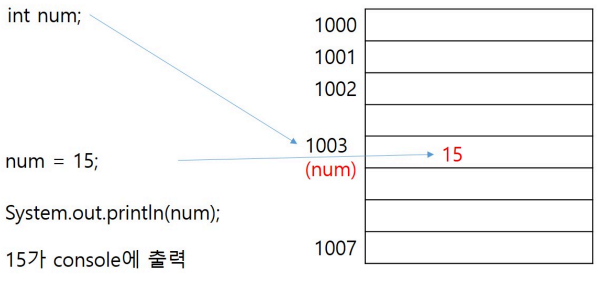
1. 변수

1. 변수란

1. 변수란 직역하면 변하는 값을 의미.
2. 프로그래밍 언어에서 변수는 값을 저장하는 메모리 공간.
3. 사용자가 로그인 시 입력하는 id나 pw를 데이터베이스에 저장되어있는 값과 비교해야 되는 데 비교하기 전에 사용자가 입력한 id와 pw를 기억하고 있을 공간이 필요하다. 프로그래밍 언어에서는 이러한 작업을 위해 변수로 메모리 공간을 할당받고 해당 메모리 공간에 값들을 저장하여 사용한다.



4. Java 코드 작성시 한 줄 한 줄을 명령이라고 생각하면 명령이 끝날 때 항상 세미콜론(;)을 붙여준다. 세미콜론은 명령의 종결을 의미한다. 세미콜론을 붙이지 않으면 에러로 인식하기 때문에 항상 습관적으로 세미콜론을 붙여준다.
5. 프로그래밍 언어에서는 항상 값을 저장하고 사용하는 로직이 포함되어 있기 때문에 변수의 선언과 사용은 필수적이다.

2. 변수의 선언

1. Java에서 변수의 선언은 **자료형 변수명;** 형태로 이뤄진다.
2. 변수를 선언 시 메모리 공간을 할당받고 변수명이 해당 메모리 공간을 가르키는 별칭이 되어서 변수명을 사용하면 메모리 공간에 저장되어 있는 값에 접근하여 사용할 수 있다.
3. 변수 선언 시 같은 자료형의 변수일 경우는 콤마(,)로 묶어서 여러개를 선언할 수 있다.

```
int num1, num2, num3;
```

4. 변수명 명명규칙
    - 변수명의 첫 번째 글자는 문자나 $나 _만 허용된다.

```
$price, price, _company(O)
1price, @price, %value(X) => 에러 발생
```

- 변수명은 대소문자를 구분한다.

```
//대소문자를 구분하기 때문에 서로 다른 변수로 인식
int intval;
int intVal;

//같은 이름으로 새로운 변수를 선언하면 에러로 인식한다.
int intval;
```

- 변수명은 영문자만 사용하는 것이 권장된다.
- 관례적으로 카멜케이스 표기법(낙타표기법)을 사용한다. 여러개의 단어가 합성된 변수명일 경우 합성되는 단어의 첫글자만 대문자로 표기하는 방식.

```
int intValue;
String bitcampJavaBasic;
```

- Java에서 변수명의 길이에는 제한이 없다.
- Java에서 사용하는 예약어들은 변수명으로 사용할 수 없다.

```
int, double, public, class, private, ....
```

- 개발자는 항상 팀 단위의 작업이 많기 때문에 항상 변수명은 누가 봐도 그 의미를 알 수 있게 직관적으로 만들어야한다.

```
//공부시간
int studyTime;
//기말고사 점수
int finalExamScore;
//배터리 남은 용량
int batteryRemainQntt;
```

3. 변수의 사용

1. 변수를 사용한다는 것은 변수가 가리키고 있는 메모리 공간에 값을 저장하거나 저장되어 있는 값을 사용한다는 의미이다.

2. 변수에 값을 저장하는 것을 할당이라고 하고 특히 변수에 최초로 값을 할당하는 것을 초기화라고 부른다.
3. 초기화되지 않은 변수를 사용하게 되면 에러가 발생한다.

```
//변수의 초기화
int num1, num2, num3;
num1 = 10;
num2 = 20;

//초기화되지 않은 변수 사용 시 에러 발생
System.out.println(num3);
```

4. 변수에 값을 저장할 때는 대입연산자(=)을 사용한다. 수학에서는 =이 같은 값이라는 의미인데 프로그래밍에서는 왼쪽 인자에 오른쪽 값을 대입(저장)한다라는 의미로 사용된다.
5. 변수를 선언하고 값을 저장해도 되지만 선언과 동시에 값을 저장할 수도 있다.

```
//변수의 선언과 저장을 동시에 진행
int num1 = 100;
```

- **예제: chap02_variables._01_variables.java**

4. 변수의 사용 범위

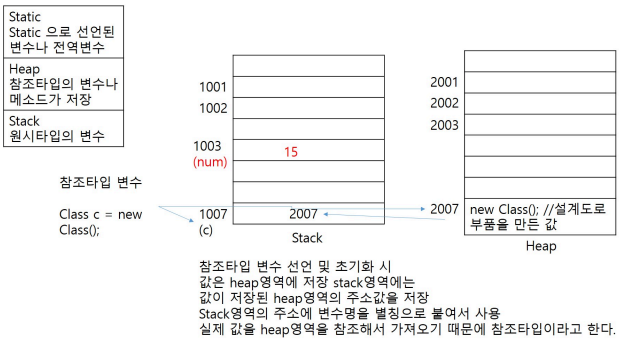
1. 변수는 중괄호({})블록 안에서 선언되고 사용된다. 중괄호 블록을 사용하는 곳은 클래스, 메소드, 제어문 등이 있다.
2. 중괄호 블록 안에서 선언된 변수는 해당 중괄호 안에서만 사용 가능하다.
3. 메소드 안에서 선언된 변수는 지역변수(로컬변수)라고 부르며 지역변수는 메소드가 호출될 때 메모리에 할당되었다가 메소드 호출이 끝나면 메모리 헤제된다.
4. 지역변수를 제외한 변수의 범위에는 인스턴스 변수, static 변수(클래스 변수)가 있다.

- **예제: chap02_variables._02_scopeOfVariables.java**

5. 변수의 자료형(타입)

1. Java에서의 자료형 두가지로 나뉜다. 기본 자료형(primitive type)과 참조 자료형(refferenced type)

2. 기본 자료형의 선언 형태는 모두 소문자로 되어있다. 소문자로 선언된 기본 자료형을 제외한 나머지 모든 타입은 참조형(객체, 배열, List, Map)이다.
3. 자료형 마다 저장되는 메모리 공간이 다르다. Java에서 메모리는 stack 영역, heap 영역, static 영역(메소드 영역)으로 구분된다.
4. stack 영역: 기본 자료형인 변수, 참조 타입의 값의 저장되어 있는 메모리 주소를 저장한다. GC의 영향을 받아서 사용하지 않는 변수는 바로 삭제된다.
5. heap 영역: 참조타입의 변수의 값, 클래스 정보등을 저장한다. GC의 영향을 받기 때문에 사용하지 않는 참조타입 변수(객체, 배열, List, ...)는 바로 메모리에서 삭제된다.
6. static 영역(메소드 영역): 메소드나 static 키워드로 생성된 변수를 저장한다. GC의 영향을 받지 않아서 프로그램 시작 시에 메모리에 값들이 저장되고 프로그램이 종료될 때까지 삭제되지 않는다.
7. 기본자료형 변수는 stack영역에 참조형 변수는 heap영역에 저장된다.
8. 참조형의 변수의 값은 heap에 저장되고 저장된 주소를 stack에 저장.



9. 참조형 변수는 stack영역에 변수명을 별칭으로 붙이고 실제 값은 stack 영역에 저장되어 있는 heap영역의 주소값을 참조해서 가져오기 때문에 참조형 변수라고 부른다.

6. 기본자료형(원시 타입, primitive type)

1. 기본 자료형은 총 8가지가 존재한다. byte, short, char, int, long, double, float, boolean
2. 정수형, 실수형, 논리형으로 구분한다.
3. 정수형 타입
    - 정수형 타입에는 5가지가 포함된다. byte, short, char, int, long
    - 위 5가지 타입을 크기대로 정렬하면 다음과 같다.
    - byte(1byte = 8bit = 00000000 ~ 11111111) < char(2byte) = short(2byte) < int(4byte) < long(8byte)
    - Java에서는 int형을 기본적인 정수타입으로 사용한다. 따라서 특별한 이유가 없을 때는 정수 값은 int형으로 선언하고 값을 저장하여 사용한다.

- byte, char, short과 같은 작은 단위의 자료형을 사용할 때는 항상 범위가 초과될 가능성을 염두해두고 사용한다. 값의 범위를 초과하면 에러가 발생하기 때문이다.
- byte 타입
  - byte 타입은 색상정보나 파일, 이미지 등 이전(바이너리) 데이터를 처리할 때 주로 사용한다.
  - -128 ~ 127의 값의 범위를 갖는다.
  - 컴퓨터는 2진수로 계산하기 때문에 최상위 비트(부호비트)고 나머지 7자리의 값이 실제 값이 된다. 1000 0000(-128) ~ 0111 1111(127)
  - 부호 비트는 0이면 양수, 1이면 음수를 나타낸다.
  - 양수일 때는 7자리의 값이 실제 값과 동일하지만 음수일 때는 7자리 값의 1의 보수를 취하고 1을 더해준 값이 실제 값이 된다.
  - 1의 보수는 각 비트(자리수)의 값을 반대로 바꿔준 값. 000 0000의 1의 보수는 111 1111
  - 음수일 경우 1의 보수의 1을 더한 값이 실제 값이 된다. 000 0000 -> 111 1111 + 1 -> 1000 0000(128)
  - byte보다 단위가 큰 char, short, int, long도 동일한 원리로 정수값을 표현한다.
  - **예제: chap02_variables._03_ByteType.java**
- char 타입(character, 문자를 담아주는 자료형)
  - Java에서는 모든 문자를 유니코드로 처리한다. 유니코드는 세계 각국의 문자를 코드 값으로 매핑해놓은 국제 표준 규격.
  - 유니코드는 0 ~ 65,535 범위의 2byte크기를 갖는 정수 값으로 이뤄져있다.
  - 0~127까지의 유니코드는 아스키코드(ASCII)가 할당되어 있고 한글 11,172자가 44,032~55,203에 매핑되어 있다.
  - 이외의 유니코드를 확인하려면 https://home.unicode.org
  - 유니코드에는 음수가 존재하지 않기 때문에 char 자료형에도 양수만 저장할 수 있다.
  - char 자료형 변수에 작은 따옴표로 묶은 문자 하나를 저장하게 되면 해당 유니코드 값이 변수에 저장된다.

```
char ch1 = 'A'; => 유니코드 값인 65가 저장된다.
//출력할 때는 유니코드 값에 해당되는 문자가 출력된
System.out.println(ch1);

//유니코드 값을 알고 있을 때는 유니코드 값을 바로
char ch2 = 65;
```

  - **예제: chap02_variables._04_CharType.java**

- short 타입
  - short 타입은 char 타입과 마찬가지로 2byte 크기를 표현할 수 있고 표현할 수 있는 개수도 동일하다.(65,536개, 2^16 개)
  - char 타입은 양수만 표현할 수 있는 반면 short 음수까지 표현이 가능하다.
  - char의 범위: 0 ~ 65,535
  - short의 범위: -32,768 ~ 32,767
- int 타입
  - int 타입은 4byte로 정수를 표현하는 타입.(2^32개)
  - 정수형 타입 중 가장 많이 사용되는 타입.
  - Java에서 정수 연산을 하기 위한 기본 타입이 int 타입이다.
  - short, byte 타입의 값들을 연산할 때 자동으로 int 형으로 변환 후 연산된다.
  - byte, short, int 중 어떤 타입으로 선언되어도 연산시에는 int 타입으로 변환되기 때문에 연산속도는 크게 차이 나지 않는다.
- long 타입
  - long 타입은 8byte 크기의 범위를 갖는 정수형 타입(2^64 개)
  - long 타입은 long 타입을 명시하기 위해서 값 뒤에 L, l을 붙인다.

```
long longVal1 = 100L;
long longVal2 = 100l;
long longVal3 = 100;
//int형의 범위를 초과하는 값을 할당할 때는 L, l을
long longVal4 = 3000000000L;
```

  - long 타입은 값이 매우 큰 수를 저장하여 사용할 때 선언한다.

- **예제: chap02_variables._05_ShortIntLong.java**

4. 실수형 타입(float, double)
   - float 타입
     - float는 4byte 크기의 값을 갖는 실수형 타입
     - Java에서 실수형 연산은 8byte로 진행되기 때문에 float 타입보다는 double 타입이 더 많이 사용된다.
     - float 타입의 값에는 F나 f를 붙여준다. float 값이라는 것을 명시
   - double 타입
     - double은 8byte 크기의 값을 갖는 실수형 타입

        - float 타입을 사용하면 범위를 초과하는 에러가 많이 발생하기 때문에 실제 개발 시에는 double 타입을 주로 사용한다.
    - **예제: chap02_variables._06_FloatDouble.java**
5. 논리형 타입(boolean)
    - boolean 타입
        - boolean은 1byte의 크기를 갖는 타입이고 값은 true, false만 저장할 수 있다.
        - boolean은 모든 값(true, false)을 1bit로 표현할 수 있는데 프로그래밍 언어의 메모리 기본 단위가 1byte여서 어쩔 수 없이 1byte를 모두 사용한다.
        - boolean은 플래그 용도로 많이 사용된다. 인스타그램에서 로그인 한 유저가 좋아요를 눌렀는지 boolena isLikeClick;
        - boolean은 조건문과 주로 많이 사용된다. if문이나 삼항연산자와 많이 사용된다.
    - **예제: chap02_variables._07_BooleanType.java**

7. var 자료형(타입 지정 없이 변수 선언, Java 10버전 이상)

1. var 자료형은 타입 지정 없이 변수를 선언하고 사용할 수 있는 자료형이다.
2. Java의 변수 자료형 추론이라는 기능을 통해서 var 변수에 저장된 값이 어떤 타입인지 자동을 추론해서 사용한다.

```
var str = "hello"; => 자료형 추론을 통해서 str의 값이 Strin
```

3. var 자료형은 항상 선언과 동시에 초기화(값의 할당)이 이뤄져야 한다.
4. 이미 자료형 추론인 끝난 변수에 다른 타입의 값을 저장하면 에러가 발생한다.

```
var str = "hello";
str = "java";
str = 10; => String으로 자료형 추론이 끝난 상태에서 다른 타
```

- **예제: chap02_variables._08_VarVariables.java**

8. 상수

1. 상수라는 것은 한 번 정해지면 변하지 않는 값을 의미한다. 프로그래밍 언어에서는 한 번 저장된 값을 변경할 수 없는 변수를 상수라고 한다.

2. Java에서는 상수를 선언할 때 final 키워드를 사용한다. 변수 선언 시 자료형 앞에 final 키워드를 붙여서 선언한다.

```
final int MONTH_COUNT = 12;
final double PI = 3.14;
```

3. 상수는 정해져있고 변하지 않는 값을 지정할 때 주로 사용.

```
final int WEEKDAYS = 7;
final int YEAR_DAYS = 365;
```

4. 상수는 관례적으로 모든 변수명을 대문자로 지정한다.
5. 두 단어 이상이 조합될 때는 스네이크표기법(단어와 단어 사이에 언더바'_' 넣어주는 표기법)을 사용한다.
6. 코드 내에서 반복적으로 사용되는 특정 값을 상수로 지정하고 사용하면 코드의 재사용성을 높이고 코드의 길이를 줄일 수 있다.

- **예제: chap02_variables._09_Constant.java**

9. 형변환(타입변환, Type Casting)

1. 형변환이라는 것은 자료형이 지정되고 값이 저장된 변수의 자료형을 변경하는 작업.

```
int num = 65;
System.out.println(num);
//A로 출력하고 싶을 때는 char으로 출력해야 한다.
System.out.println((char)num);
```

2. 형변환에는 묵시적(자동) 형변환과 명시적(강제) 형변환이 존재한다.
3. 자동 형변환
    - 자동 형변환 1: 자료형의 byte 크기가 작은 타입에서 큰 타입으로 자동으로 형변환된다.

```
int iNum = 10;
//int 형은 4byte이기 때문에 자동으로 8byte인 long 형으
long lNum = iNum;

//에러 발생
short sNum = iNum;
```

- 자동 형변환 2: 좀 더 세밀한 표현범위를 갖는 타입으로는 자동 형변환된다.(정수형들은 실수형으로 자동 형변환된다.)

```
int iNum = 20;
//int와 float의 byte 크기가 같지만 float가 좀 더 세밀한
float fNum1 = iNum;

long lNum = 30L;
//long 형이 float 형보다 byte 크기가 크지만 float형이
float fNum2 = lNum;
```

- 자동 형변환 순서 byte(1byte) -> short(2byte) -> int(4byte) -> long(8byte) -> float(4byte) -> double(8byte)

4. 강제 형변환
   - 강제 형변환은 byte 크기가 작은 자료형으로 변환하거나 덜 세밀한 표현 범위를 갖는 자료형으로 변환할 때 사용
   - 강제 형변환을 하는 방식은 형변환을 하고 싶은 값이나 변수 앞에 소괄호()와 변환할 타입을 지정해준다.

```
int iNum = 30;
short sNum = (short)iNum;

float fNum = 3.14f;
long lNum = (long)fNum;
```

   - 범위를 넘어가는 값을 형변환 했을 경우에는 에러가 발생한다.

```
int iNum = 128;
byte bNum = (byte)iNum; => 범위 초과 에러 발생
```

5. 연산에서의 형변환
   - 서로 다른 자료형을 연산할 때는 자동 형변환 순서에 따라 자동 형변환이 이뤄진 후 연산이 진행된다.
   - 결과 값도 형변환이 된 자료형으로 나타난다.

```
int iNum = 10;
double dNum = 32.123;

//int 형인 iNum이 double 형으로 형변환이 이뤄진 후 연산
double result = iNum + dNum;
```

6. 문자열 결합 연산자(+): 서로 다른 문자열을 이어 줄 때 사용하는 연산자.

- 문자열(String) 타입과 다른 자료형을 + 연산하게 되면 문자열 결합으로 인식하여 다른 자료형들이 String 타입으로 변환된다.

```
int iNum = 17;
String str = "java";

str + iNum => "java" + "17" => "java17"
```

- **예제: chap02_variables._10_TypeCasting.java**