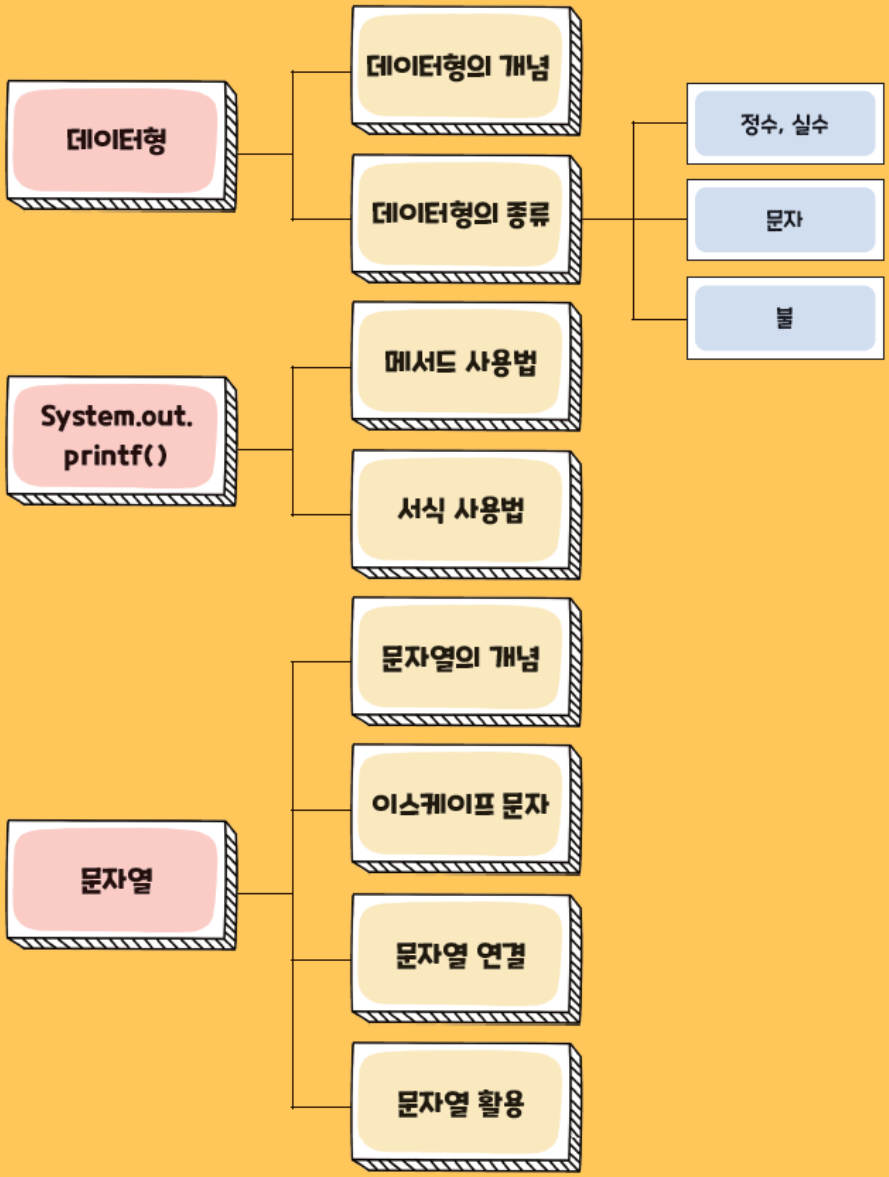
bantaehak

Chapter 04
데이터형과 문자열

목차

Preview



학습목표

- 자바의 데이터형에 대해 알아봅니다.
- 데이터형에 따른 출력 방법을 익힙니다.
- 문자열을 이해하고 응용 방법을 익힙니다.
- 문자열 메서드의 종류와 활용법을 파악합니다.

Section 01
데이터형의 개요

1. 데이터형의 개념

■ 데이터형

- 변수나 상수의 종류를 데이터형이라고 함
- 가장 많이 사용되는 기본 데이터형

  → 정수, 실수, 문자, 불형(Boolean), 문자열

```
int var1 = 100;
double var2 = 3.14;
char var3 = '자';
boolean var4 = true;
String var5 = "난생처음";
```

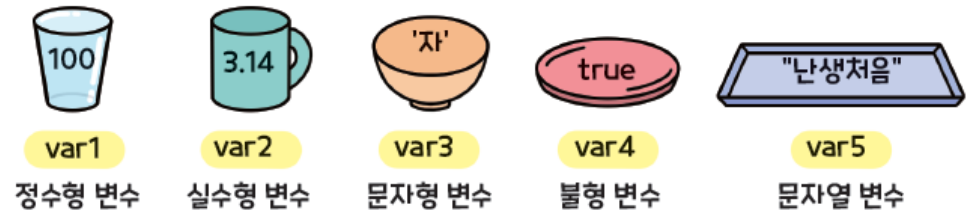


그림 4-1 변수의 종류

String은 자바의 기본 데이터형이 아니고 클래스로 제공됩니다. 하지만 문자열이 많이 사용되기 때문에 기본 데이터형과 함께 소개했습니다.

2. 변수에 맞는 데이터형의 값 대입

- 변수에 맞는 데이터형의 값 대입
  - 변수에는 해당 데이터형의 값을 대입해야 함

그림 4-2 변수에 맞는 데이터형의 값 대입

2. 변수에 맞는 데이터형의 값 대입

[하나 더 알기] 다른 데이터형을 넣어도 되는 경우

물컵에 밥을 담으면 안 되겠지만 물 대신 오렌지 주스를 담을 수는 있습니다. 이와 마찬가지로 변수에 다른 데이터형을 넣는 것이 허용되는 경우가 있습니다. 정수를 실수형 변수에 넣으면 실수로 변환됩니다. 소수점만 붙을 뿐 같은 값이기 때문에 별 문제가 없습니다.

```
double var1 = 1234;
System.out.println(var1)
```

```
1234.0
```

하지만 반대로 정수형 변수에 실수를 넣으면 오류가 발생합니다

```
int var2 = 100.0; // 오류
```

2. 변수에 맞는 데이터형의 값 대입

확인문제

다음 빈칸에 알맞은 말을 넣으시오.

- 정수형(int) 변수에는 [㉠]를 대입하고, 문자형(char) 변수에는 [㉡]를 대입해야 한다.
- 가장 많이 사용되는 기본 데이터형은 정수, 실수, [㉢], [㉣], [㉤]이다.

정답

Click!

Section 02
데이터형의 종류

1. 정수 데이터형

정수형

- 자바 프로그래밍을 하면서 정수형을 사용할 때는 그 크기를 고려해야 함
- 정수형은 상황에 따라 값의 범위를 달리 함

표 4-1 정수 데이터형

정수형	의미	크기	값의 범위
byte	아주 작은 정수	1바이트	-2^7(–128)~2^7–1(127)
short	작은 정수	2바이트	-2^{15}(–32768)~2^{15}–1(32767)
int	정수	4바이트	2^{31}(약 –21억)~2^{31}–1(약 21억)
long	큰 정수	8바이트	-2^{63}(약 –900경)~2^{63}–1(약 900경)

그림 4-3 정수의 다양한 범위

1. 정수 데이터형

■ 정수형 예제

▪ 데이터형의 값을 초과되는 경우

코드 4-1 Code04_01.java

```
01   public class Code04_01 {
02       public static void main(String[] args) {
03           byte age = 127;              // byte형의 최댓값
04           short birth = 32767;         // short형의 최댓값
05           int money = 2147483647;      // int형의 최댓값
06
07           System.out.println((byte)(age + 1));
08           System.out.println((short)(birth + 1));
09           System.out.println((int)(money + 1));
10       }
11   }
```

```
-128
-32768
-2147483648
```

(byte)와 같은 () 연산자를 캐스팅 연산자라고 하며, 이는 데이터형을 잠깐 변경하는 기능을 합니다.

2. 실수 데이터형

- 실수형
  - 3.14, -8.8과 같이 소수점이 있는 숫자를 의미함
  - 기본형은 double형임
  - float형의 경우 float를 의미하는 'f '를 숫자 뒤에 붙여야 함([예] 3.14f)

표 4-2 실수 데이터형

실수형	의미	크기	값의 범위
float	실수	4바이트	약 –3.4×1038~3.4×1038
double	큰 실수	8바이트	약 –1.79×10308~1.79×10308

2. 실수 데이터형

실수형

- float형과 double형은 둘 다 값의 범위가 크기 때문에 이 둘을 구분하여 사용하는 경우는 그리 흔치 않음
  - → float형과 double형의 중요한 차이점: 소수점 아래 자릿수
  - → double형: 소수점 아래 15~16자리
  - → float형: 소수점 아래 7자리

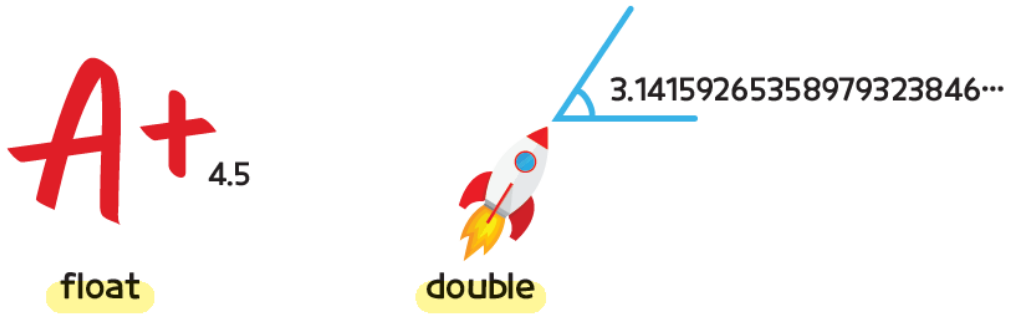


그림 4-4 실수형의 정밀도

2. 실수 데이터형

실수형 예제1

- float형과 double형의 정밀도 비교

코드 4-2 Code04_02.java

```
01  public class Code04_02 {
02      public static void main(String[] args) {
03          float f = 0.123456789012345678901234f;
04          double d = 0.123456789012345678901234;
05
06          System.out.println(f);
07          System.out.println(d);
08      }
09  }
```

```
0.12345679
0.12345678901234568
```

2. 실수 데이터형

- 실수형 예제2
  - 정수와 정수의 연산 결과

    → Q) 다음 코드에서 이상한 점은?

코드 4-3 Code04_03.java

```
01   public class Code04_03 {
02       public static void main(String[] args) {
03           int n1 = 1;
04           int n2 = 2;
05
06           System.out.println(n1+n2);
07           System.out.println(n1-n2);
08           System.out.println(n1*n2);
09           System.out.println(n1/n2);
10       }
11   }
```

```
3
-1
2
0
```

2. 실수 데이터형

실수형 예제2

- 정수와 정수의 연산 결과

→ Q) 다음 코드에서 이상한 점은?

→ A) 9행의 나누기 연산(1/2) 결괏값이 0.5가 아니라 0임
정수와 정수를 연산했기 때문에 소수점을 버린 0이 출력된 것

→ 이를 해결하려면 다음과 같이 4행의 int를 double로 변경해야 함

```
double n2 = 2;
```

```
3.0
-1.0
2.0
0.5
```

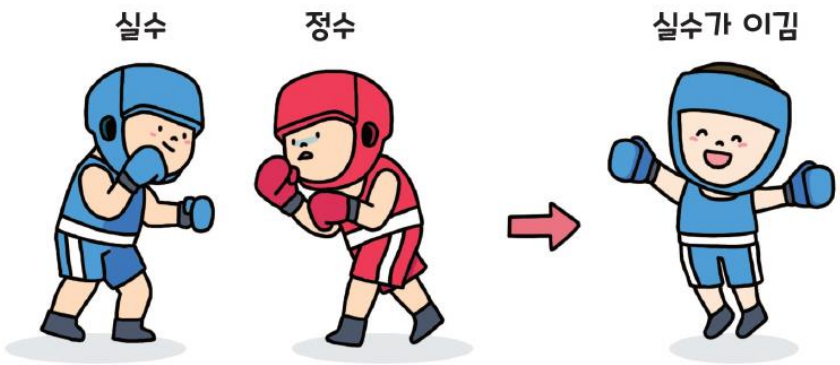


그림 4-5 정수와 실수의 연산 결과

3. 문자 데이터형

문자형

- 작은따옴표로 묶어서 대입함

표 4-3 문자 데이터형

문자형	의미	크기	값의 범위
char	문자	2바이트	영문 또는 한글 한 글자

```
char ch1 = 'A';
char ch2 = '난';
char ch3 = '5';
```

- char형에는 문자뿐만 아니라 값의 범위에 해당하는 정수를 대입할 수도 있음
  → 각 문자에는 고유한 숫자가 할당되어 있는데, 이를 아스키코드(ASCII code)라고 함

```
char ch4 = 100;
System.out.println(ch4);
```

```
d
```

표 4-4 아스키코드

아스키코드	10진수
0~9	48~57
A~Z	65~90
a~z	97~122

3. 문자 데이터형

[하나 더 알기] 유니코드 문자

영문과 숫자뿐만 아니라 한글, 중국어, 아랍어 등 다양한 언어도 문자에 포함됩니다. 유니코드(Unicode)는 이러한 수많은 종류의 문자를 표현하기 위한 것으로, 자바의 문자형인 char는 유니코드를 표현하기 위해 2바이트를 할당하므로 0~2^{16}-1(0~65535)의 범위를 표현할 수 있습니다.

예를 들어 '난'과 '生'의 유니코드 숫자는 다음과 같이 확인할 수 있습니다.

```
char ch1 = '난';
System.out.println(ch1 + "-->" + (int)ch1);
char ch2 = '生';
System.out.println(ch2 + "-->" + (int)ch2);
```

```
난-->45212
生-->29983
```

4. 불 데이터형

불형

- 논리형이라고도 부르며 참(true) 또는 거짓(false)만 저장할 수 있음

표 4-5 불 데이터형

불형	의미	크기	값의 범위
boolean	참 또는 거짓	1바이트	true 또는 false

- 불형 예제

코드 4-4 Code04_04.java

```
01   public class Code04_04 {
02       public static void main(String[] args) {
03           boolean boo1, boo2;
04
05           boo1 = true;
06           System.out.println(boo1);
07
08           boo2 = (100 < 10);
09           System.out.println(boo2);
10       }
11   }
```

```
true
false
```

4. 불 데이터형

확인문제

1. 다음 중 옳지 않은 것을 고르시오.

① 자바의 정수형에는 byte, short, int, long이 있다.
② 자바에서는 정수의 크기에 제한이 있다.
③ 자바의 기본 정수형은 short형이다.
④ byte형의 최댓값을 초과하면 최솟값인 −128로 바뀐다.

2. 다음 중 옳지 않은 것을 고르시오.

① 정수와 정수를 더하면 정수가 된다.
② 정수를 정수로 나누면 실수가 된다.
③ 실수와 실수를 더하면 실수가 된다.
④ 정수와 실수를 더하면 실수가 된다.

3. 다음 중 결과가 다른 것을 고르시오.

① `var1 = (30 > 300)`
② `var1 = (300 <= 300)`
③ `var1 = (30 <= 300)`
④ `var1 = (300 > 30)`

정답

Click!

5. 문자열 클래스

- 문자열
  - 한 문자가 여러 개 이어진 문자형의 집합을 의미함
  - 큰따옴표로 감사서 표현함
  - 자바는 문자열 데이터형을 따로 지원하지 않으며 String 클래스를 사용해야 함
  - 원칙적으로 문자열은 기본 데이터형이 아니지만 자주 사용되므로 데이터형에 포함하여 다룸

표 4-6 문자열 클래스

문자열 클래스	의미	크기	값의 범위
String	문자열	(입력된 문자 수×2)바이트	입력된 모든 문자

Section 03
System.out.printf( )
메서드의 서식 지정

1. System.out.printf( ) 사용법

- System.out의 대표적인 메서드
  - System.out.println()
    → 괄호 안의 내용을 출력한 후 행갈이 함
  - System.out.print()
    → 괄호 안의 내용을 출력한 후 행갈이를 하지 않음
  - System.out.printf()
    → 서식을 지정하여 출력할 수 있음

```
System.out.printf("Hello");
```

```
Hello
```

System.out.println()의 ln은 'line feed'의 약자이고, system.out.printf()의 f는 'format'의 약자입니다.

1. System.out.printf( ) 사용법

- System.out.printf()
  - 문자열 200 vs 숫자 200
    - → 큰따옴표 안에 들어 있는 것이 '문자'든 '숫자 형태의 문자'든 무조건 문자로 취급
    - → 서식(%d)이 지정된 숫자는 숫자 자체를 의미하며, %d는 정수형으로 출력

```
System.out.printf("200");
```

```
200
```

```
System.out.printf("%d", 200);
```

```
200
```

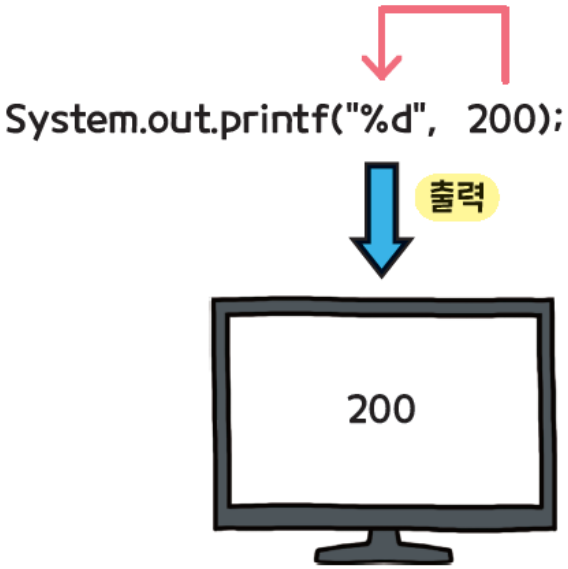


그림 4-6 서식과 숫자의 대응 1

1. System.out.printf( ) 사용법

- System.out.printf() 예제

코드 4-5 Code04_05.java

```
01   public class Code04_05 {
02       public static void main(String[] args) {
03           System.out.printf("200+300");
04           System.out.println();
05           System.out.printf("%d", 200+300);
06           System.out.println();
07       }
08   }
```

```
200+300
500
```

System.out.printf()는 출력 후 다음 행으로 넘어가지 않기 때문에 4행과 6행을 작성하여 다음 행으로 넘어가도록 했습니다. 3행과 4행을 합하여 다음과 같이 작성해도 동일한 결과를 얻을 수 있습니다.

```
System.out.printf("200+300 \n");
```

1. System.out.printf( ) 사용법

- 정수 서식 %d
  - %d는 정수(decimal)를 의미하며 이러한 것을 서식이라고 함
  - %d가 여러 개일 수도 있는데, 이럴 때는 짝이 맞아야 함

```
System.out.printf("%d", 200, 300);
```

```
200
```

```
System.out.printf("%d %d", 200);
```

```
200 Exception in thread "main" ~~~생략~~~
```

```
System.out.printf("%d %d", 200, 300);
```

```
200 300
```

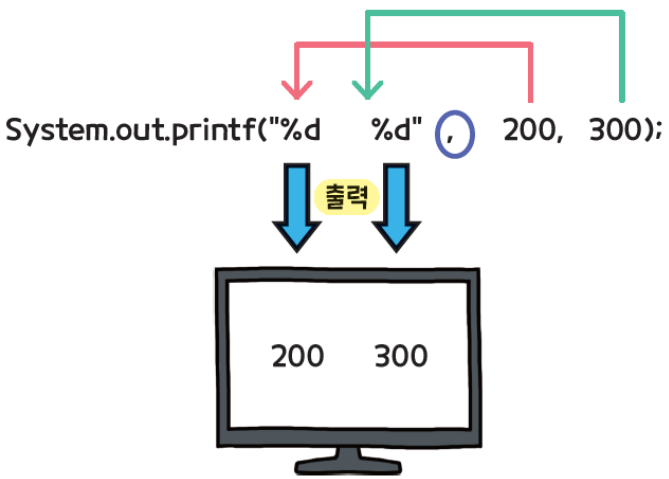


그림 4-7 서식과 숫자의 대응 2

2. 자주 사용되는 기타 서식

- 실수의 서식
  - 정수를 표현하는 %d를 사용하는 경우 오류 발생
  - 출력하는 결괏값에 따라 서식을 달리해야 함

코드 4-6 Code04_06.java

```
01   public class Code04_06 {
02       public static void main(String[] args) {
03           System.out.printf("%d / %d = %d", 100, 200, 0.5);
04       }
05   }
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> Code04_06 [Java Application] C:₩jdk₩bin₩javaw.exe  (2023. 2. 4. 오후 6:25:28 – 오후 6:25:28) [pid: 1624]
100 / 200 = Exception in thread "main" java.util.IllegalFormatConversionException: d != java.lang.Double
        at java.base/java.util.Formatter$FormatSpecifier.failConversion(Formatter.java:4442)
        at java.base/java.util.Formatter$FormatSpecifier.printInteger(Formatter.java:2963)
        at java.base/java.util.Formatter$FormatSpecifier.print(Formatter.java:2918)
        at java.base/java.util.Formatter.format(Formatter.java:2689)
        at java.base/java.io.PrintStream.format(PrintStream.java:1209)
        at java.base/java.io.PrintStream.printf(PrintStream.java:1105)
        at Code04_06.main(Code04_06.java:3)
```

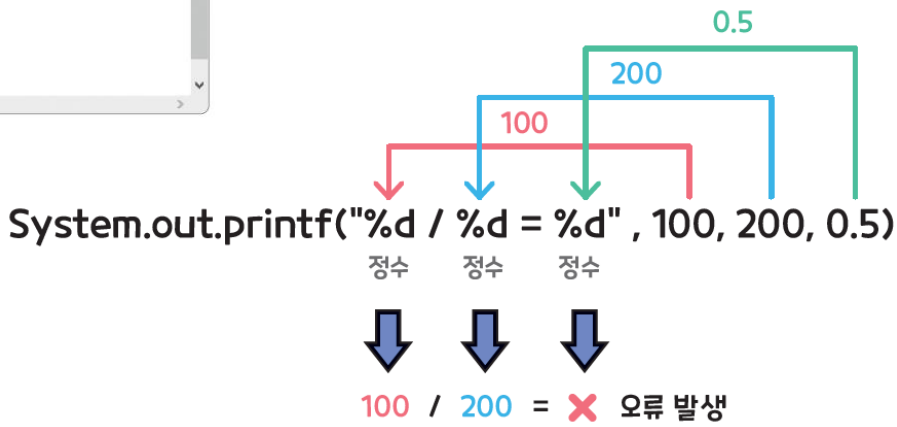


그림 4-8 서식과 숫자의 불일치

2. 자주 사용되는 기타 서식

실수의 서식

- System.out.printf()에서 사용할 수 있는 대표적인 서식

표 4-7 System.out.printf()의 대표 서식

서식	설명	예
%d, %x, %o	정수(10진수, 16진수, 8진수)	10, 100, 1234
%f	실수(소수점이 있는 수)	0.5, 1.0, 3.14
%c	한 개의 문자(작은따옴표로 묶음)	'a', 'b', 'F'
%s	한 개 이상의 문자로 이루어진 문자열(큰따옴표로 묶음)	"안녕", "abcdefg", "a"

- [코드 4-6]의 오류 수정

```
System.out.printf("%d / %d = %f", 100, 200, 0.5);
```

```
100 / 200 = 0.500000
```

- 0.5만 출력되도록 변경

```
System.out.printf("%d / %d = %3.1f", 100, 200, 0.5);
```

```
100 / 200 = 0.5
```

2. 자주 사용되는 기타 서식

- 정수의 자릿수를 넓히고 싶은 경우

```
System.out.printf("%d", 123);
System.out.println();
System.out.printf("%5d", 123);
System.out.println();
System.out.printf("%05d", 123);
```

```
123
  123
00123
```

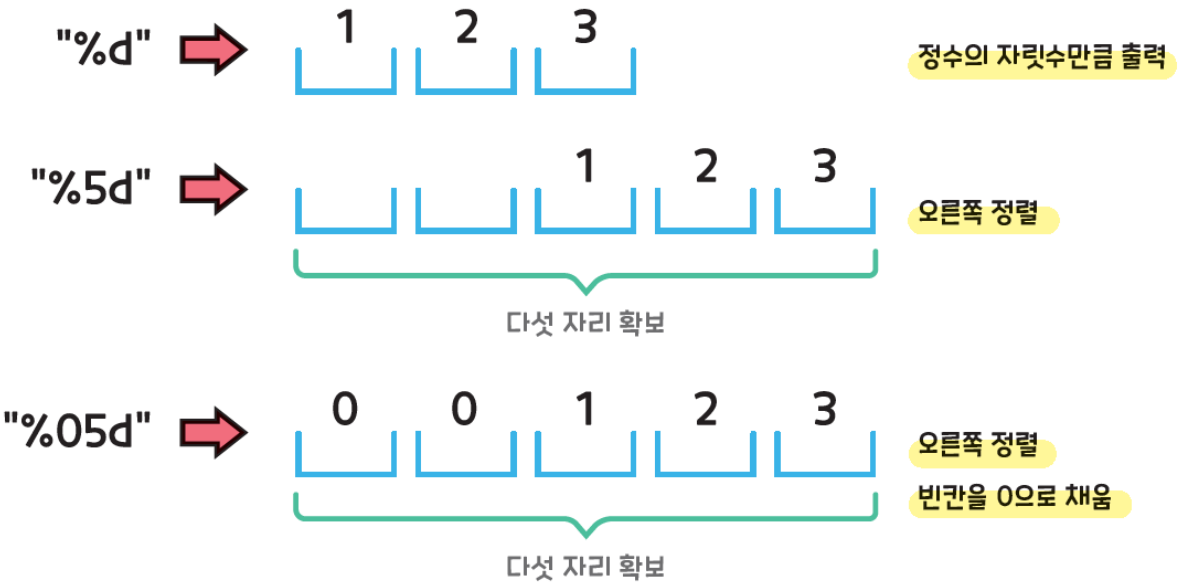


그림 4-9 정수형 데이터 서식 지정

2. 자주 사용되는 기타 서식

문자열 서식

문자열 서식 지정

```
System.out.printf("난생처음 자바");
System.out.println();
System.out.printf("%s", "난생처음 자바");
```

```
난생처음 자바
난생처음 자바
```

문자열 출력 자릿수 지정

```
System.out.printf("난생처음 자바 \n");
System.out.printf("%20s", "난생처음 자바 \n");
```

```
난생처음 자바
        난생처음 자바
```

2. 자주 사용되는 기타 서식

확인문제

1. 다음 중 오류가 발생하는 것을 고르시오.

① System.out.printf("%d", 30);

② System.out.printf("%d %d", 30, 40);

③ System.out.printf("%d %d", 30, 40, 50);

④ System.out.printf("%d %d %d", 30, 40, 50);

2. 다음 중 오류가 발생하는 것을 고르시오.

① System.out.printf("%d", 30);

② System.out.printf("%f", 3.14);

③ System.out.printf("%s", "자바");

④ System.out.printf("%g", 50);

정답

Click!

Section 04
문자열

1. 다양한 문자열

■ 문자열 안에 따옴표 표현하기

- 문자열은 큰따옴표로 묶어서 표현함

```
String var1 = "난생처음 자바";
String var2 = "난";
String var3 = "";
```

- 문자열 중간에 작은따옴표나 큰따옴표를 넣을 수도 있음
- 작은따옴표나 큰따옴표 앞에 백슬래시(\)를 붙이기

  → 백슬래시 다음의 작은따옴표나 큰따옴표가 문자로 인식됨

코드 4-7 Code04_07.java

```
01  public class Code04_07 {
02      public static void main(String[] args) {
03          String var1 = "작은따옴표는 \' 모양입니다.";
04          String var2 = "큰따옴표는 \" 모양입니다.";
05
06          System.out.println(var1);
07          System.out.println(var2);
08      }
09  }
```

```
작은따옴표는 ' 모양입니다.
큰따옴표는 " 모양입니다.
```

1. 다양한 문자열

- 여러 행의 문자열 만들기
  - 더하기(+) 연산자로 연결하면 여러 행을 문자열로 만들 수 있음
  - \n을 사용하면 행갈이됨

코드 4-8 Code04_08.java

```
01  public class Code04_08 {
02      public static void main(String[] args) {
03          String var1 = "난생처음 \n" +
04                        "자바를 \n" +
05                        "열공 중입니다.";
06          System.out.println(var1);
07      }
08  }
```

```
난생처음
자바를
열공 중입니다.
```

2. 이스케이프 문자를 사용한 문자열

- 이스케이프(escape) 문자
  - 서식 문자라고도 함
  - 이스케이프 문자는 앞에 백슬래시를 붙이는 것이 특징임

표 4-8 이스케이프 문자

이스케이프 문자	설명
\n	다음 행으로 이동(Enter와 같은 효과)
\t	다음 탭으로 이동(Tab과 같은 효과)
\b	뒤로 한 칸 이동(Backspace와 같은 효과)
\'	' 출력
\"	" 출력
\\	\ 출력

이클립스에서 \b를 사용하면 깨진 글자로 보일 수 있습니다.

2. 이스케이프 문자를 사용한 문자열

- 이스케이프(escape) 문자
  - 이스케이프 문자 예제

코드 4-9 Code04_09.java

```
01   public class Code04_09 {
02       public static void main(String[] args) {
03           System.out.println("\n줄바꿈\n연습 ");
04           System.out.println("\t탭키\t연습");
05           System.out.println("어떤 글자를 \"강조\"하는 효과1");
06           System.out.println("어떤 글자를 \'강조\'하는 효과2");
07           System.out.println("\\\\ 백슬래시 2개 출력");
08       }
09   }
```

```
줄바꿈
연습
        탭키    연습
어떤 글자를 "강조"하는 효과1
어떤 글자를 '강조'하는 효과2
\\ 백슬래시 2개 출력
```

2. 이스케이프 문자를 사용한 문자열

확인문제

다음 중 이스케이프 문자가 아닌 것을 고르시오.

① \n ② \t ③ \u ④ \\

정답

Click!

3. 문자열 연결

문자열 연결

- 더하기(+) 연산자 사용

```
String var1 = "난생" + "처음" + "자바";
System.out.println(var1);
```

```
난생처음자바
```

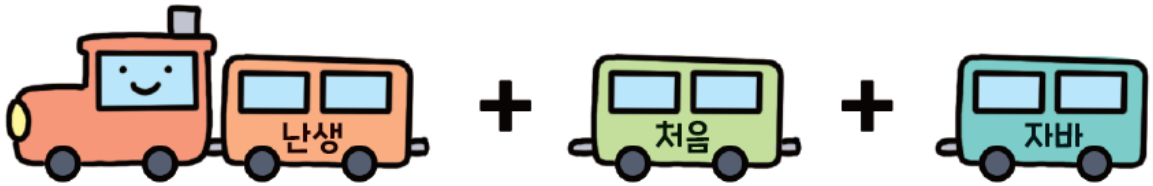


그림 4-10 문자열을 연결하는 방법

- 여러 행에 걸쳐 연결하는 경우

```
String var1 = "난생";
var1 = var1 + "처음";
var1 += "자바";
System.out.println(var1);
```

```
난생처음자바
```

3. 문자열 연결

확인문제

다음 중 옳지 않은 것을 고르시오.

① 문자열에 숫자를 더하면 오류가 발생한다.
② 문자열에서 문자열을 빼면 오류가 발생한다.
③ 문자열에서 숫자를 빼면 오류가 발생한다.
④ 문자열에 숫자를 곱하면 오류가 발생한다.

정답

Click!

4. 문자열 관련 메서드

String 클래스

- 문자열 관련 메서드를 사용하려면 먼저 String 클래스 유형의 변수를 선언해야 함

```
String 문자열_변수;
문자열변수.문자열_메서드();
```

length( )

- 문자열의 길이를 확인할 때 사용하는 메서드

코드 4-10 Code04_10.java

```
01   public class Code04_10 {
02       public static void main(String[] args) {
03           String str = "난생처음 자바";
04           int len;
05
06           len = str.length();
07
08           System.out.println("문자열 : " + str);
09           System.out.println("문자열 길이 : " + len);
10       }
11   }
```

```
문자열 : 난생처음 자바
문자열 길이 : 8
```

4. 문자열 관련 메서드

length( )

- 두 개의 문자열을 입력받아 어떤 문자열이 얼마나 더 긴지 출력하는 예제

코드 4-11 Code04_11.java

```
01  import java.util.Scanner;
02  public class Code04_11 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          String var1, var2;
06          int diff;
07
08          System.out.print("첫 번째 문자열 ==> ");
09          var1 = s.nextLine();
10          System.out.print("두 번째 문자열 ==> ");
11          var2 = s.nextLine();
12
13          diff = var1.length() - var2.length();
14          System.out.println("두 문자열 길이의 차이는 " + diff + "입니다." );
15          s.close();
16      }
17  }
```

```
첫 번째 문자열 ==> 난생처음 자바
두 번째 문자열 ==> First Java
두 문자열 길이의 차이는 -3입니다.
```

사용자가 입력

4. 문자열 관련 메서드

- toUpperCase( ), toLowerCase( )
  - toUpperCase( ): 영문 소문자를 대문자로 변환하는 메서드
  - toLowerCase( ): 대문자를 소문자로 변환하는 메서드
  - 영문을 제외한 한글, 숫자, 기호 등은 toUpperCase( ), toLowerCase() 메서드의 영향을 받지 않음

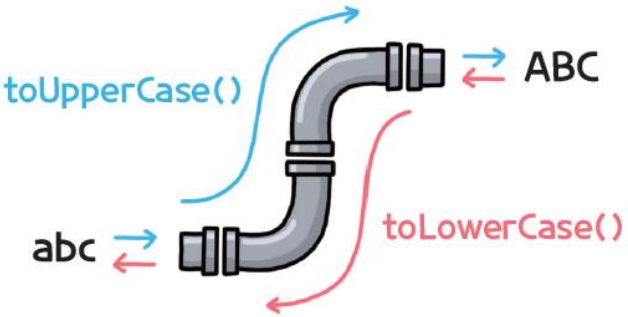


그림 4-11 대·소문자 변환 메서드

4. 문자열 관련 메서드

- toUpperCase( ), toLowerCase( )
  - 대소문자 변경 예제

코드 4-12 Code04_12.java

```
01   public class Code04_12 {
02       public static void main(String[] args) {
03           String ss = "First Java를 공부 중입니다.";
04           String var1, var2;
05
06           var1 = ss.toUpperCase();
07           System.out.println(var1);
08           var2 = ss.toLowerCase();
09           System.out.println(var2);
10       }
11   }
```

```
FIRST JAVA를 공부 중입니다.
first Java를 공부 중입니다.
```

4. 문자열 관련 메서드

- trim()
  - 문자열 앞뒤의 공백을 잘라내는 메서드

코드 4-13 Code04_13.java

```
01   public class Code04_13 {
02       public static void main(String[] args) {
03           String str = "   한글   ABCD  efgh  ";
04           String cutStr = "";
05
06           cutStr = str.trim();
07
08           System.out.println("기존 문자열 ==> [" + str + "]");
09           System.out.println("공백 제거  ==> [" + cutStr + "]");
10       }
11   }
```

```
기존 문자열 ==> [   한글   ABCD  efgh  ]
공백 제거  ==> [한글   ABCD  efgh]
```

4. 문자열 관련 메서드

- replaceAll(기존문자열, 새문자열)
  - trim()은 문자열 중간의 공백을 잘라내지는 않음
  - 따라서 문자열 전체의 공백을 제거하려면 replaceAll() 메서드를 사용해야 함

```
cutStr = str.replaceAll(" ", "");
```

```
기존 문자열 ==> [   한글   ABCD  efgh  ]
공백 제거  ==> [한글ABCDefgh]
```

[하나 더 알기] replaceAll()의 활용

replaceAll(기존문자열, 새문자열)은 공백만 제거하는 메서드가 아니라 특정 문자열을 다른 문자열로 대체하는 메서드입니다. 다시 말해 replaceAll(기존문자열, 새문자열)을 사용하면 어떤 문자열이든 원하는 문자열로 바꿀 수 있습니다. 예를 들어 replaceAll("난생", "First")는 "난생"이라는 글자를 모두 "First"로 바꿉니다.

4. 문자열 관련 메서드

- indexOf(찾을문자열)
  - 어떤 글자가 문자열의 몇 번째에 위치하는지 찾는 메서드
  - 문자열의 위치가 0번부터 시작됨
  - 공백도 하나의 문자이므로 문자열의 순서에 포함됨

그림 4-12 문자열의 순서

코드 4-14 Code04_14.java

```
01  public class Code04_14 {
02      public static void main(String[] args) {
03          String str = "난생처음 자바";
04
05          System.out.println(str.indexOf("난생"));
06          System.out.println(str.indexOf("처"));
07      }
08  }
```

```
0
2
```

4. 문자열 관련 메서드

- indexOf(찾을문자열, 시작위치)
  - 똑같은 문자가 여러 개 있을 때는 원하는 문자의 위치를 찾는 메서드
  - 찾는 문자열이 시작위치에서 몇 번째에 있는지 알려줌

코드 4-15 Code04_15.java

```
01  public class Code04_15 {
02      public static void main(String[] args) {
03          String str = "난생처음 자바를 처음 학습 중입니다. 자바는 처음이지만 재미있네요.";
04
05          System.out.println(str.indexOf("처음"));
06          System.out.println(str.indexOf("처음"));
07          System.out.println(str.indexOf("처음", 4));
08      }
09  }
```

```
2
2
9
```

4. 문자열 관련 메서드

- charAt(위치)
  - 문자열의 각 문자에 접근하는 메서드
  - 문자열은 여러 개의 문자로 이루어져 있기 때문에 각 문자마다 위치가 지정되어 있음

→ 문자열의 길이와 순번은 다르기 때문에 주의해야 함

```
String str = "Java";
System.out.println(str.length());
```

```
4
```

그림 4-13 문자열의 순번

→ 문자열의 길이는 4이지만, 4번 문자는 존재하지 않음

코드 4-16 Code04_16.java

```
01  public class Code04_16 {
02      public static void main(String[] args) {
03          String str = "Java";
04
05          System.out.println(str.charAt(0));
06          System.out.println(str.charAt(1));
07          System.out.println(str.charAt(2));
08          System.out.println(str.charAt(3));
09          System.out.println(str.charAt(4));
10      }
11  }
```

```
J
a
v
a
오류 발생
```

4. 문자열 관련 메서드

- substring(시작위치,끝위치)
  - charAt()로 얻은 결과는 하나의 문자를 의미하는 char형 문자임

그림 4-14 문자열을 문자로 변경하는 charAt()

  - 만약 추출한 하나의 문자를 문자열로 취급하고 싶다면 substring()을 사용해야 함
  - 이때 끝 위치의 문자가 포함되지 않음

```
String str = "Java";
System.out.println(str.substring(0,1))
```

```
J
```

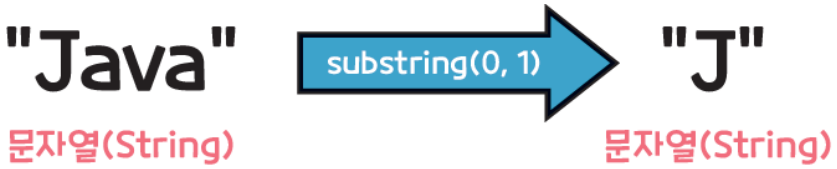


그림 4-15 문자열을 문자열로 추출하는 substring()

4. 문자열 관련 메서드

- substring(시작위치,끝위치)
  - 여러 문자를 추출할 수도 있음

```
String str = "Java";
System.out.println(str.substring(2,4));
```

```
va
```

4. 문자열 관련 메서드

[하나 더 알기] 배열 맛보기

배열(array)은 자바에서 가장 중요한 데이터형입니다. 9장에서 자세히 다룰 테지만, 문자열을 문자 배열로 변환하여 사용하는 경우가 많기 때문에 미리 간단히 살펴보겠습니다.
배열은 여러 개의 값을 하나로 묶어놓은 꾸러미라고 볼 수 있습니다. 다음은 'J', 'a', 'v', 'a'라는 네 개의 값을 하나의 배열로 묶은 것입니다.


```
char[] arr = {'J', 'a', 'v', 'a'};
```

arr의 개수를 구하려면 length 속성을 사용하며, 다음 코드의 경우 4가 출력됩니다. 메서드와 달리 속성에는 괄호가 붙지 않습니다.

```
System.out.println(arr.length);
```

arr[위치]를 사용하면 각 값에 접근할 수 있습니다. 문자열과 마찬가지로 배열의 위치도 0번부터 시작하므로 'v'를 출력하려면 다음과 같이 작성해야 합니다.

```
System.out.println(arr[2]);
```

또한 문자열을 배열로 변환하려면 toCharArray() 메서드를 사용합니다

```
String str = "Java";
char[] arr = str.toCharArray();
```

4. 문자열 관련 메서드

확인문제

다음 빈칸에 알맞은 말을 넣으시오.

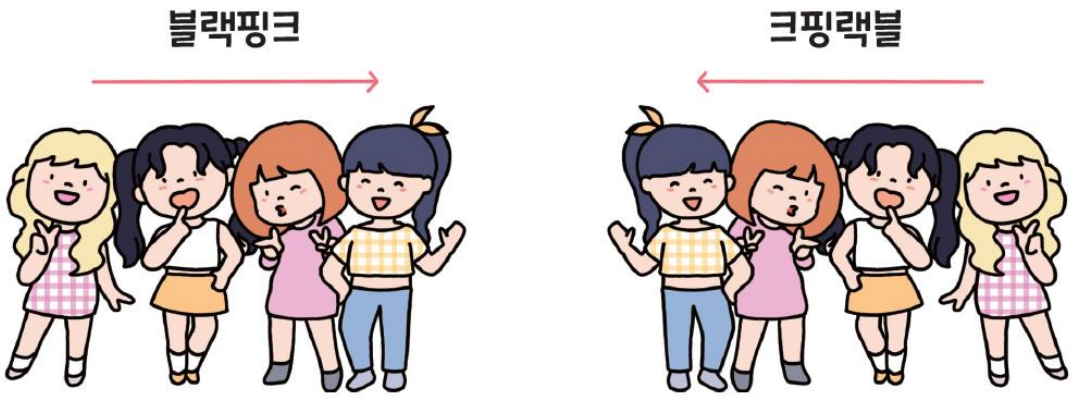
- 문자열의 길이를 구하는 메서드는 [ㄱ] 이다.
- 문자열을 대문자로 변경하는 메서드는 [ㄴ] 이고, 소문자로 변경하는 메서드는 [ㄷ] 이다.
- 문자열의 앞뒤 공백을 제거하는 메서드는 [ㄹ] 이고, 모든 공백을 제거하는 메서드는 [ㅁ] 이다.
- 어떤 글자의 위치를 찾는 메서드는 [ㅂ] 이다.
- 문자열의 길이가 5라면 각 문자의 위치에 [ㅅ] 번부터 [ㅇ] 번이 할당된다.

정답

Click!

[LAB] 문자열 거꾸로 출력하기

"블랙핑크"라는 문자열을 거꾸로 출력하는 프로그램을 만들어봅시다



실행 결과

```
원본 문자열 ==> 블랙핑크
반대 문자열 ==> 크핑랙블
```

[LAB] 문자열 거꾸로 출력하기

1. Lab04_01.java 파일을 만들고, 변수 ss에 "블랙핑크"라는 문자열을 저장하여 출력하기

```
String ss = "블랙핑크";
System.out.println("원본 문자열 ==> " + ss);
```

2. 거꾸로 출력될 문자열을 표시하기

```
System.out.print("반대 문자열 ==> ");
```

3. 문자열의 길이가 4이므로 마지막 글자가 ss.charAt(3)이며, ss.charAt(3)부터 출력하기
   - 3번부터 0번까지 글자를 차례대로 출력하면 문자열을 거꾸로 출력하는 것과 마찬가지임

```
System.out.print(ss.charAt(3));
System.out.print(ss.charAt(2));
System.out.print(ss.charAt(1));
System.out.print(ss.charAt(0));
```

[LAB] 대·소문자 변환하기

대문자는 소문자로, 소문자는 대문자로 변환하는 프로그램을 만들어봅시다

실행 결과

```
원본 문자열 ==>Java
변환 문자열 ==>jAVA
```

지금까지 배운 내용만으로 프로그램을 만들어야 해서 상당히 비효율적일 수밖에 없지만, 6장까지 공부하고 나면 훨씬 효율적으로 코딩할 수 있습니다. 여기서는 대·소문자 변환에만 초점을 맞추겠습니다.

[LAB] 대·소문자 변환하기

1. Lab04_02.java 파일을 만들고, 변수 ss에 "Java"라는 문자열을 저장하여 출력하기. 그리고 대·소문자를 변경한 문자를 저장할 빈 문자열 변수 newSS를 선언하기

```
String ss = "Java";
System.out.println("원본 문자열 ==> " + ss);
String newSS = "";
```

2. 첫 번째 글자는 대문자이므로 toLowerCase() 메서드를 사용하여 소문자로 변경하기

```
newSS = ss.substring(0,1).toLowerCase();
```

3. 두 번째부터 네 번째 글자는 소문자이므로 toUpperCase() 메서드를 사용하여 대문자로 변경한 후 newSS에 연결하기

```
newSS += ss.substring(1,2).toUpperCase();
newSS += ss.substring(2,3).toUpperCase();
newSS += ss.substring(3,4).toUpperCase();
```

4. 결국 newSS에는 대문자인 첫 번째 글자가 소문자로 저장되고, 소문자인 나머지 글자가 대문자로 저장됨

```
System.out.print("변환 문자열 ==> ");
System.out.print(newSS);
```

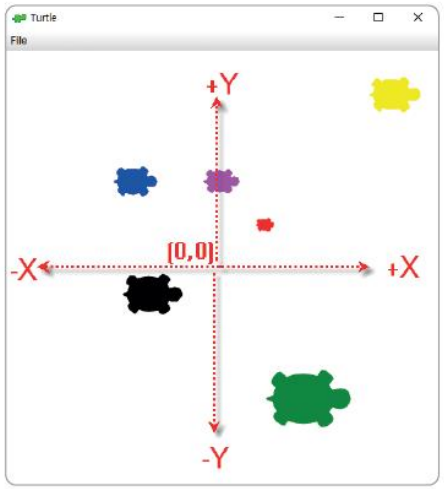
[실전예제]
여행지에서
스탬프 찍는 거북이

[실전 예제] 여행지에서 스탬프 찍는 거북이

[문제]

거북이가 여행을 다니면서 각 여행지에서 스탬프를 찍는 프로그램을 작성해봅시다.

사용자가 X와 Y 위치, 스탬프의 크기와 색상을 입력하면 거북이가 해당 위치에서 스탬프를 찍으며, 이때 거북이는 화면의 중앙에서 출발합니다.



```
X 위치(-200 ~ +200) ==> 50
Y 위치(-200 ~ +200) ==> 50
스탬프 크기(1~100) ==> 20
스탬프 색상(red, green, blue, black, yellow) ==> red
X 위치(-200 ~ +200) ==> -100
Y 위치(-200 ~ +200) ==> 100
스탬프 크기(1~100) ==> 50
스탬프 색상(red, green, blue, black, yellow) ==> blue
```

[실전 예제] 여행지에서 스탬프 찍는 거북이

[해결]

Ex04_01.java

```
01  import java.util.Scanner;
02  public class Ex04_01 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          Turtle turtle = new Turtle();
06
07          double x, y;
08          int tsize;
09          String color;
10          turtle.up();
11
12          while(true) {
13              System.out.print("X 위치(-200 ~ +200) ==> ");
14              x = s.nextDouble();
15              System.out.print("Y 위치(-200 ~ +200) ==> ");
16              y = s.nextDouble();
17              System.out.print("스탬프 크기(1~100) ==> ");
18              tsize = s.nextInt();
19              System.out.print("스탬프 색상(red, green, blue, black, yellow) ==> ");
20              color = s.next();
21
22              turtle.setPosition(x, y);
23              turtle.shapeSize(tsize, tsize);
24              turtle.outlineColor(color);
25              turtle.fillColor(color);
26              turtle.stamp();
27          }
28      }
29  }
```

Thank you!