

컴퓨터 프로그래밍 (Computer Programming)

이 선 순



4. 자료형 변환



목차

1. 자동타입변환
2. 강제타입변환
3. 정수연산에서 자동타입변환
4. 실수연산에서 자동타입변환
- 5.+연산에서 문자열 자동타입변환
6. 문자열을 기본타입으로 강제타입변환

01

자동타입변환

01 자동타입변환

■ 자료형 변환 또는 타입변환

- 두 변수의 타입이 동일할 경우, 한쪽 변수 값을 다른 쪽 변수에 복사해서 저장할 수 있음
- 만약 두 변수의 타입이 다르다면 어떻게 될까? 값의 저장이 가능할 수도 있고 그렇지 않을 수도 있다.
- a 변수에 저장된 값을 b변수에 복사해서 저장하는 코드는 다음과 같음

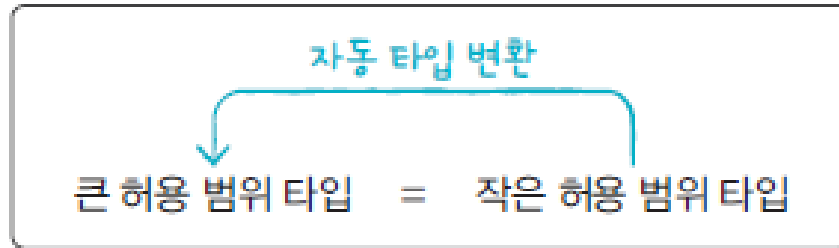
```
byte a = 10;    //byte 타입 변수 a에 10을 저장
int b = a;      //byte 타입 변수 a에 저장된 10을 int 타입 변수 b에 복사해서 저장
```

- 원래 10은 byte 타입의 값이었으나, 복사해서 저장할 때 int 타입의 값으로 변환됨. ⇒ 타입변환
- 데이터 타입을 다른 데이터 타입으로 변환하는 것을 말한다.
- byte 타입 → int타입
- int 타입 → byte 타입
- double 타입 → int 타입
- string 타입 → int 타입
- 변수 값을 다른 타입의 변수에 저장할 때 타입변환이 이루어짐

01 자동타입변환

■ 자동 타입 변환 (promotion)

- 자동으로 타입 변환이 일어나는 것을 의미함
- 자동타입변환은 값의 허용 범위가 작은 타입이 큰 타입으로 저장될 경우에 발생함
- 자동타입변환은 프로그램 실행 도중에 자동으로 타입변환이 일어남



- 기본 타입의 허용 범위 순

01 자동타입변환

■ 자동 타입 변환 (promotion)

- 기본 타입의 허용 범위 순

```
byte < short < int < long < float < double
```

- int 타입이 byte 타입보다 허용범위가 더 크기 때문에 자동타입 변환 발생

```
byte byteValue = 10;  
int intValue = byteValue;    //자동 타입 변환됨
```

- 정수타입이 실수타입으로 저장될 경우에는 무조건 자동타입변환이 일어남. 실수 타입은 정수 타입보다 허용범위가 더 크기 때문임

```
long longValue = 5000000000L;  
float floatValue = longValue;    //5.0E9f로 저장됨  
double doubleValue = longValue;  //5.0E9로 저장됨
```

01 자동타입변환

■ 자동 타입 변환 (promotion)

- char 타입의 경우 int 타입으로 자동변환되면 유니코드 값이 int 타입에 저장

```
char charValue = 'A';  
int intValue = charValue;    //65가 저장됨
```

■ 자동타입변환에서의 예외

- char 타입보다 허용 범위가 작은 byte 타입은 char 타입으로 자동타입 변환될 수 없음 ⇒ char 타입의 허용범위는 음수를 포함하지 않는데, byte 타입은 음수를 포함하기 때문임

```
byte byteValue = 65;  
char charValue = byteValue; ← 컴파일 에러
```


01 자동타입변환

■ 자동 타입 변환 (promotion)

■ 예

```
2
3 public class PromotionExample {
4     public static void main(String[] args) {
5         //자동 타입 변환
6         byte byteValue = 10;
7         int intValue = byteValue;
8         System.out.println("intValue: " + intValue);
9
10        char charValue = '가';
11        intValue = charValue;
12        System.out.println("가의 유니코드: " + intValue);
13
14        intValue = 50;
15        long longValue = intValue;
16        System.out.println("longValue: " + longValue);
17
18        longValue = 100;
19        float floatValue = longValue;
20        System.out.println("floatValue: " + floatValue);
21
22        floatValue = 100.5F;
23        double doubleValue = floatValue;
24        System.out.println("doubleValue: " + doubleValue);
25    }
26 }
27
```

Problems @ Javadoc Declaration Console

<terminated> PromotionExample [Java Application] C:₩

intValue: 10
가의 유니코드: 44032
longValue: 50
floatValue: 100.0
doubleValue: 100.5

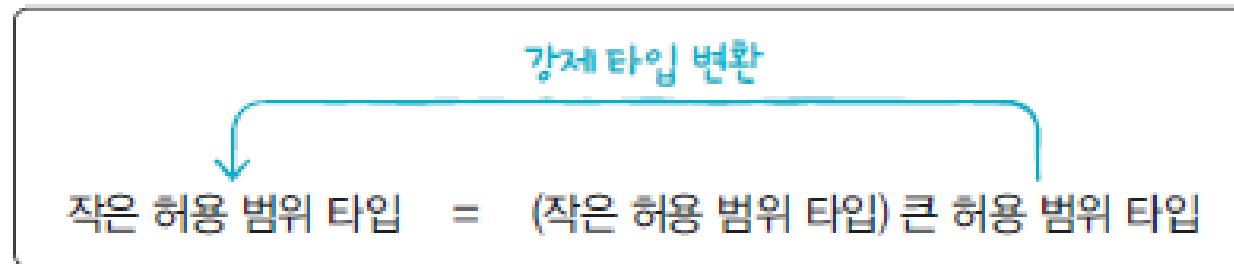
02

강제타입변환

02 강제타입변환

■ 강제 타입 변환 (casting)

- 큰 허용 범위 타입은 작은 허용 범위 타입으로 자동 타입 변환될 수 없음
- 큰 허용 범위 타입을 작은 허용 범위 타입으로 강제로 나누어 한 조각만 저장
- 강제 타입 변환은 큰 타입을 작은 타입으로 강제로 나눠서 저장하는 것을 말함
- 캐스팅 연산자 괄호 () 사용: 괄호 안에 들어가는 타입은 나누는 단위



- 예. int 타입은 byte 타입보다 더 큰 허용범위를 가짐. 따라서 int 타입은 byte 타입으로 자동변환되지 않음.
 - (byte) 캐스팅 연산자를 사용해서 byte 타입으로 강제 변환할 수 있음

```
int intValue = 10;  
byte byteValue = (byte) intValue;    //강제 타입 변환
```

02 강제타입변환

■ 강제 타입 변환 (casting)

- 예. int 타입은 char 타입보다 더 큰 허용범위를 가짐. 따라서 int 타입은 char 타입으로 자동 변환되지 않음

- (char) 캐스팅 연산자를 사용해서 char 타입으로 강제 변환할 수 있음
- 이유 : 문자출력을 위함

```
int intValue = 65;  
char charValue = (char) intValue;  
System.out.println(charValue);    //"A"가 출력
```

- 예. 실수 타입(float, double)은 정수 타입(byte, short, int, long)으로 자동 변환되지 않기 때문에 강제타입변환을 사용해야 함.

- 소수점 이하 부분 버려지고 정수 부분만 저장

```
double doubleValue = 3.14;  
int intValue = (int) doubleValue;    //intValue는 정수 부분인 3만 저장
```

02 강제타입변환

■ 예.

```
2
3 public class CastingExample {
4     public static void main(String[] args) {
5         int intValue = 44032;
6         char charValue = (char) intValue;
7         System.out.println(charValue);
8
9         long longValue = 500;
10        intValue = (int) longValue;
11        System.out.println(intValue);
12
13        double doubleValue = 3.14;
14        intValue = (int) doubleValue;
15        System.out.println(intValue);
16    }
17 }
18
```

Problems @ Javadoc Declaration Console

<terminated> CastingExample [Java Application] C:\Program Files\Java\

가
500
3

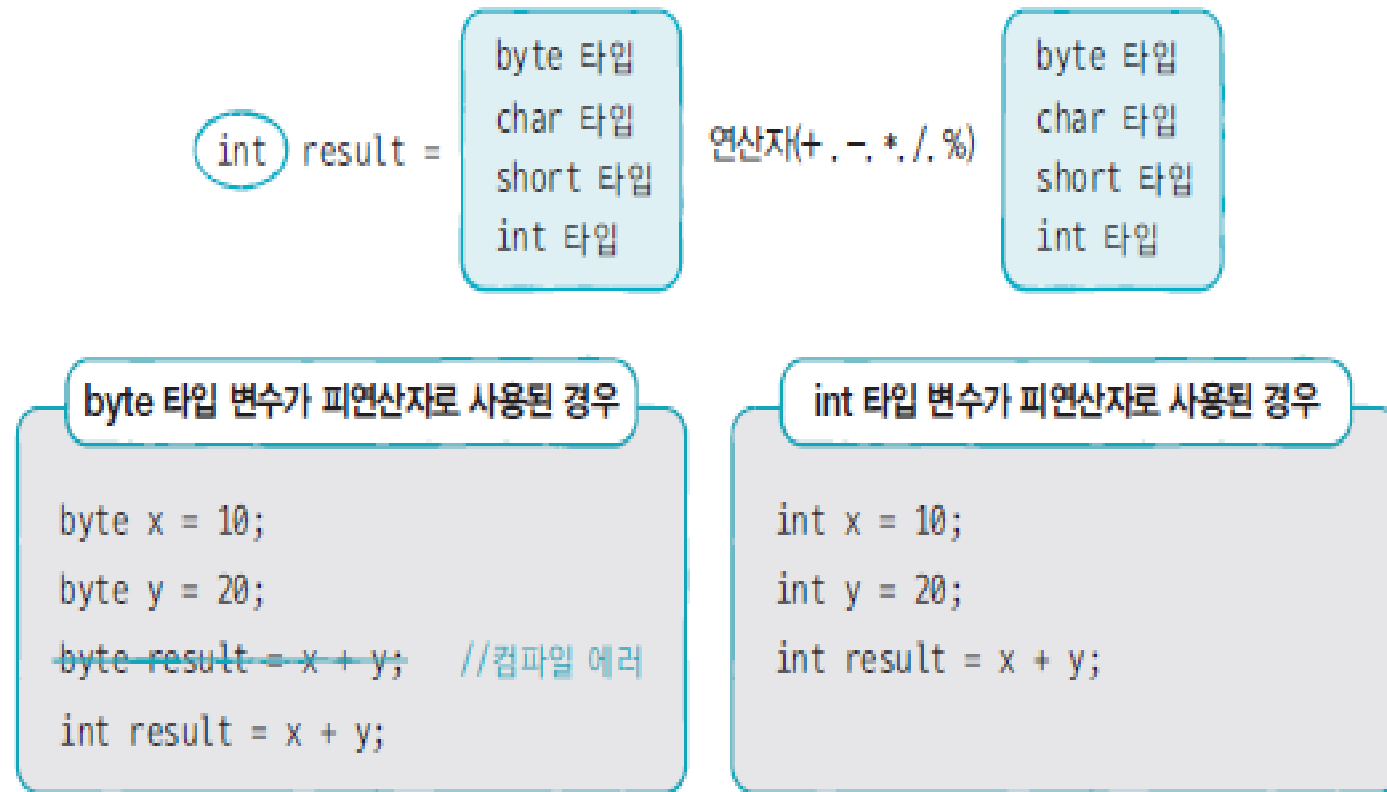
03

정수연산에서의 자동타입변환

03 정수연산에서의 자동타입변환

■ 정수 타입 변수가 산술 연산식에서 피연산자로 사용되는 경우

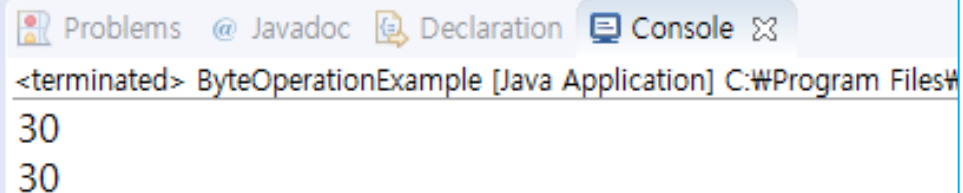
- 정수타입변수가 산술 연산식에서 피연산자로 사용되면 int타입보다 작은 byte, short 타입의 변수는 int 타입으로 자동 타입변환되어 연산을 수행함



03 정수연산에서 자동타입변환

- 예. 정수타입의 연산

```
2
3 public class ByteOperationExample {
4     public static void main(String[] args) {
5         byte result1 = 10 + 20;
6         System.out.println(result1);
7
8         byte x = 10;
9         byte y = 20;
10        int result2 = x + y;
11        System.out.println(result2);
12    }
13 }
14
```



Problems Javadoc Declaration Console

<terminated> ByteOperationExample [Java Application] C:\Program Files\

30

30

03 정수연산에서의 자동타입변환

■ 정수 타입 변수가 산술 연산식에서 피연산자로 사용되는 경우

- 특별한 경우 아니라면 정수 연산에 사용하는 변수는 int 타입으로 선언하는 것이 효과적
- 피 연산자 중 하나가 long 타입이면 다른 피연산자는 long 타입으로 자동 변환

`long` result = `long` 타입 연산자(+, -, *, /, %)

byte 타입
char 타입
short 타입
int 타입

03 정수연산에서 자동타입변환

- 예. 정수타입의 연산

```
2  
3 public class LongOperationExample {  
4     public static void main(String[] args) {  
5         byte value1 = 10;  
6         int value2 = 100;  
7         long value3 = 1000L;  
8         long result = value1 + value2 + value3;  
9         System.out.println(result);  
10    }  
11 }  
12
```

Problems Javadoc Declaration Console

<terminated> LongOperationExample [Java Application] C:\WP
1110

04

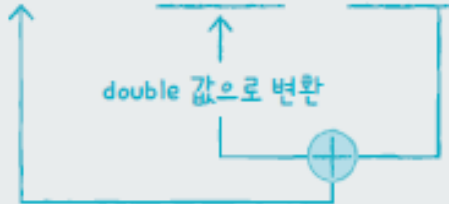
실수연산에서의 자동타입변환

04 실수연산에서 자동타입변환

■ 피연산자 중 하나가 double 타입일 경우 다른 피연산자도 double 타입으로 자동변환

- int 타입과 double 타입 연산 : int 타입의 피연산자가 double 타입으로 자동변환되고 연산을 수행함

```
int intValue = 10;  
double doubleValue = 5.5;  
double result = intValue + doubleValue;    //result에 15.5가 저장됨
```



■ 다른 타입 연산이 필요할 경우 먼저 강제 변환한 뒤 연산 수행

- 만약 꼭 int 타입으로 연산을 해야한다면 double 타입을 int 타입으로 강제 변환하고 덧셈 연산을 수행하면 됨

```
int intValue = 10;  
double doubleValue = 5.5;  
int result = intValue + (int) doubleValue;    //result에 15가 저장됨
```

04 실수연산에서 자동타입변환

■ 실수 리터럴 연산

- 자바에서는 소문자 `f` 또는 대문자 `F`가 있는 실수 리터럴을 `double` 타입으로 해석됨. 그렇기때문에 연산 결과는 `double` 타입변수에 저장해야함
 - `double result = 1.5 + 2.3;`
- `Float` 타입 변수에 저장하면 컴파일 에러가 발생함
 - ~~`float result = 1.5 + 2.3`~~
- `Float` 타입에 꼭 저장하고 싶다면 실수 리터럴 뒤에 소문자 `f` 나 대문자 `F` 를 붙여 컴파일러가 `float` 타입을 알도록 해야함
 - `float result = 1.5f + 2.3f;`

04 실수연산에서 자동타입변환

■ 정수 연산의 결과를 실수로 저장할 때 주의할 점

■ 정수 연산의 결과는 정수

```
int x = 1;
int y = 2;
double result = x / y;
System.out.println(result);
```

- 위 코드를 실행하면 0.5가 출력되는 것이 아니라 0.0이 출력됨 : 자바에서 정수연산의 결과는 정수가 됨
- x/y 의 연산 결과는 0.5 가 아니라 0 이 되고, 0을 double 타입 변수 result에 저장하므로 0.0 이 됨
- 위 코드의 결과가 0.0 이 아니라 0.5가 되게 하려면 x/y 를 정수 연산이 아니라 실수연산으로 변경해야 함 $\Rightarrow x, y$ 둘 중 하나 또는 둘 모두를 double 타입으로 변환해야 함

```
float floatValue = (float) 정수;
double doubleValue = (double) 정수 ;
```

04 실수연산에서 자동타입변환

■ 정수 연산의 결과를 실수로 저장할 때 주의할 점

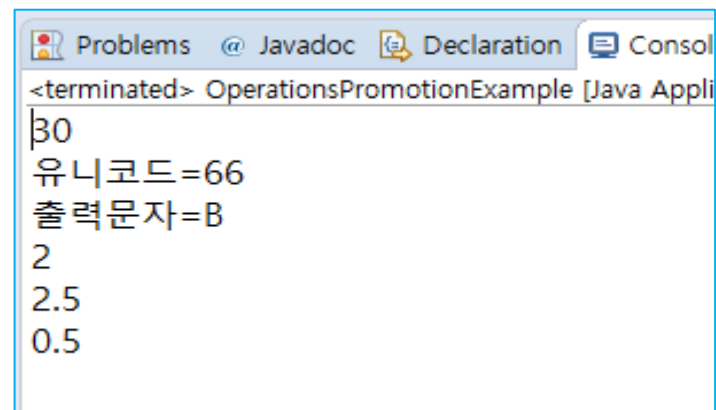
- 실수 결과를 얻으려면 실수 연산으로의 변환 필요

방법 1	<pre>int x = 1; int y = 2; double result = (double) x / y; System.out.println(result);</pre>
방법 2	<pre>int x = 1; int y = 2; double result = x / (double) y; System.out.println(result);</pre>
방법 3	<pre>int x = 1; int y = 2; double result = (double) x / (double) y; System.out.println(result);</pre>

04 실수연산에서 자동타입변환

■ 예. 연산식에서 자동타입변환

```
2
3 public class OperationsPromotionExample {
4     public static void main(String[] args) {
5         byte byteValue1 = 10;
6         byte byteValue2 = 20;
7         //byte byteValue3 = byteValue1 + byteValue2; //컴파일 에러
8         int intValue1 = byteValue1 + byteValue2;
9         System.out.println(intValue1);
10
11         char charValue1 = 'A';
12         char charValue2 = 1;
13         //char charValue3 = charValue1 + charValue2; //컴파일 에러
14         int intValue2 = charValue1 + charValue2;
15         System.out.println("유니코드=" + intValue2);
16         System.out.println("출력문자=" + (char)intValue2);
17
18         int intValue3 = 10;
19         int intValue4 = intValue3/4;
20         System.out.println(intValue4);
21
22         int intValue5 = 10;
23         //int intValue6 = 10 / 4.0; //컴파일 에러
24         double doubleValue = intValue5 / 4.0;
25         System.out.println(doubleValue);
26
27         int x = 1;
28         int y = 2;
29         double result = (double) x / y;
30         System.out.println(result);
31     }
32 }
33
```



```
Problems @ Javadoc Declaration Console
<terminated> OperationsPromotionExample [Java Appli
30
유니코드=66
출력문자=B
2
2.5
0.5
```


05

+ 연산에서 문자열 자동타입변환

05 + 연산에서 문자열 자동타입변환

■ + 연산

- 피연산자가 모두 숫자일 경우 덧셈 연산 수행
- 피연산자 중 하나가 문자열일 경우 나머지 피연산자도 문자열로 자동 변환되고 문자열 결합 연산 수행

```
int value = 3 + 7;    → int value = 10;  
String str = "3" + 7; → String str = "3" + "7"; → String str = "37";  
String str = 3 + "7"; → String str = "3" + "7"; → String str = "37";
```

■ + 연산은 앞에서부터 순차적으로 수행

- 먼저 수행된 연산이 덧셈연산인 경우 덧셈 결과를 가지고 그 다음 + 연산을 수행
- 먼저 수행된 연산이 결합 연산인 경우 이후 모든 연산이 결합 연산이 됨

```
int value = 1 + 2 + 3;    → int value = 3 + 3;    → int value = 6;  
String str = 1 + 2 + "3"; → String str = 3 + "3"; → String str = "33";  
String str = 1 + "2" + 3; → String str = "12" + 3; → String str = "123";  
String str = "1" + 2 + 3; → String str = "12" + 3; → String str = "123";
```

05 + 연산에서 문자열 자동타입변환

■ + 연산

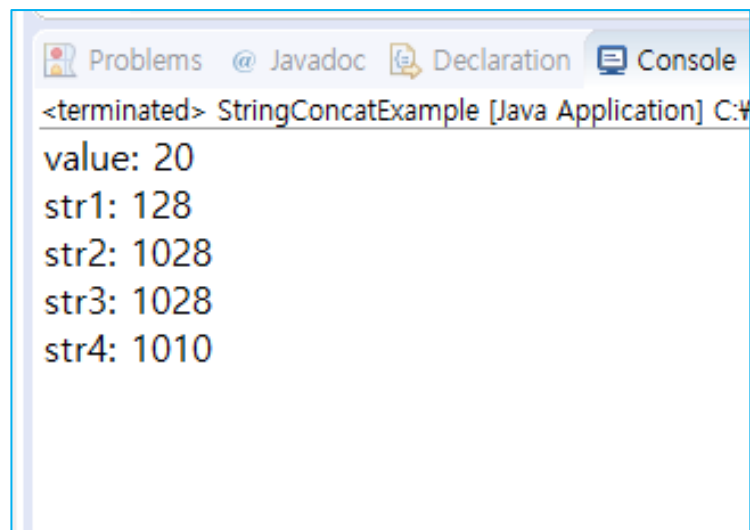
- 특정부분을 연산하고 싶을 경우 : 앞에서부터 순차적으로 + 연산을 수행하지 않고 우선 연산하고 싶은 부분을 괄호 ()로 묶어서 연산을 수행함.
- 괄호()는 최우선으로 연산을 수행함

```
String str = "1" +(2+3) ; ⇒ String str = "1" + 5 ; ⇒ String str = "15";
```

05 + 연산에서 문자열자동타입변환

예. 문자열 결합 연산

```
2
3 public class StringConcatExample {
4     public static void main(String[] args) {
5         //숫자 연산
6         int value = 10 + 2 + 8;
7         System.out.println("value: " + value);
8
9         //결합 연산
10        String str1 = 10 + 2 + "8";
11        System.out.println("str1: " + str1);
12
13        String str2 = 10 + "2" + 8;
14        System.out.println("str2: " + str2);
15
16        String str3 = "10" + 2 + 8;
17        System.out.println("str3: " + str3);
18
19        String str4 = "10" + (2 + 8);
20        System.out.println("str4: " + str4);
21    }
22 }
23
```



The screenshot shows an IDE's console window with the following output:

```
<terminated> StringConcatExample [Java Application] C:\...
value: 20
str1: 128
str2: 1028
str3: 1028
str4: 1010
```

06

문자열을 기본타입으로 강제변환

06 문자열을 기본 타입으로 강제 타입 변환

■ 문자열을 기본 타입으로 강제 변환하는 방법

- 문자열을 기본 타입으로 변환하는 경우가 매우 많음
- 예. "12" 와 "3.5"를 정수 및 실수 타입으로 변환해서 숫자 연산을 하는 경우
- **Integer.parseInt()** : 문자열을 정수 int 타입으로 변환
- **Double.parseDouble()** : 문자열을 실수 double 타입으로 변환

06 문자열을 기본 타입으로 강제 타입 변환

■ 문자열을 기본 타입으로 강제 변환하는 방법

변환 타입	사용 예
String → byte	<pre>String str = "10"; byte value = Byte.parseByte(str);</pre>
String → short	<pre>String str = "200"; short value = Short.parseShort(str);</pre>
String → int	<pre>String str = "300000"; int value = Integer.parseInt(str);</pre>
String → long	<pre>String str = "400000000000"; long value = Long.parseLong(str);</pre>
String → float	<pre>String str = "12.345"; float value = Float.parseFloat(str);</pre>
String → double	<pre>String str = "12.345"; double value = Double.parseDouble(str);</pre>
String → boolean	<pre>String str = "true"; boolean value = Boolean.parseBoolean(str);</pre>

06 문자열을 기본 타입으로 강제 타입 변환

- 문자열이 숫자가 아닌 알파벳이나 특수 문자, 한글 등의 요소를 포함하고 있을 경우 숫자타입으로 변환을 시도할 경우 숫자 형식 예외(Number Format Exception) 발생

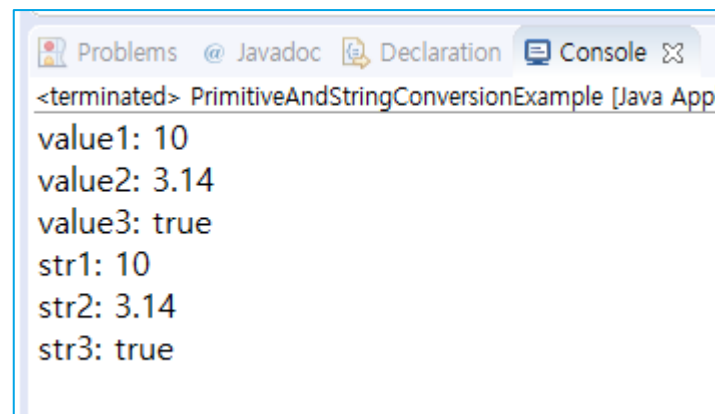
```
String str = "1a";  
int value = Integer.parseInt(str);    //NumberFormatException 발생
```

- 반대로 기본타입(byte, short, char, int, long, float, double, boolean)의 값을 문자열로 변경하는 경우에는 **String.valueOf()** 메소드 사용하여 기본 타입을 문자열로 변환
 - String str = String.valueOf(3)을 실행하면 문자열 "3"을 얻을 수 있음

06 문자열을 기본 타입으로 강제 타입 변환

예. 기본타입과 문자열 간의 변환

```
2
3 public class PrimitiveAndStringConversionExample {
4     public static void main(String[] args) {
5         int value1 = Integer.parseInt("10");
6         double value2 = Double.parseDouble("3.14");
7         boolean value3 = Boolean.parseBoolean("true");
8
9         System.out.println("value1: " + value1);
10        System.out.println("value2: " + value2);
11        System.out.println("value3: " + value3);
12
13        String str1 = String.valueOf(10);
14        String str2 = String.valueOf(3.14);
15        String str3 = String.valueOf(true);
16
17        System.out.println("str1: " + str1);
18        System.out.println("str2: " + str2);
19        System.out.println("str3: " + str3);
20    }
21 }
22
```



The screenshot shows an IDE console window with the following tabs: Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the program. The output is as follows:

```
<terminated> PrimitiveAndStringConversionExample [Java App
value1: 10
value2: 3.14
value3: true
str1: 10
str2: 3.14
str3: true
```

1. 다음 코드에서 컴파일 에러가 발생하는 위치와 이유를 설명해보세요.

```
01 short s1 = 1 ;  
02 short s2 = 2;  
03 int i1 = 3;  
04 int i2 = 4;  
05 short result = s1+ s2;  
06 int result = i1 +i2 ;
```

2. 다음 () 안에 들어갈 타입은 무엇이며, 출력되는 결과와 그 이유를 설명해보세요.

```
int x= 5 ;  
int y =2 ;  
( ) result = x/y ;  
System.out.println(result) ;
```

3. 다음 코드를 실행했을 때 출력 결과를 적어보세요.

```
String str1 = 2+3 + "" ;  
String str2 = 2 + "" +3 ;  
String str3 = "" + 2 + 3 ;  
System.out.println(str1) ;  
System.out.println(str2) ;  
System.out.println(str3) ;
```

질문은 이메일을 이용해주세요.
ds.june2@gmail.com

감사합니다