World"));
        System.out.println(str1.equals("Hello World"));
        System.out.println(str1.equals("hello world"));

        //trim, replace,replaceAll
        System.out.println(" trim test ".trim());
        System.out.println("hello".replace('l', 'L'));
        System.out.println("hello world".replaceAll("world","WORLD"));
    }
}
```

<실행결과>

```
W
6
true
false
trim test
heLLo
hello WORLD
```

3. 문자열 클래스

2) StringBuffer 클래스

String 클래스는 immutable(변하지 않는)객체이기 때문에, 자주 사용할 수록 인스턴스를 만들기 때문에 느려진다는 단점이 있다.

따라서, 이런 단점을 보완하기 위한 클래스가 **StringBuilder, StringBuffer** 클래스다.

이 클래스들을 객체 안의 데이터를 내부적으로 변경할 수 있으므로, 새로운 객체를 만들지 않는 mutable(변할 수 있는) 객체를 만든다.

3. 문자열 클래스

2) StringBuffer 클래스

메서드	의미
append( )	매개변수로 입력된 값을 문자열로 바꾸어서 더해주는 메서드
reverse( )	문자열의 순서를 반대로 나열하는 메서드
insert(int pos, Object obj)	두 번째 매개변수의 값을 문자열로 바꾸어서 pos 위치에 추가하는 메서드
delete(int start, int end)	start 위치부터 end 직전 위치의 문자열을 제거하는 메서드
deleteCharAt(int index)	index위치에 있는 문자를 제거하는 메서드

3. 문자열 클래스

<실습> StringBufferEx.java

```
public class StringBufferEx {
    public static void main(String[] args) {
//      StringBuffer buffer="test"; --> 불가능
        StringBuffer buffer=new StringBuffer("test");
        String str="test";

        System.out.println("str:"+str);
        System.out.println("buffer:"+buffer);
        //append
        buffer.append(" Test");
        System.out.println("buffer.append:"+buffer);
        str.concat(" Test");
        System.out.println("str.concat:"+str);

        //replace(int start, int end,String str)
        buffer.replace(0, 10, "HELLO");
        System.out.println("buffer.replace:"+buffer);

        str.replace('t','T');
        System.out.println("str.replace:"+str);
```

<실행결과>

```
str:test
buffer:test
buffer.append:test Test
str.concat:test
buffer.replace:HELLO
str.replace:test
```

3. 문자열 클래스

<실습> StringBufferEx.java

```
//reverse()
buffer.reverse();
System.out.println("buffer.reverse:"+buffer);
//str.reverse()->없음

//insert()
buffer.insert(2,"BYE");
System.out.println("buffer.insert:"+buffer);

//delete
buffer.delete(0,3);
System.out.println("buffer.delete:"+buffer);
//deleteCharAt()
buffer.deleteCharAt(3);
System.out.println("buffer.deleteCharAt:"+buffer);
```

<실행결과>

```
buffer.reverse:OLLEH
buffer.insert:OLBYELEH
buffer.delete:YELEH
buffer.deleteCharAt:YELH
```

3. 문자열 클래스

String vs StringBuffer ?

String 객체: immutable 객체 - 변할 수 없는 객체

StringBuffer 객체: mutable 객체 - 변경 가능한 객체

StringBuffer vs StringBuilder?

StringBuffer: 다중 작업 처리

StringBuilder: 단일 작업 처리

3. 문자열 클래스

3) StringTokenizer 클래스

라이브러리: `import java.util.StringTokenizer;`

java.util 패키지에 있는 클래스로, 문자열을 구분문자열을 기준으로 분리할 때 사용한다.

ex)

“이름:전화번호” 문자열에서 ‘:’ 문자를 기준으로 이름과 전화번호로 나누고 싶을 때

3. 문자열 클래스

3) StringTokenizer 클래스

<사용방법>

1) 문자열 생성
2) 구분할 문자열 생성
3) StringTokenzier 객체 생성

StringTokenizer st = new StringTokenizer(문자열,구분문자열);

```
String str="kim:010-1234-234";
String delim=":";
StringTokenizer st = new StringTokenizer(str, delim);
```

3. 문자열 클래스

3) StringTokenizer 클래스

<사용할 주요 메서드>

- nextToken( ): 다음 토큰으로 이동
- hasMoreTokens( ): 다음 토큰이 존재하면 true, 존재하지 않으면 false
- countTokens( ): 남은 토큰의 개수

3. 문자열 클래스

3) StringTokenizer 클래스

<실습> StringTokenizerEx.java

```
import java.util.StringTokenizer;

public class StringTokenizerEx {
    public static void main(String[] args) {
        String str="kim:010-1234-234";
        String delim=":";
        StringTokenizer st = new StringTokenizer(str, delim);

        System.out.println(st.countTokens());

        System.out.println(st.nextToken());
        System.out.println(st.hasMoreTokens());
        System.out.println(st.countTokens());

        System.out.println(st.nextToken());
        System.out.println(st.hasMoreTokens());
        System.out.println(st.countTokens());

    }
}
```

<실행결과>

```
2
kim
true
1
010-1234-234
false
0
```