Chapter 12. 패키지와 java.lang 패키지

학습목표

- 패키지의 개념과 사용 방법을 학습합니다.
- JDK에서 제공되는 표준 API 라이브러리 사용 방법을 학습합니다.
- 사용자 패키지를 생성하는 방법과 생성된 패키지의 사용 방법을 학습합니다.
- 자바의 기본 패키지인 java.lang 패키지의 개요에 관해 학습합니다.
- 기본 자료형에 대응되는 포장(Wrapper) 클래스를 학습합니다.
- 문자열의 개요에 대해 학습합니다.
- String 클래스의 개념과 사용 방법에 관해 학습합니다.
- StringBuffer 클래스의 개념과 사용 방법에 관해 학습합니다.

목차

Section 1.

패키지의 개요와 패키지의 사용

1-1 패키지의 개요와 표준 API

- 패키지 : 기능이 유사한 클래스들을 하나로 묶어 놓은 것

서로 다른 패키지에서는 동일한 이름을 중복하여 사용할 수 있습니다

그림 12-1 패키지와 클래스 이름

1-1 패키지의 개요와 표준 API

- 자바 프로그램은 클래스들의 집합
- 자바를 개발한 개발사(오라클)에 의해 수많은 라이브러리들이 JDK(자바 개발 환경)의 표준 API(Application Programming Interface)로 제공
- JDK의 많은 라이브러리 클래스들은 패키지로 구분되어 제공
- 클래스의 이름
  - 클래스 이름 : Scanner, Object, Random, Test1 등
  - 완전 이름(FQN : Full Qualified Name) : java.util.Scanner, java.lang.Object, java.util,Random, a_package.Test1 등

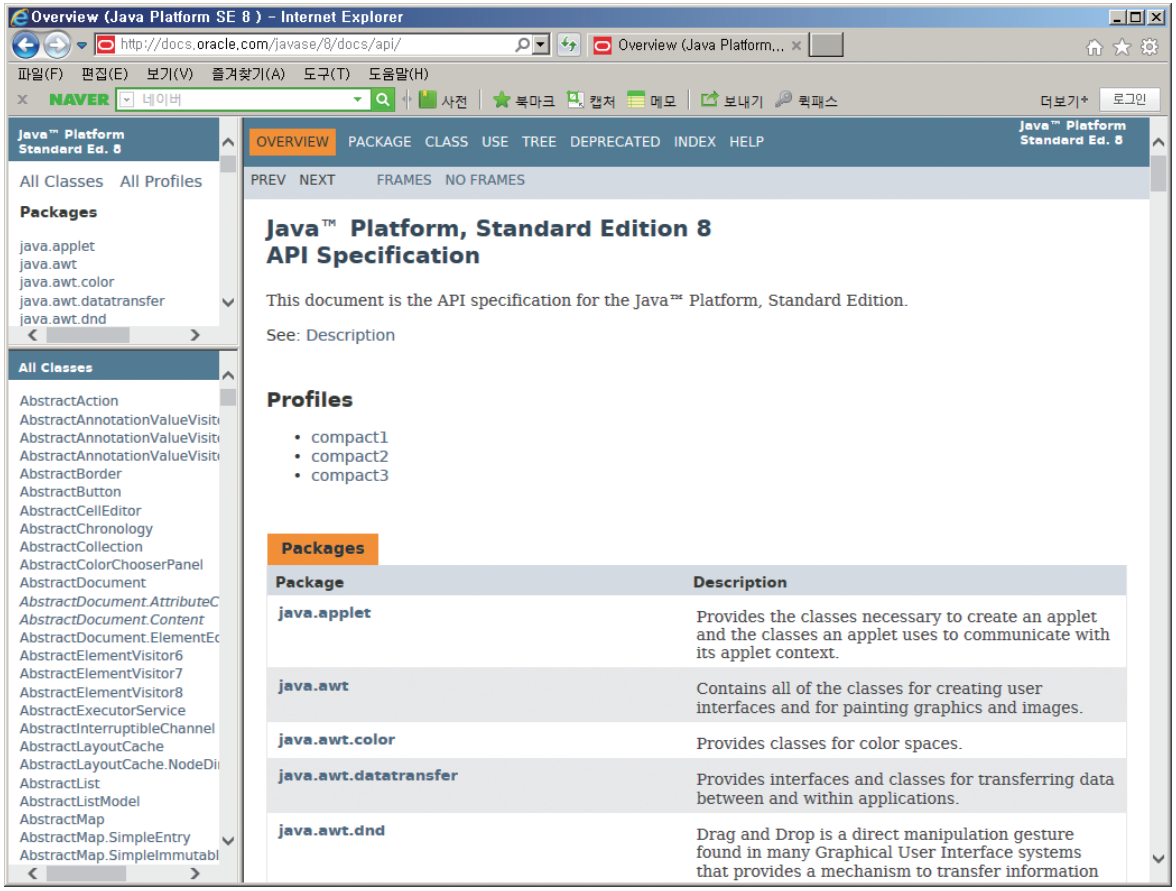
1-1 패키지의 개요와 표준 API

표 12-1 JDK에서 제공되는 주요 패키지

패키지 이름	설명
java.lang	클래스에서 지정하지 않아도 묵시적으로 포함되는 패키지로서, 자바 프로그램의 기본적인 기능을 제공하는 패키지입니다. 자바 프로그램의 최상위 클래스인 Object 클래스가 이 패키지에 포함되어 있습니다.
java.util	다양한 기능을 제공하는 유틸리티(약 50여개) 클래스들을 모아 제공하는 패키지입니다.
java.io	입출력 기능을 제공하는 클래스들을 제공하는 패키지입니다.
java.net	네트워킹과 관련된 기능을 제공하는 패키지로서 TCP/UDP 소켓과 telnet, ftp, http와 같은 프로토콜을 사용할 수 있는 클래스를 제공합니다.
java.awt	그래피컬 사용자 인터페이스(GUI)를 구축하기 위한 다양한 컴포넌트를 제공하는 패키지로서 사용자는 이 패키지를 이용하여 원하는 인터페이스를 구축할 수 있습니다.
java.awt.event	AWT 컴포넌트들의 이벤트를 제어하는 패키지입니다.
java.applet	웹 검색기에서 수행되는 애플릿 프로그램을 작성하기 위해 필요한 클래스를 제공하는 패키지입니다.
javax.swing	JVM에서 제공되는 다양한 사용자 인터페이스 컴포넌트를 제공하는 패키지입니다.

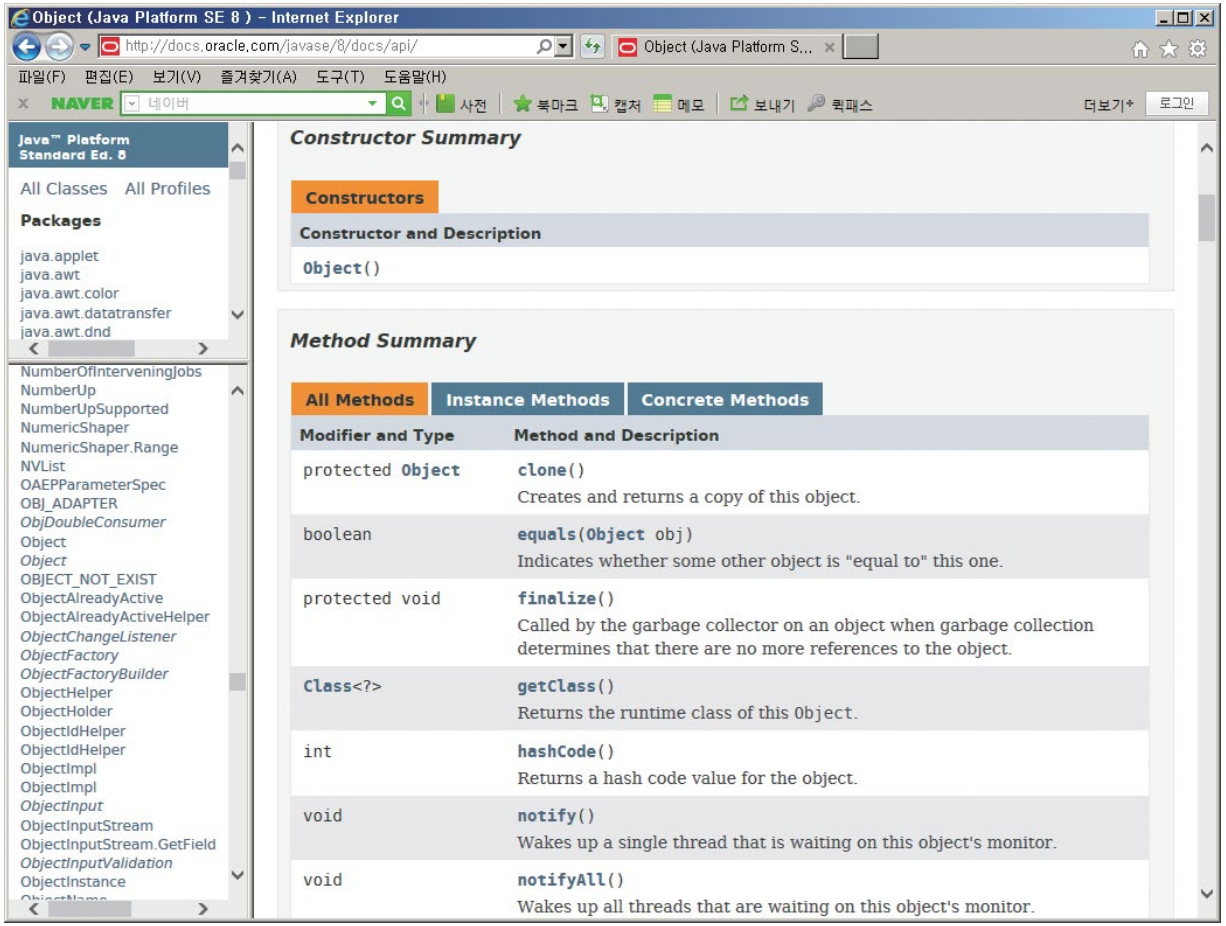
1-2 표준 API의 사용

- 자바의 개발사(오라클사)에서 제공되는 API에 관한 정보를 http://docs.oracle.com/javase/7/docs/api/" 에서 제공



1-2 표준 API의 사용

- 자바 개발자는 표준 API를 익숙하게 사용해야 한다. API에서는 클래스의 기능(메소드)과 자세한 속성 및 사용 방법을 소개하고 있다.(Object 클래스의 검색화면)



1-3 패키지의 사용

- 클래스에서 패키지를 사용하기 위해 import 예약어를 사용

```
01: import java.util.Date;
02: class My_class {
03:     ......
04:     Date date = new Date();
05:     ......
06: }
```

완전한 이름의 클래스명을 지정

```
01: import java.util.*;
02: class My_class {
03:     ......
04:     Date date = new Date();
05:     Random random = new Random();
06:     Stack stack = new Stack();
07:     Hashtable hashtable = new Hashtable();
08:     ......
09: }
```

util 패키지의 모든 클래스를 import 한다.

1-3 패키지의 사용

- 클래스에서 패키지를 사용하기 위해 import 예약어를 사용

```
01: import java.util.*;   <-------- util 패키지의 모든 클래스를 import 한다.
02: import java.sql.*;    <-------- sql 패키지의 모든 클래스를 import 한다.
03: class My_class {
04:     ......
05:     Date date = new Date();   <-------- 오류 발생. 두 개의 패키지에 Date 클래스 존재
06:     //java.util.Date = new java.util.Date(); 형태로 사용해야 한다
07:     ......
08: }
```

Section 2.

사용자 패키지 생성과 사용

1-3 패키지의 사용

- 사용자가 작성한 클래스들을 모아서 패키지로 만들 수 있다
- 사용자가 작성한 클래스를 패키지로 만들기 위해서 예약어 package를 사용

[형식] 패키지 생성

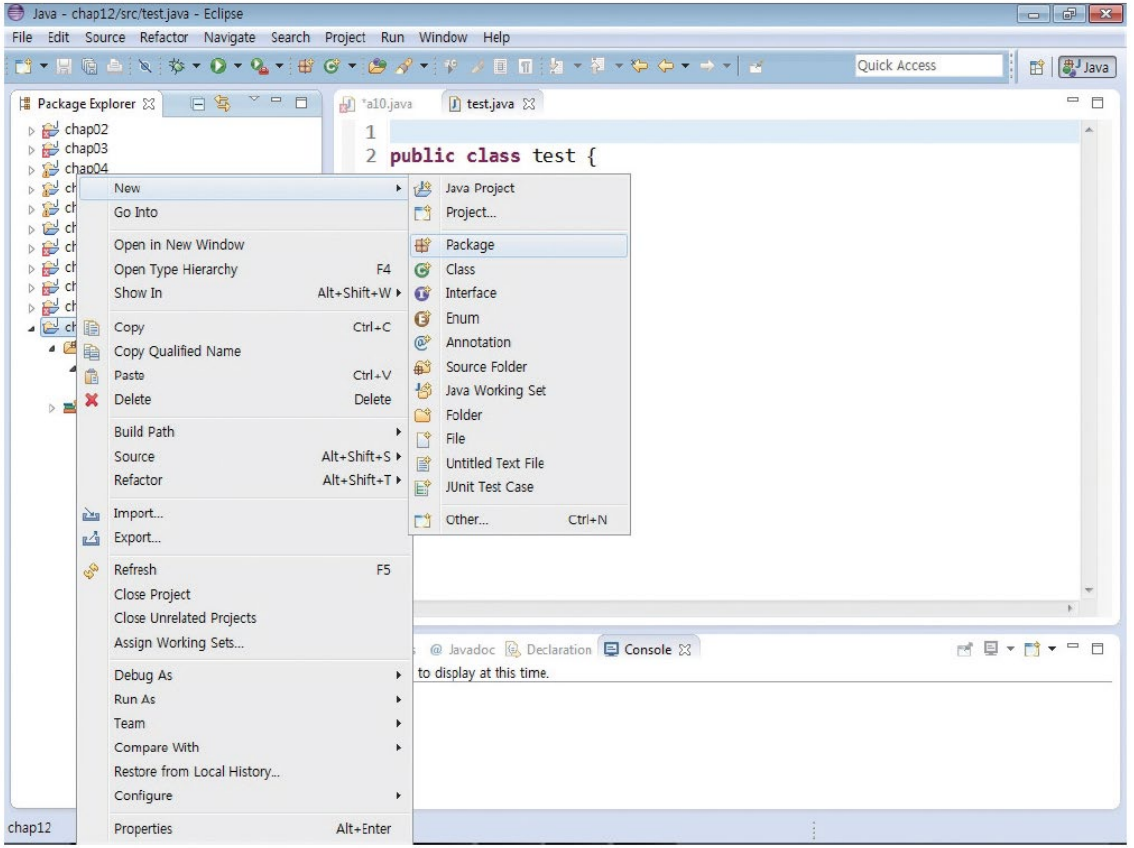
```
package 패키지명;
```

* 클래스의 첫 번째 라인에 패키지가 선언되어야 합니다

1-3 패키지의 사용

● 이클립스에서 클래스를 특정 패키지로 생성하는 과정

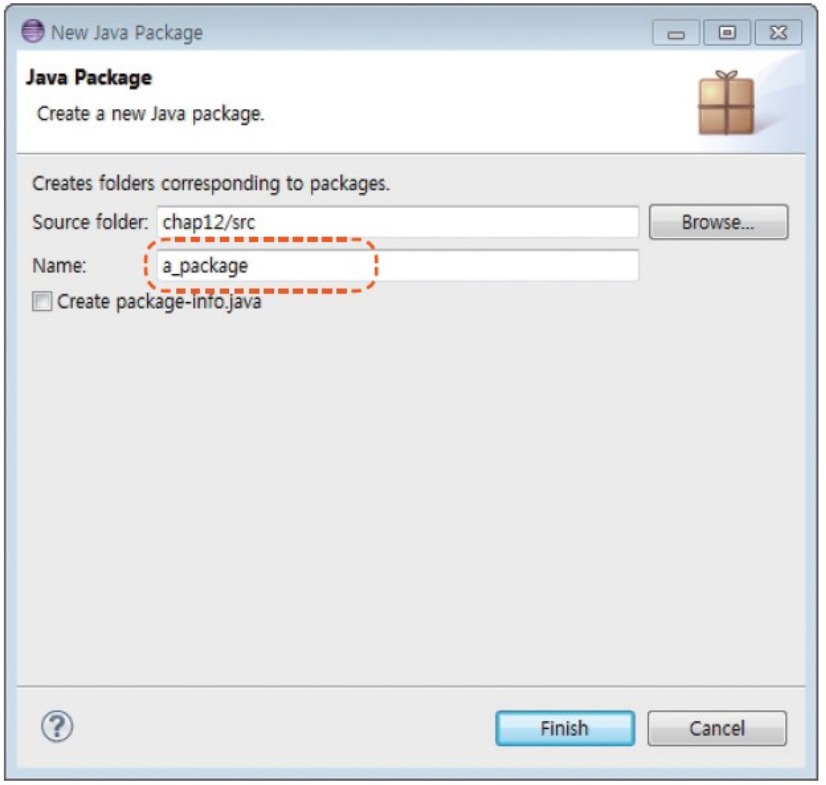
«Step 1 우선 패키지를 생성하기 위해 이클립스에서 패키지를 생성합니다.



1-3 패키지의 사용

- 이클립스에서 클래스를 특정 패키지로 생성하는 과정

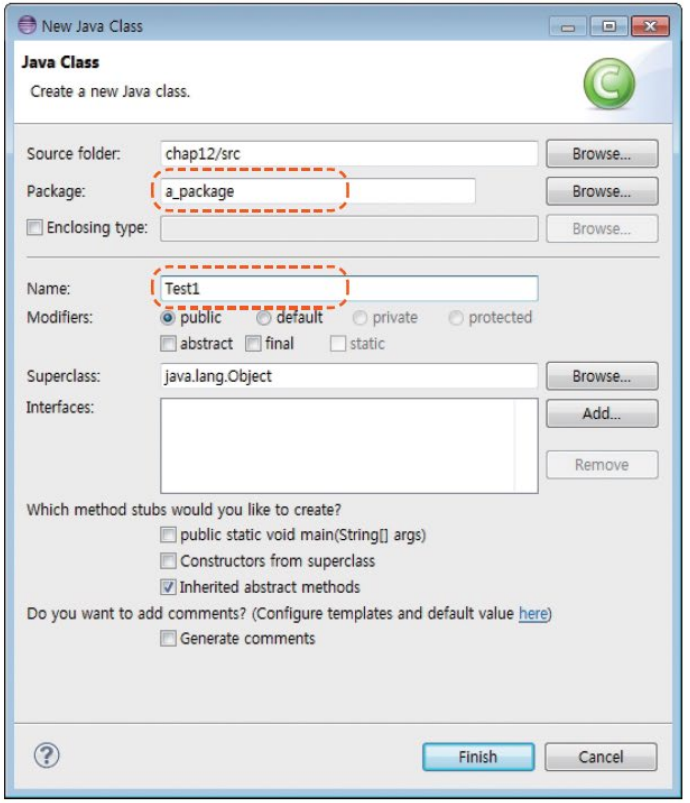
«Step 2 패키지명을 입력합니다.



1-3 패키지의 사용

● 이클립스에서 클래스를 특정 패키지로 생성하는 과정

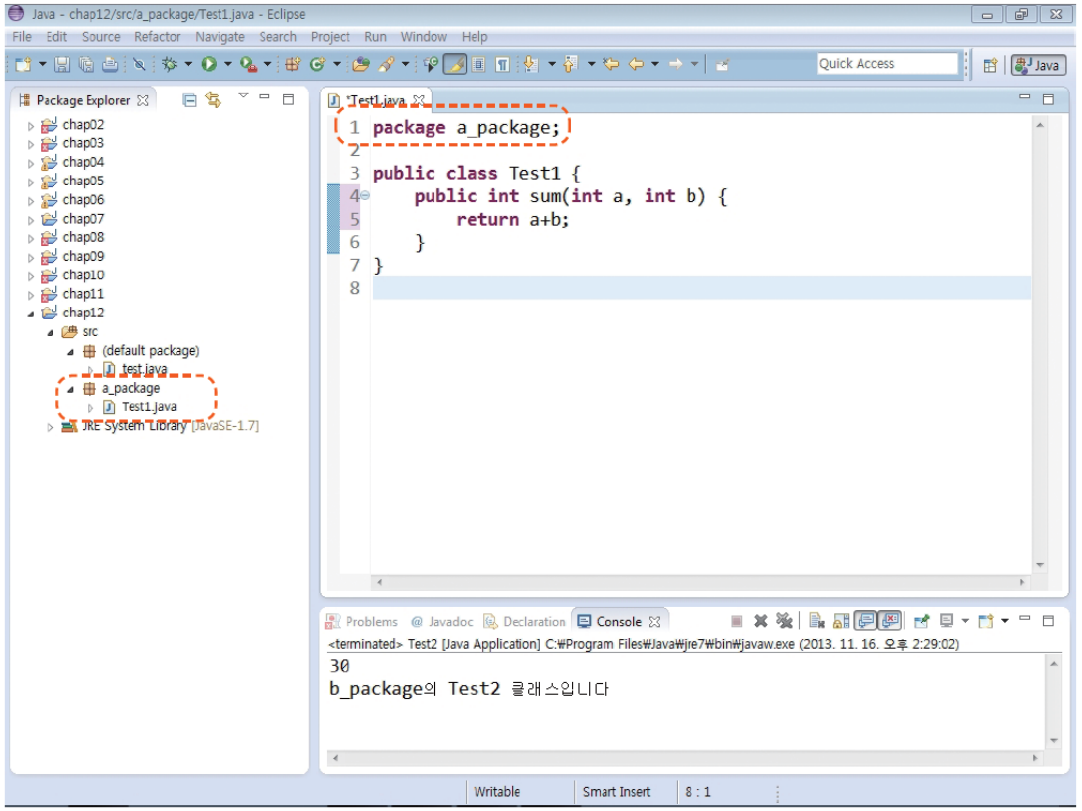
«Step 3 패키지가 생성되었습니다. 생성된 패키지명을 마우스 오른쪽 버튼으로 선택하여 Test1 클래스를 생성합니다.



1-3 패키지의 사용

● 이클립스에서 클래스를 특정 패키지로 생성하는 과정

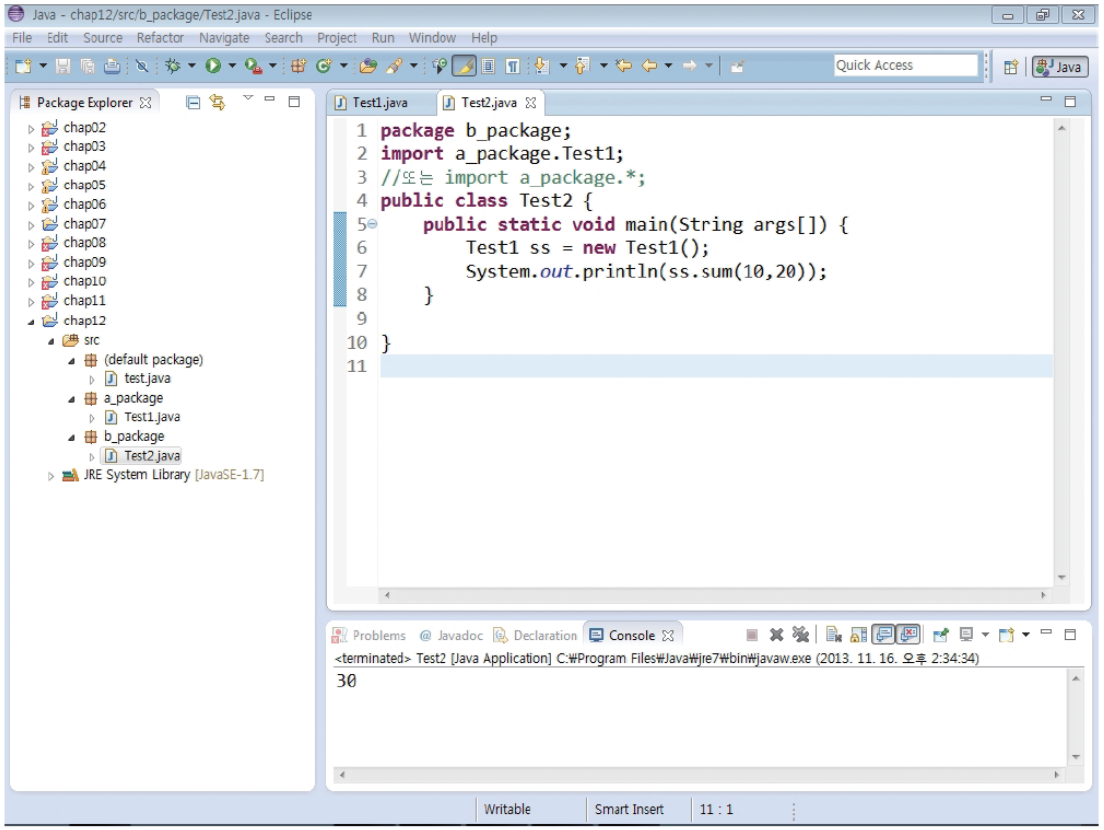
«Step 4 프로그램 편집 창의 첫 번째 라인에 자동으로 package a_package;가 삽입되었습니다. 아래의 그림과 같이 Test1 프로그램을 입력합니다.



1-3 패키지의 사용

● 이클립스에서 클래스를 특정 패키지로 생성하는 과정

≪Step 5 ①~④번까지의 과정을 반복하여 b_package를 만들고, 그 패키지에 Test2 클래스를 다음과 같이 생성하여 실행시킵니다.



1-3 패키지의 사용

예제 12.1 Test1.java

```
01: package a_package;
02: public class Test1 {
03:     public int sum(int a, int b) {
04:         return a+b;
05:     }
06: }
```

작성된 클래스의 소속을 a_package로 선언

예제 12.2 Test2.java

```
01: package b_package;
02: import a_package.Test1;
03: //또는 import a_package.*;
04: public class Test2 {
05:     public static void main(String args[]) {
06:         Test1 ss = new Test1();
07:         System.out.println(ss.sum(10,20));
08:     }
09: }
```

작성된 클래스의 소속을 b_package로 선언

Test1을 사용하기 위해 패키지를 import(완전이름 사용)

와일드 문자를 이용하여 import

a_package의 Test1 클래스 이용

실행 결과

```
30
```

Section 3.

java.lang 패키지의 개요

3 java.lang 패키지의 개요

- java.lang 패키지는 자바 프로그램에서 import문을 사용하여 포함시키지 않아도 자동적으로 포함되는 기본 패키지

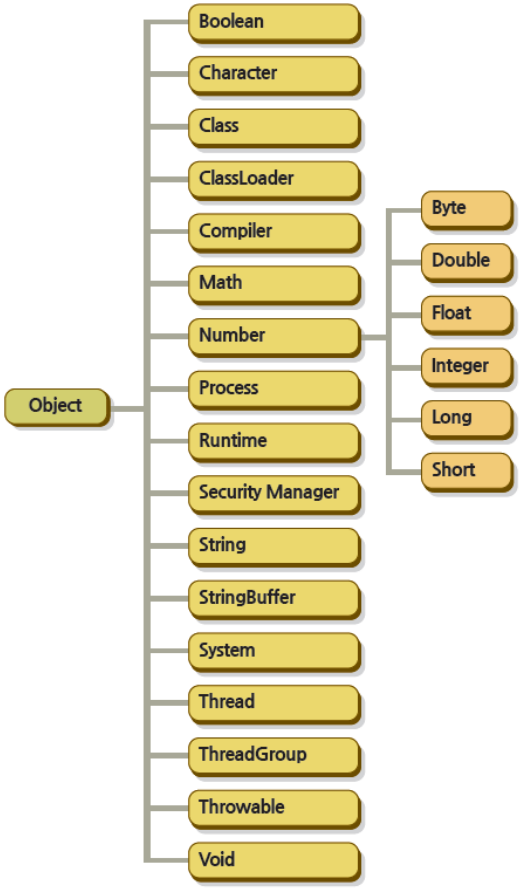


그림 12-4 java.lang 패키지의 클래스

Section 4.

포장 클래스

4 포장 클래스

- 자바는 기본 자료형과 관련된 클래스를 지원하고 있으며, 이러한 클래스들을 포장(wrapper) 클래스라고 한다
- 8개의 기본 자료형을 제공하는 포장 클래스에는 Boolean, Character, Byte, Short, Integer, Long, Float, Double 클래스가 있다

```
01: public class WrapperTest1 {
02:     public static void main(String args[]) {
03:         int num1 = 20;
04:         Integer num2 = new Integer(30);
05:         double div = num1 / num2.doubleValue();
06:         System.out.println("나눈 값 : " + div);
07:         System.out.println("두 수가 같은가? : " + num2.equals(num1));
08:     }
09: }
```

30을 가진 Integer 클래스의 객체 생성

Integer 클래스의 doubleValue() 메소드 호출

Integer 클래스의 equals() 메소드 호출

실행 결과

```
나눈 값 : 0.6666666666666666
두 수가 같은가? : false
```

4 포장 클래스

- 포장 클래스들는 유용하게 사용할 수 있는 함수 형태의 클래스 메소드를 많이 제공
- 포장 클래스의 값과 기본 자료형의 값이 자동으로 호환

```
01: int a = Integer.parseInt(args[0]);
02: double b = Double.parseDouble(args[1]);
03: char c = Character.toLowerCase('A');
04: ..........
05:
```

클래스 이름을 통하여 클래스 메소드 호출

```
01: ..........
02: Integer a = 10;
03: Double d = 3.14;
04: int aa = a;
05: double dd = d
06: ..........
```

new 연산자를 사용하지 않고 객체 생성을 통하여 클래스 메소드 호출

기본 자료형 변수에 객체가 가진 값을 저장

4.1 Integer 클래스

- **Integer 클래스는 정수값을 포장하는 클래스로서 다음과 같은 생성자를 가진다**

[형식] Integer 클래스 생성자

```
Integer(int n)
Integer(String s)
```

n : 정수값

s : 문자열

4.1 Integer 클래스

- **Integer 클래스의 주요 메소드**

메소드 이름	설명
byte byteValue()	현 객체의 값을 byte 값으로 변환하여 반환
double doubleValue()	현 객체의 값을 double 값으로 변환하여 반환
float floatValue()	현 객체의 값을 float 값으로 변환하여 반환
int intValue()	현 객체의 값을 int 값으로 변환하여 반환
long longValue()	현 객체의 값을 long 값으로 변환하여 반환
short shortValue()	현 객체의 값을 short 값으로 변환하여 반환
String toString()	현 객체의 값을 문자열로 변환하여 반환
boolean equals(Object obj)	현 객체와 obj로 지정된 객체의 값이 같으면 true, 다르면 false를 반환
static Integer decode(String str) throws NumberFormatException	문자열 str에 해당하는 Integer 객체를 반환
static int parseInt(String str) throws NumberFormatException	문자열 str에 해당하는 int 값을 반환
static int parseInt(String str, int radix) throws NumberFormatException	str로 지정된 문자열(radix 진법으로 취급)에 해당하는 int 값을 반환
static String toBinaryString(int num)	num으로 지정된 정수값의 2진법 표현을 가지는 String 객체를 반환
static String toHexString(int num)	num으로 지정된 정수값의 16진법 표현을 가지는 String 객체를 반환
static String toOctalString(int num)	num으로 지정된 정수값의 8진법 표현을 가지는 String 객체를 반환
static Integer valueOf(String str) throws NumberFormatException	문자열 str에 해당하는 Integer 객체를 반환
static Integer valueOf(String str, int radix) throws NumberFormatException	문자열 str에 해당하는 Integer 객체를 radix로 지정된 진법으로 반환

4.1 Integer 클래스

예제 12.3 IntegerTest1.java

```
01: public class IntegerTest1 {
02:     public static void main(String args[]) {
03:         Integer num1 = new Integer(13);
04:         Integer num2 = 25;
05:         num2 = num1 + num2;
06:         System.out.println("num1이 포장하고 있는 정수는 : " + num1.intValue());
07:         System.out.println("num2가 포장하고 있는 정수는 : " + num2);
08:         System.out.println("두수의 합  = " + num2);
09:         System.out.println("합의 2진 표현 : " + Integer.toBinaryString(num2));
10:         System.out.println("합의 8진 표현 : " + Integer.toOctalString(num2));
11:         System.out.println("합의 16진 표현 : " + Integer.toHexString(num2));
12:         System.out.println("if(num1 == num2) 는 : " + num1.equals(num2));
13:         Integer num3 = new Integer("444");
14:         System.out.println("문자열 '444'의 값은 : " + num3.intValue());
15:         System.out.println("2진수 '1001001'의 10진 값은 : " +
    Integer.parseInt("1001001", 2));
16:     }
17: }
```

객체 변수만 지정해도 출력 가능 (toString() 메소드 호출)

정수를 직접 지정하는 형태로 생성 가능(JDK 5.0 이상)

객체 변수만 사용하여 연산도 가능(JDK 5.0 이상)

문자열로 객체 생성

2진수를 10진수로 바꾸어 출력

클래스 메소드 호출

실행 결과

```
num1이 포장하고 있는 정수는 : 13
num2가 포장하고 있는 정수는 : 38
두수의 합  = 38
합의 2진 표현 : 100110
합의 8진 표현 : 46
합의 16진 표현 : 26
if(num1 == num2) 는 : false
문자열 '444'의 값은 : 444
2진수 '1001001'의 10진 값은 : 73
```

4.2 Character 클래스

- **Character 클래스는 char형의 값을 저장할 수 있다**
  - Character 클래스는 대부분의 메소드가 클래스 메소드

[형식] Character 클래스

```
Character(char c)
```

c : 기본 자료형 char 값

메소드 이름	설명
static boolean isDefined(char ch)	ch가 Unicode이면 true 아니면 false를 반환
static boolean isDigit(char ch)	ch가 숫자이면 true 아니면 false를 반환
static boolean isLetter(char ch)	ch가 문자이면 true 아니면 false를 반환
static boolean isLetterOrDigit(char ch)	ch가 문자 또는 숫자이면 true 아니면 false를 반환
static boolean isLowerCase(char ch)	ch가 소문자이면 true 아니면 false를 반환
static boolean isSpace(char ch)	ch가 공백 문자이면 true 아니면 false를 반환
static boolean isUpperCase(char ch)	ch가 대문자이면 true 아니면 false를 반환
static char toLowerCase(char ch)	ch로 지정된 문자를 소문자로 변환하여 반환
static char toUpperCase(char ch)	ch로 지정된 문자를 대문자로 변환하여 반환

4.2 Character 클래스

● 예제 12.4

```
01: public class CharacterTest1 {
02:     public static void main(String args[]) {
03:         char a[] = {'a','?','김','5','A'};   ← 문자의 배열을 선언
04:         for(int i=0; i < a.length; i++) {
05:             System.out.println("a[" + i + "] 번째 요소 = " + a[i]);
06:             if(Character.isDigit(a[i]))
07:                 System.out.println("  숫자입니다.");
08:             if(Character.isLetter(a[i]))
09:                 System.out.println("  문자입니다.");
10:             if(Character.isUpperCase(a[i]))
11:                 System.out.println("  대문자입니다.");
12:             if(Character.isLowerCase(a[i]))
13:                 System.out.println("  소문자입니다.");
14:         }
15:         if(Character.isDefined(a[2])) {   ← 클래스 메소드 호출
16:             System.out.println("a[2] 번째 요소 = " + a[2]);
17:             System.out.println("  유니코드입니다.");
18:         }
19:     }
20: }
```

클래스 메소드 호출

실행 결과

```
a[0] 번째 요소 = a
  문자입니다.
  소문자입니다.
a[1] 번째 요소 = ?
a[2] 번째 요소 = 김
  문자입니다.
a[3] 번째 요소 = 5
  숫자입니다.
a[4] 번째 요소 = A
  문자입니다.
  대문자입니다.
a[2] 번째 요소 = 김
  유니코드입니다.
```

Section 5.

문자열의 개요

- **자바는 문자열을 객체로 취급**
- **문자열을 위해 String 클래스와 StringBuffer 클래스를 제공**
  - 한번 생성된 다음에 변하지 않는 문자열, 즉 상수 문자열을 사용할 때는 String 클래스를 이용하고, 프로그램에서 계속 변하는 문자열을 사용할 때는 StringBuffer 클래스를 이용

Section 6.

String 클래스

- **String 클래스는 변하지 않는 문자열, 즉 상수 문자열을 위해 사용**

[형식] String 클래스 생성자

```
String()
String(char chars[])
String(char chars[], int startindex, int numChars)
String(String strObj)
String(byte asciiChars[], byte highOrderByte)
String(byte asciiChars[], byte highOrderByte, int startindex, int numChars)
String(byte asciiChars[])
String(byte asciiChars[], int startIndex, int numChars)
```

chars[] : 문자(character) 배열을 의미

startindex : 배열에서 부분 문자열을 추출하기 위해 지정된 인덱스 값을 의미

numChars : startindex에서 시작하여 추출할 문자 또는 바이트의 개수를 의미

asciiChars[] : 바이트 배열로서 아스키(ascii) 문자 배열을 의미

highOrderByte : 각 문자들(아스키, 한글 등)의 상위 바이트를 지정하는 값이며, 아스키 문자를 지정하기 위해서는 0을 사용하여야 한다.

String 클래스의 생성 예

```
01: String name1 = "cskim";
02: String name2 = "cskim";
03: String name3 = new String("cskim");
04: String name4 = new String("cskim");
05: System.out.println(name1 == name2);
06: System.out.println(name1 == name3);
07: System.out.println(name3 == name4);
08: System.out.println(name3.equals(name4));
```

문자열을 직접 지정하여 객체 생성

new 연산자를 사용하여 객체 생성

true 출력

false 출력

false 출력

true 출력

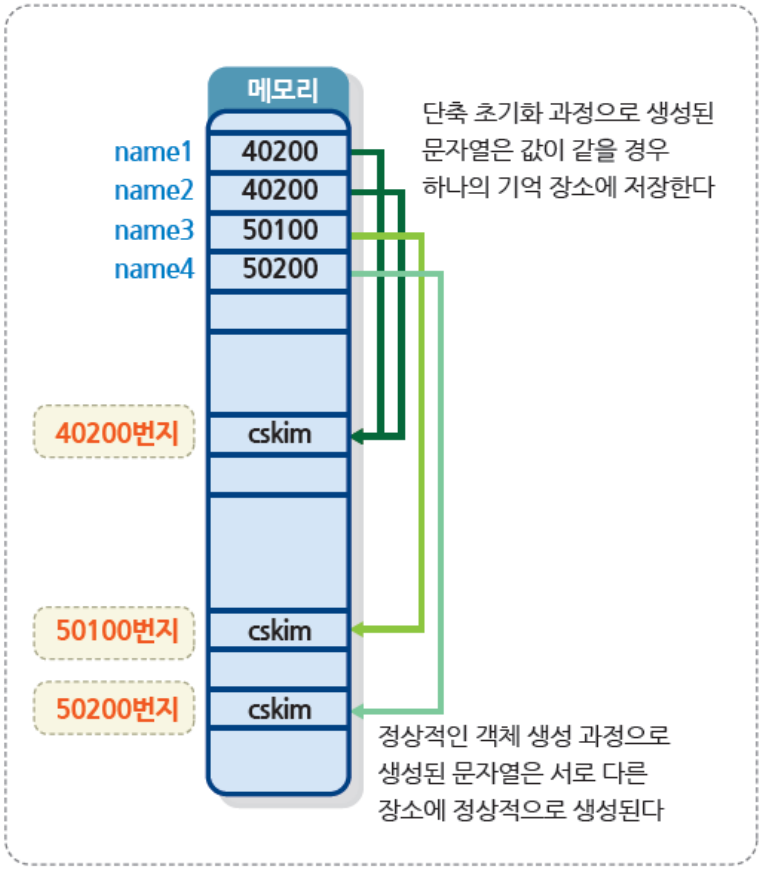


그림 12-5 문자열과 메모리

String 클래스의 주요 메소드

메소드 이름	설명
int length()	문자열의 길이를 반환
char charAt(int i)	i번째 문자를 반환
void getChars(int sourceStart, int sourceEnd, char target[], int targetStart)	문자열의 일부를 문자 배열(target[])로 작성
byte[] getBytes()	현재의 문자열을 바이트 배열로 반환
boolean equals(Object str)	현재의 문자열과 str로 지정된 문자열이 같으면 true, 다르면 false를 반환
boolean equalsIgnoreCase(String str)	현재의 문자열과 str로 지정된 문자열이 같으면 true, 다르면 false를 반환. 단, 비교 시 대소문자를 무시
boolean startsWith(String str)	현재의 문자열이 str로 시작하면 true, 아니면 false를 반환

String 클래스의 주요 메소드

메소드 이름	설명
boolean endsWith(String str)	현재의 문자열이 str로 끝나면 true, 아니면 false를 반환
int compareTo(String str)	두 개의 문자열을 비교하여 결과로 양수, 음수, 0의 값을 반환
int indexOf(char ch)	현재의 객체가 가지고 있는 문자열 내에서 ch로 지정된 문자의 첫 번째 인덱스를 반환
int indexOf(String str)	문자열 str의 첫 번째 인덱스를 반환
int indexOf(int ch, int startIndex)	인덱스 startIndex 이후의 문자 ch의 첫 번째 인덱스를 반환
int indexOf(String str, int startIndex)	인덱스 startIndex 이후의 문자열 str의 첫 번째 인덱스를 반환
String intern()	문자열의 단축 초기화 문자열을 반환
int lastIndexOf(char ch)	문자 ch의 마지막 인덱스를 반환
int lastIndexOf(String str)	문자열 str의 마지막 인덱스를 반환
int lastIndexOf(int ch, int startIndex)	인덱스 startIndex 이전의 문자 ch의 마지막 인덱스를 반환
int lastIndexOf(String str, int startIndex)	인덱스 startIndex 이전의 문자열 str의 마지막 인덱스를 반환
String[] split(String regex)	문자열을 regex로 지정된 분리 문자(delimiter)로 나누어 문자열의 배열로 반환
String substring(int startIndex)	startIndex로부터 시작하는 부분 문자열을 반환

String 클래스의 주요 메소드

String substring(int startIndex, int endIndex)	startIndex와 endIndex 사이의 부분 문자열을 반환
String concat(String str)	현재의 문자열에 str로 지정된 문자열을 결합
String replace(char original, char replacement)	original로 지정된 문자를 replacement로 지정된 문자로 대치
String trim()	문자열의 앞뒤 공백(whitespace)을 제거
static String valueOf(double num)	num을 문자열로 변환하여 반환
static String valueOf(long num)	num을 문자열로 변환하여 반환
static String valueOf(Object obj)	객체가 가지고 있는 데이터를 문자열로 변환하여 반환
static String valueOf(char chars[])	문자 배열을 문자열로 변환하여 반환
static String valueOf(char chars[], int startIndex, int numChars)	문자 배열의 일부를 문자열로 변환하여 반환
String toLowerCase()	문자열을 모두 소문자로 변환하여 반환
String toUpperCase()	문자열을 모두 대문자로 변환하여 반환

6 String 클래스

예제 12.5 StringTest1.java

```
01: public class StringTest1 {
02:     public static void main(String args[]) {
03:         char a[] = { 'C','o','m','p','u','t','e','r' };
04:         String s1 = new String(a);
05:         String s2 = new String(a,3,2);
06:         String s3 = new String("처음 시작하는 자바");
07:         String s4 = "단축 초기화 문자열";
08:         System.out.println("문자열 s1 : " + s1);
09:         System.out.println("문자열 s1의 길이 : " + s1.length());
10:         System.out.println("문자열 s2 : " + s2);
11:         System.out.println("문자열 s3 : " + s3);
12:         System.out.println("문자열 s4 : " + s4);
13:     }
14: }
```

04: 문자의 배열로 문자열 생성

05: 배열의 일부 요소로 문자열 생성

07, 08: 단축 초기화 문자열 생성

10: 문자열의 길이를 출력하는 메소드 호출

실행 결과

```
문자열 s1 : Computer
문자열 s1의 길이 : 8
문자열 s2 : pu
문자열 s3 : 처음 시작하는 자바
문자열 s4 : 단축 초기화 문자열
```

예제 12.6 StringTest2.java

```
01: public class StringTest2 {
02:     public static void main(String args[]) {
03:         String s1 = "Java Korea";
04:         String s2 = new String("Java Korea");
05:         String s3 = s2.intern();
06:         String s4 = "Java Korea";
07:         String s5 = new String("Java Korea");
08:         System.out.println("s1과  s2가 같은 장소? : " + (s1 == s2));
09:         System.out.println("s1과  s2의 값이 같은가? : " + (s1.equals(s2)));
10:         System.out.println("s1과  s3가 같은 장소? : " + (s1 == s3));
11:         System.out.println("s2와  s5가 같은 장소? : " + (s2 == s5));
12:         System.out.println("s2과  s5의 값이 같은가? : " + (s2.equals(s5)));
13:     }
14: }
```

단축 초기화 문자열로 생성

단축 초기화 문자열로 바꾸어 주는 메소드

같은 장소인지 값이 같은지를 출력

실행 결과

```
s1과  s2가 같은 장소? : false
s1과  s2의 값이 같은가? : true
s1과  s3가 같은 장소? : true
s2와  s5가 같은 장소? : false
s2과  s5의 값이 같은가? : true
```

6 String 클래스

예제 12.7 StringTest3.java

```
01: public class StringTest3 {
02:     public static void main(String args[]) {
03:         String s1 = "WorldCup Korea";
04:         System.out.println("추출된 문자 : " + s1.charAt(2));
05:         String s2 = "Apple";
06:         String s3 = "APPLE";
07:         System.out.println("s2와 s3가 동일한 문자열?(대소문자 무시) : " +
    s2.equalsIgnoreCase(s3));
08:         System.out.println("s1문자열은 \"World\"로 시작하는가? " +
    s1.startsWith("World"));
09:         System.out.println("s1문자열은 \"rea\"로 끝나는가? " +
    s1.endsWith("rea"));
10:         String s4 = "처음 시작하는 자바";
11:         System.out.println("인덱스 5부터 7이전까지의 문자열 : " +
    s4.substring(5,7));
12:         System.out.println(s4.concat("와 예제 프로그램"));
13:         System.out.println(s4.replace('하','되'));
14:     }
15: }
```

한 문자를 추출

대소문자를 무시하고 비교하는 메소드

특정 문자열로 시작하는지 알려주는 메소드

특정 문자열로 끝나는지를 알려주는 메소드

문자열의 부분 문자열을 반환하는 메소드

문자열에 지정된 문자열을 붙인다 .

문자열의 특정 문자를 바꾼다.

실행 결과

```
추출된 문자 : r
s2와 s3가 동일한 문자열?(대소문자 무시) : true
s1문자열은 "World"로 시작하는가? true
s1문자열은 "rea"로 끝나는가? true
인덱스 5부터 7이전까지의 문자열 : 하는
처음 시작하는 자바와 예제 프로그램
처음 시작되는 자바
```

6 String 클래스

예제 12.8 StringTest4.java

```
01: public class StringTest4 {
02:     static String array1[] = {"IMF", "제주도", "자바도사", "한글나라",
03:         "Computer",    "모카", "JAVA", "인터넷탐색", "초롱초롱", "come",
04:         "바람", "스크립터", "군고구마","도서", "their",    "country" };
05:     public static void main(String args[]) {
06:         System.out.println("═══════ 정렬 전 데이터 ═══════");
07:         for(String s : array1)
08:             System.out.print(s + "  ");
09:         System.out.println();
10:         System.out.println("═══════ 정렬 후 데이터 ═══════");
11:         for(int i = 0; i < array1.length; i++) {
12:             for(int j = i + 1; j < array1.length; j++) {
13:                 if(array1[i].compareTo(array1[j]) > 0) {
14:                     String t = array1[i];
15:                     array1[i] = array1[j];
16:                     array1[j] = t;
17:                 }
18:             }
19:             System.out.print(array1[i] + "  ");
20:         }
21:     }
22: }
```

compareTo()를 이용하여 두 개의 값을 정렬

실행 결과

```
═══════ 정렬 전 데이터 ═══════
IMF 제주도 자바도사 한글나라 Computer 모카 JAVA 인터넷탐색 초롱초롱 come 바람 스크립터
군고구마 도서 their country
═══════ 정렬 후 데이터 ═══════
Computer IMF JAVA come country their 군고구마 도서 모카 바람 스크립터 인터넷탐색 자바
도사 제주도 초롱초롱 한글나라
```

Section 7.

StringBuffer 클래스

- **String 클래스가 변하지 않는 문자열을 가지는 반면, StringBuffer 클래스는 변할 수 있는 문자열을 가진다**

```
String name1 = new String("Cskim");
String name2 = new String("Gentleman");
name1 = name1 + name2;
System.out.println(name1);
```

위 프로그램은 오류 없이 수행되어 "CskimGentleman"을 출력합니다. 변할 수 없는 name1 문자열이 변한 것 같지만, 위와 같은 경우에는 name1을 변경하는 것이 아니라, 새로운 name1 객체를 생성하여 "CskimGentleman"을 배정한 것입니다. 지속적으로 변하는 문자열을 사용할 경우에는 String 클래스보다는 StringBuffer 클래스가 더 효율적입니다. String 클래스는 값이 변할 때마다 새로운 객체를 생성해야 하기 때문입니다.

StringBuffer 클래스의 생성자

[형식] StringBuffer 클래스 생성자

```
StringBuffer()
StringBuffer(int size)
StringBuffer(String str)
```

size : 사이즈로 지정된 크기의 빈 객체 생성
str : 생성될 문자열

StringBuffer() 생성자는 묵시적으로 16개의 문자를 저장할 수 있는 객체를 생성합니다. StringBuffer(String str) 생성자는 str로 지정된 문자열과 추가로 16개의 문자를 더 저장할 수 있는 객체를 생성합니다.

StringBuffer 클래스의 메소드

메소드 이름	설명
StringBuffer append(boolean b)	b를 현재의 문자열 끝에 첨부
StringBuffer append(char ch)	ch를 현재의 문자열 끝에 첨부
StringBuffer append(double d)	d를 현재의 문자열 끝에 첨부
StringBuffer append(float f)	f를 현재의 문자열 끝에 첨부
StringBuffer append(int i)	i를 현재의 문자열 끝에 첨부
StringBuffer append(long l)	l을 현재의 문자열 끝에 첨부
StringBuffer append(Object obj)	obj가 가진 문자열을 현재의 문자열 끝에 첨부
StringBuffer append(Object obj)	obj가 가진 문자열을 현재의 문자열 끝에 첨부
StringBuffer append(String str)	str을 현재의 문자열 끝에 첨부
int capacity()	현재의 문자열 버퍼의 크기를 반환
char charAt(int i)	i번째 인덱스에 해당하는 문자를 반환
StringBuffer delete(int start, int end)	문자열의 start에서 end까지를 삭제
StringBuffer insert(int i, boolean b)	i번째 인덱스 전에 b를 삽입

StringBuffer 클래스의 메소드

StringBuffer insert(int i, char ch)	i번째 인덱스 전에 ch를 삽입
StringBuffer insert(int i, int j)	i번째 인덱스 전에 j를 삽입
StringBuffer insert(int i, long l)	i번째 인덱스 전에 l을 삽입
StringBuffer insert(int i, Object obj)	i번째 인덱스 전에 obj를 삽입
StringBuffer insert(int i, String str)	i번째 인덱스 전에 str을 삽입
int length()	문자열 버퍼에 있는 문자의 개수를 반환
StringBuffer reverse()	문자열을 역순의 문자열로 변환하여 반환
void setCharAt(int i, char ch)	i번째 문자를 ch로 설정
void setLength(int len)	버퍼의 크기를 len 크기로 설정
String toString()	현재의 문자열을 String 객체로 반환
String substring(int s, int e)	문자열의 s부터 e까지를 string 객체로 반환

7 StringBuffer 클래스

예제 12.9 StringBufferTest1.java

```
01: public class StringBufferTest1 {
02:     public static void main(String args[]) {
03:         StringBuffer sb1 = new StringBuffer("Hello JAVA");
04:         StringBuffer sb2 = new StringBuffer("처음 시작하는 자바");
05:         System.out.println("문자열 => " + sb1);
06:         System.out.println("문자열 길이 => " + sb1.length());
07:         System.out.println("버퍼를 포함한 길이 => " + sb1.capacity());
08:          System.out.println("버퍼에 들어 있는 내용 => " + sb2);

09:         System.out.println("문자열 끼워넣기 => " + sb2.insert(8,"Power "));
10:         System.out.println("버퍼의 8번째 문자 => " + sb2.charAt(8));
11:         sb2.setCharAt(5, '되');
12:         System.out.println("5번째 값을 '되'로 변경 => " + sb2);
13:         sb2.setLength(5);
14:         System.out.println("버퍼의 새로운 값 => " + sb2);
15:         System.out.println("문자열의 역순출력하기 => " + sb2.reverse());
16:     }
17: }
```

버퍼를 포함하여 배정된 공간을 출력

특정 위치의 값을 바꾼다.

문자열 중간에 문자열을 삽입

문자열의 길이를 고정(나머지는 삭제)

문자열을 역순으로 출력

실행 결과

```
문자열 => Hello JAVA
문자열 길이 => 10
버퍼를 포함한 길이 => 26
버퍼에 들어 있는 내용 => 처음 시작하는 자바
문자열 끼워넣기 => 처음 시작하는 Power 자바
버퍼의 8번째 문자 => P
5번째 값을 '되'로 변경 => 처음 시작되는 Power 자바
버퍼의 새로운 값 => 처음 시작
문자열의 역순 출력하기 => 작시 음처
```

Thank You!