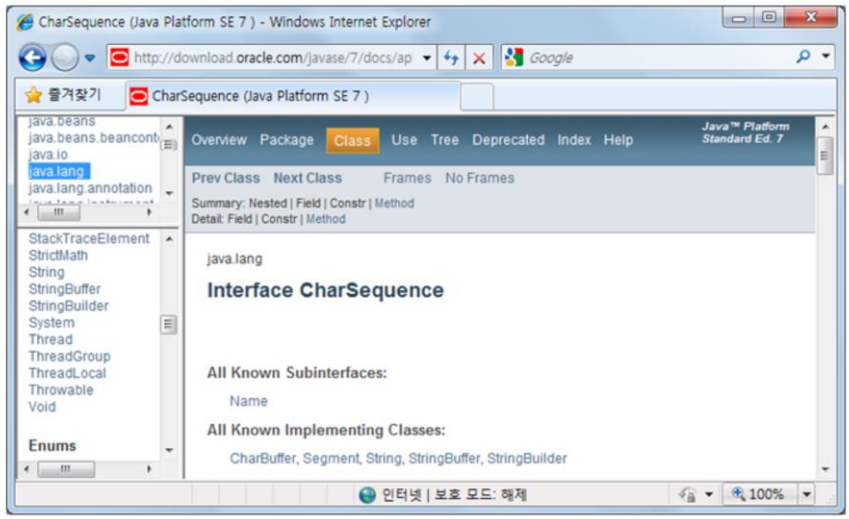
문자열

- 문자열 API
- String
- StringBuffer
- StringBuilder

11-1

문자열 API

- **CharSequence**
  - 문자열을 처리하는 최상위 인터페이스



11-2

String

- VM에서 기본적으로 제공하는 객체
- “”으로 사용하거나 new String()으로 사용할 수 있다.
- 고정 문자열
- java.lang.String
- J2SE 1.0

```
String s1="jeon";
String s2="jeon";
String s3=new String("jeon");
String s4=new String("jeon");

System.out.println(s1==s2);
System.out.println(s3==s4);
System.out.println(s1==s4);
System.out.println(s1.equals(s2));
System.out.println(s3.equals(s4));
System.out.println(s1.equals(s4));
```

11-3

String

- String 메서드 사용법
  - 메서드의 처리 결과를 s1에 추가하지 않으면 s1의 값은 유지된다.

```
String s1="Hello!!! Jeonhye0 ";

System.out.println(s1.length());
System.out.println(s1.replace('H', 'h'));
System.out.println(s1.charAt(4));
System.out.println(s1.trim().length());
System.out.println(s1.substring(2, 5));
System.out.println(s1.concat("~~~"));
System.out.println(s1.toLowerCase());
System.out.println(s1.toUpperCase());
System.out.println(s1.toLowerCase().equals(s1.toUpperCase()));
System.out.println(s1.toLowerCase().equalsIgnoreCase(s1.toUpperCase()));
```

11-4

String

- **String 생성자 사용법**

```
byte [] bytes = {65, 66, 67, 68};
char [] chars={'a', 'b', 'c', 'd'};

String s1=new String(bytes);
String s2=new String(chars);
String s3=new String(bytes, 1, 2);
String s4=new String("hello");
String s5=new String(s4);

s5=s5.replace('h', 'H');

System.out.println(s1); System.out.println(s2);System.out.println(s3);
System.out.println(s4); System.out.println(s5.toString());
```

StringBuffer

- 버퍼 메모리를 사용하여 문자열 크기 가변적으로 사용
- java.lang.StringBuffer
- J2SE 1.0

```
StringBuffer sb1=new StringBuffer("jeon");
StringBuffer sb2=new StringBuffer("jeon");

System.out.println(sb1==sb2);
System.out.println(sb1.equals(sb2));
System.out.println(sb1.toString().equals(sb2.toString()));
```

11-6

StringBuffer

- StringBuffer 메서드 사용법
  - 메서드의 처리 결과를 sb1에 추가하지 않아도 sb1의 값은 계속 변경

```
StringBuffer sb1=new StringBuffer("Hello");

System.out.println(sb1.length());
System.out.println(sb1.append("!!!"));
System.out.println(sb1.length());
System.out.println(sb1.charAt(5));
System.out.println(sb1.insert(5, "~~~"));
System.out.println(sb1.replace(0, 2, "xxxx"));
System.out.println(sb1);
```

StringBuilder

- 버퍼 메모리를 사용하여 문자열 크기 가변적으로 사용
- 단 멀티 쓰레드나 데이터를 공유하지는 못한다.
- java.lang.StringBuilder
- J2SE 5.0

```
StringBuilder sb1=new StringBuilder ("jeon");
StringBuilder sb2=new StringBuilder ("jeon");

System.out.println(sb1==sb2);
System.out.println(sb1.equals(sb2));
System.out.println(sb1.toString().equals(sb2.toString()));
```

StringBuilder

- StringBuilder 메서드 사용법
  - 메서드의 처리 결과를 sb1에 추가하지 않아도 sb1의 값은 계속 변경

```
StringBuffer sb1=new StringBuffer("Hello");

System.out.println(sb1.length());
System.out.println(sb1.append("!!!"));
System.out.println(sb1.length());
System.out.println(sb1.charAt(5));
System.out.println(sb1.insert(5, "~~~"));
System.out.println(sb1.replace(0, 2, "xxxx"));
System.out.println(sb1);
```

String vs StringBuffer vs StringBuilder

	Multi Thread	메서드 결과	메모리	equals()	비고
String	O	값 유지	고정	O	고정 데이터 유리
StringBuffer	O	값 변경	가변	X	가변 데이터 유리
StringBuilder	X	값 변경	가변	X	가변 데이터 유리

LAB

11-11