2. 변수

#1.인강/0.자바/1.자바-입문

변수 시작

변수에 대해서 본격적으로 알아보기 전에 다음 코드를 작성하고 실행해보자.

Var1

```
package variable;

public class Var1 {

    public static void main(String[] args) {
        System.out.println(10);
        System.out.println(10);
        System.out.println(10);
    }
}
```

패키지(package)

- 이번에는 처음으로 패키지를 만든다.
- 패키지는 지금 단계에서는 자바 파일을 구분하기 위한 폴더로 이해하면 된다.
- `variable` 라는 패키지를 만들었다면, 해당 패키지에 들어가는 자바 파일 첫줄에 `package variable;` 와 같이 소속된 패키지를 선언해주어야 한다.
- 자바 파일이 위치하는 패키지와 `package variable` 선언 위치가 같아야 한다.

실행 결과

```
10
10
10
```

단순히 숫자 10을 3번 출력하는 코드이다. 그런데 여기서 숫자 10을 3번 출력하는 대신에 숫자 20을 3번 출력하도록 코드를 변경해보자. 어떻게 해야할까?

Var1

```
package variable;

public class Var1 {

    public static void main(String[] args) {
        System.out.println(20); //변경 10 -> 20
        System.out.println(20); //변경 10 -> 20
        System.out.println(20); //변경 10 -> 20
    }
}
```

숫자 10이라고 적혀 있는 곳을 모두 찾아서 숫자 20으로 변경해야 한다. 여기서는 총 3번의 코드 변경이 발생했다.
단순한 예제여서 코드를 3번만 변경했지만, 만약 숫자 10을 출력하는 부분이 100개라면 100개의 코드를 모두 변경해야 한다.

더 나아가서 사용자가 숫자를 입력하고, 사용자가 입력한 숫자를 출력하고 싶다면 어떻게 해야할까? 사용자가 입력한 값은 항상 변한다. 누군가는 100을 입력하고 누군가는 200을 입력할 수도 있다. (사용자 입력은 뒤에서 다룬다)
결국 어딘가에 값을 보관해두고 필요할 때 값을 꺼내서 읽을 수 있는 저장소가 필요하다. 쉽게 비유하자면 데이터를 담을 수 있는 그릇이 필요하다.

모든 프로그래밍 언어는 이런 문제를 해결하기 위해 **변수(variable)**라는 기능을 제공한다. 변수는 이름 그대로 변할 수 있다는 뜻이다.

다음 코드를 작성해보자.

Var2

```
package variable;

public class Var2 {

    public static void main(String[] args) {
        int a; //변수 선언
```

```
        a = 10; //변수 초기화
        System.out.println(a);
        System.out.println(a);
        System.out.println(a);
    }
}
```

a = 10 실행 결과

```
10
10
10
```

이번에는 `a = 20`으로 변경해서 실행해보자

Var2

```
package variable;

public class Var2 {

    public static void main(String[] args) {
        int a; //변수 선언
        a = 20; //10 -> 20으로 변경
        System.out.println(a);
        System.out.println(a);
        System.out.println(a);
    }
}
```

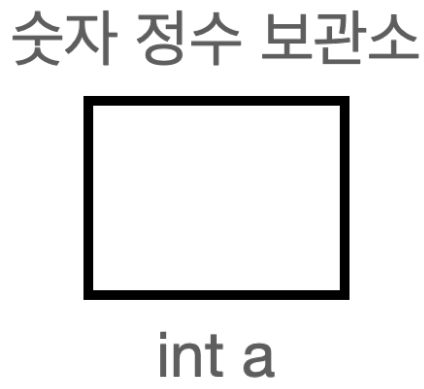
a = 20 실행 결과

```
20
20
20
```

`a`의 값을 변경하면 출력결과가 모두 함께 변경되는 것을 확인할 수 있다.

코드를 간단히 분석해보자.

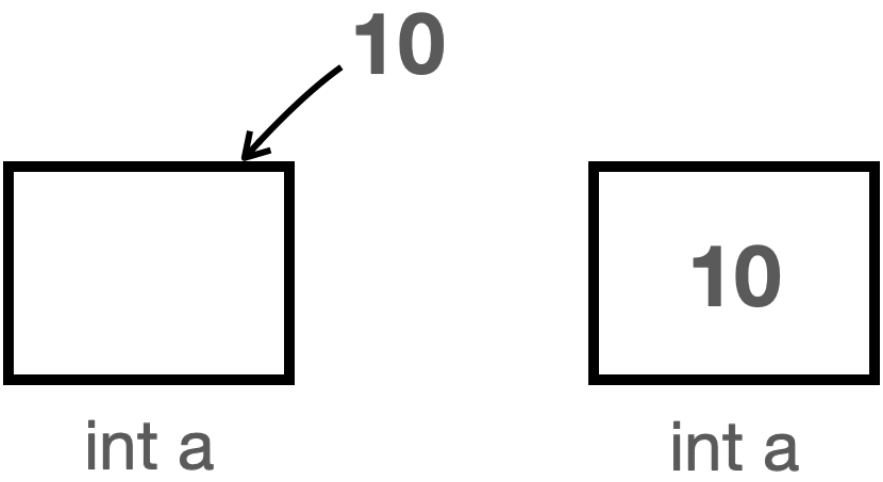
변수 선언



`int a`

- 숫자 정수(`integer`)를 보관할 수 있는 이름이 `a`라는 데이터 저장소를 만든다. 이것을 변수라 한다.
- 이렇게 변수를 만드는 것을 변수 선언이라 한다.
- 이제 변수 `a`에는 숫자 정수를 보관할 수 있다.
- 숫자 정수 뿐만 아니라 문자, 소수와 같이 다양한 종류 값을 저장할 수 있는 변수들이 있다. 우선은 숫자 정수를 저장하는 `int`를 알아두자

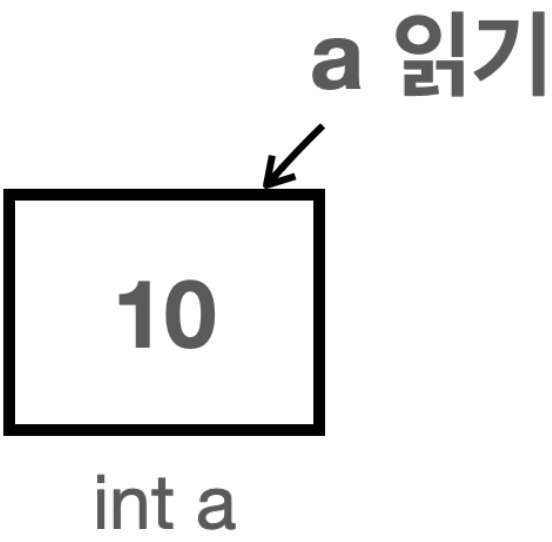
변수에 값 대입



`a = 10`

- 자바에서 `=`은 오른쪽에 있는 값을 왼쪽에 저장한다는 뜻이다. 수학에서 이야기하는 두 값이 같다(`equals`)와는 다른 뜻이다!
- 숫자를 보관할 수 있는 데이터 저장소인 변수 `a`에 값 `10`을 저장한다.
- 이처럼 선언한 변수에 처음으로 값을 대입해서 저장하는 것을 변수 초기화라 한다.

변수 값 읽기



`System.out.println(a)`

- 변수에 저장되어 있는 값을 읽어서 사용하는 방법은 간단하다. 변수 이름을 적어주기만 하면 된다.
- 변수 `a`에 `10`이 들어가 있다면 자바는 실행 시점에 변수의 값을 읽어서 사용한다. 따라서 다음과 같이 해석된다.
  - `System.out.println(a)  //변수 a의 값을 읽음`
  - `System.out.println(10) //a의 값인 10으로 변경, 숫자 10 출력`
- 참고로 변수의 값은 반복해서 읽을 수 있다. 변수의 값을 읽는다고 값이 없어지는 것이 아니다.

변수 값 변경

변수는 이름 그대로 변할 수 있는 수이다. 쉽게 이야기해서 변수 `a`에 저장된 값을 언제든지 바꿀 수 있다는 뜻이다.

이번에는 중간에 변수의 값을 변경해보자.

Var3

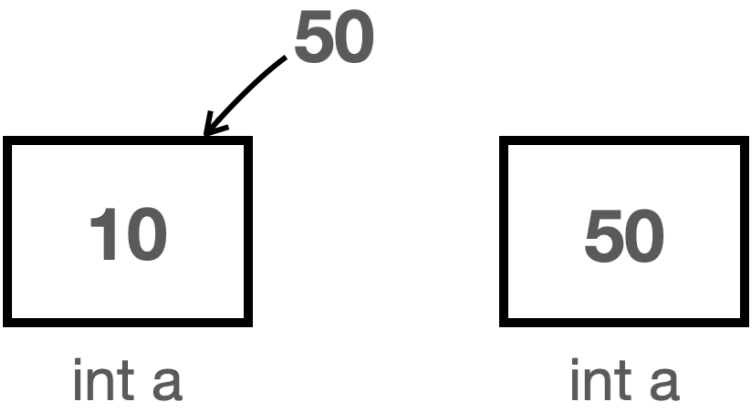
```
package variable;

public class Var3 {

    public static void main(String[] args) {
        int a; //변수 선언
        a = 10; //변수 초기화: a(10)
        System.out.println(a); //10 출력
        a = 50; //변수 값 변경: a(10 -> 50)
        System.out.println(a); //50 출력
    }
}
```

실행 결과

```
10
50
```

변수의 값이 변경된 이후에는 10 대신에 50이 출력된 것을 확인할 수 있다.

변수 값 변경



프로그램은 한 줄씩 순서대로 실행된다. 어떻게 실행된 것인지 자세히 확인해보자.

```
a = 10; //변수 초기화: a(10) //1. 변수 a에 10을 저장한다.
System.out.println(a); //2. 변수 a의 값을 읽는다. a에는 10이 들어있다. 10을 출력한다.
a = 50; //변수 값 변경:   //3. 변수 a의 값을 50으로 변경한다. a(10 -> 50)
System.out.println(a); //4. 변수 a의 값을 읽는다. a에는 50이 들어있다. 50을 출력한다.
```

참고로 변수의 값을 변경하면 변수에 들어있던 기존 값은 값은 삭제된다.

변수 선언과 초기화

변수 선언

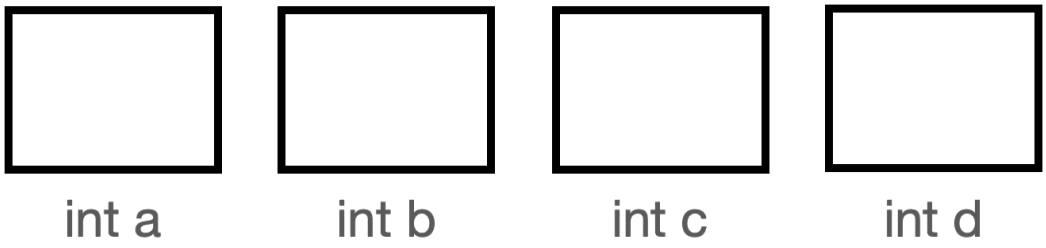
변수를 선언하면 컴퓨터의 메모리 공간을 확보해서 그곳에 데이터를 저장할 수 있다. 그리고 변수의 이름을 통해서 해당 메모리 공간에 접근할 수 있다. 쉽게 이야기해서 데이터를 보관할 수 있는 공간을 만들고, 그곳에 이름을 부여한다.

Var4

```
package variable;

public class Var4 {
```

```
    public static void main(String[] args) {
        int a;
        int b;

        int c,d;
    }
}
```

변수 선언



변수는 다음과 같이 하나씩 선언할 수도 있고

```
int a;
int b;
```

다음과 같이 한번에 여러 변수를 선언할 수도 있다.

```
int c,d;
```

변수 초기화

변수를 선언하고, 선언한 변수에 처음으로 값을 저장하는 것을 변수 초기화라 한다.

Var5

```
package variable;

public class Var5 {

    public static void main(String[] args) {
        //1. 변수 선언, 초기화 각각 따로
        int a;
        a = 1;
        System.out.println(a);

        int b = 2; //2. 변수 선언과 초기화를 한번에
        System.out.println(b);
```

```
        int c = 3, d = 4; //3. 여러 변수 선언과 초기화를 한번에
        System.out.println(c);
        System.out.println(d);
    }
}
```

1. 변수의 선언과 초기화를 각각 따로 할 수 있다.
2. 변수를 선언하면서 동시에 초기화 할 수 있다.
   `int b`를 사용해서 변수 `b`를 만들고 그 다음에 바로 `b = 2`를 사용해서 변수 `b`에 값 2를 저장한다.
3. 여러 변수를 선언하면서 초기화도 동시에 진행할 수 있다.

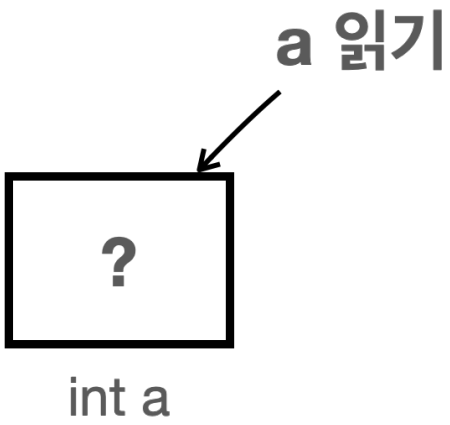
변수는 초기화 해야한다

만약 변수를 초기화 하지 않고 사용하면 어떻게 될까?

Var6

```
package variable;

public class Var6 {

    public static void main(String[] args) {
        int a;
        System.out.println(a); //주석을 풀면 컴파일 에러 발생
    }
}
```

초기화 하지 않은 변수 읽기



다음과 같은 컴파일 에러가 발생한다.

```
java: variable a might not have been initialized
```

해석해보면 변수가 초기화되지 않았다는 오류이다.

왜 이런 오류가 발생할까? 컴퓨터에서 메모리는 여러 시스템이 함께 사용하는 공간이다. 그래서 어떠한 값들이 계속 저

장된다.
변수를 선언하면 메모리상의 어떤 공간을 차지하고 사용한다. 그런데 그 공간에 기존에 어떤 값이 있었는지는 아무도 모른다. 따라서 초기화를 하지 않으면 이상한 값이 출력될 수 있다. 이런 문제를 예방하기 위해 자바는 변수를 초기화 하도록 강제한다.

> **참고**: 지금 학습하는 변수는 지역 변수(Local Variable)라고 하는데, 지역 변수는 개발자가 직접 초기화를 해주어야 한다. 나중에 배울 클래스 변수와 인스턴스 변수는 자바가 자동으로 초기화를 진행해준다.

> **참고**: 컴파일 에러는 자바 문법에 맞지 않았을 때 발생하는 에러이다. 컴파일 에러는 오류를 빨리, 그리고 명확하게 찾을 수 있기 때문에 사실은 좋은 에러이다. 덕분에 빠르게 버그를 찾아서 고칠 수 있다.

에러를 확인하고 나면 꼭 다음과 같이 해당 라인 전체에 주석을 적용하자. 그렇지 않으면 다른 예제를 실행할 때도 이 부분에 컴파일 에러가 발생할 수 있다.

```
//System.out.println(a); //주석을 풀면 컴파일 오류 발생
```

변수 타입1

변수는 데이터를 다루는 종류에 따라 다양한 형식이 존재한다.
다음 코드를 실행해보자.

Var7

```
package variable;

public class Var7 {

    public static void main(String[] args) {
        int a = 100; //정수
        double b = 10.5; //실수
        boolean c = true; //불리언(boolean) true, false 입력 가능
        char d = 'A'; //문자 하나
        String e = "Hello Java"; //문자열, 문자열을 다루기 위한 특별한 타입

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        System.out.println(d);
```

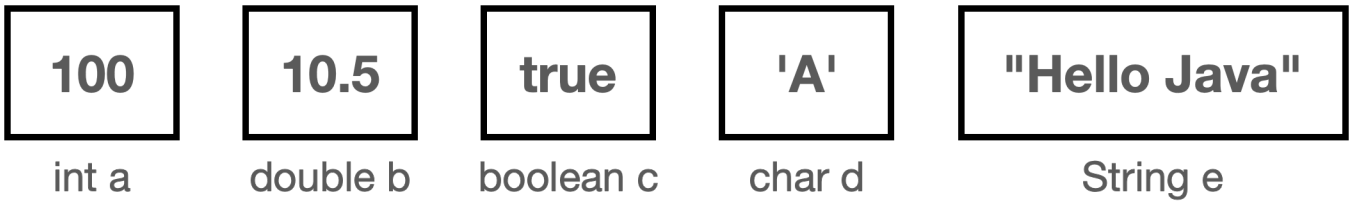
```
            System.out.println(e);
        }
    }
```

실행 결과

```
100
10.5
true
A
Hello Java
```

변수는 데이터를 다루는 종류에 따라 다양한 형식이 존재한다. 이러한 형식을 영어로는 `타입`(type)이라 하고, 우리말로는 `형식` 또는 `형`이라 한다. 예를 들어서 `int` 타입, `int` 형식, `int`형 등으로 부른다. 특별히 구분하지 않고 섞어서 부르기 때문에 모두 같은 말로 이해하면 된다.

변수 타입의 예

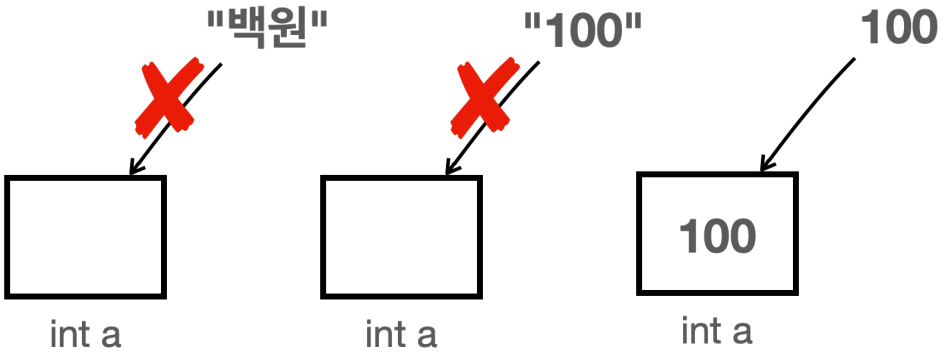


- `int`: 정수를 다룬다. 예) `1`, `100`, `1000`
- `double`: 실수를 다룬다. 예) `0.2`, `1.5`, `100.121`
- `boolean`: 불리언 타입이라 한다. `true`, `false` 값만 사용할 수 있다. 주로 참과 거짓을 판단하는 곳에서 사용한다.
- `char`: 문자 하나를 다룰 때 사용한다. 작은따옴표(`'`)를 사용해서 감싸야 한다. 예) `'A'`, `'가'`
- `String`: 문자열을 다룬다. 큰따옴표를 사용해야 한다. 예) `"hello java"`

> 참고: `String`은 첫 글자가 대문자로 시작하는 특별한 타입이다. 이 부분은 뒤에 클래스를 배워야 자세히 이해할 수 있다. 지금은 문자열을 다루는 특별한 타입이라고 이해하면 된다. `String`에 대한 자세한 내용은 별도로 다룬다.

자신의 타입에 맞는 데이터 사용

각 변수는 지정한 타입에 맞는 값을 사용해야 한다. 예를 들어서 다음의 앞의 두 코드는 컴파일 오류가 발생한다



- `int a = "백원"` : 정수 타입에 문자열(X)
- `int a = "100"` : 정수 타입에 문자열(X), 이것은 숫자 100이 아니라 문자열 `"100"` 이다. 문자를 나타내는 쌍따옴표(`"`)로 감싸져 있다.
- `int a = 100` : 정수 타입에 정수 100(O)

리터럴

코드에서 개발자가 직접 적은 `100`, `10.5`, `true`, `'A'`, `"Hello Java"` 와 같은 고정된 값을 프로그래밍 용어로 리터럴(literal)이라 한다.

```
int a = 100; //정수 리터럴
double b = 10.5; //실수 리터럴
boolean c = true; //불리언 리터럴
char d = 'A'; //문자 하나 리터럴
String e = "Hello Java"; //문자열 리터럴
```

변수의 값은 변할 수 있지만 리터럴은 개발자가 직접 입력한 고정된 값이다. 따라서 리터럴 자체는 변하지 않는다.

> 참고: 리터럴(literal)이라는 단어의 어원이 문자 또는 글자를 의미한다.

변수 타입2

숫자 타입

이번에는 다양한 숫자 타입을 자세히 알아보자.

바로 다음 코드를 작성해보자.

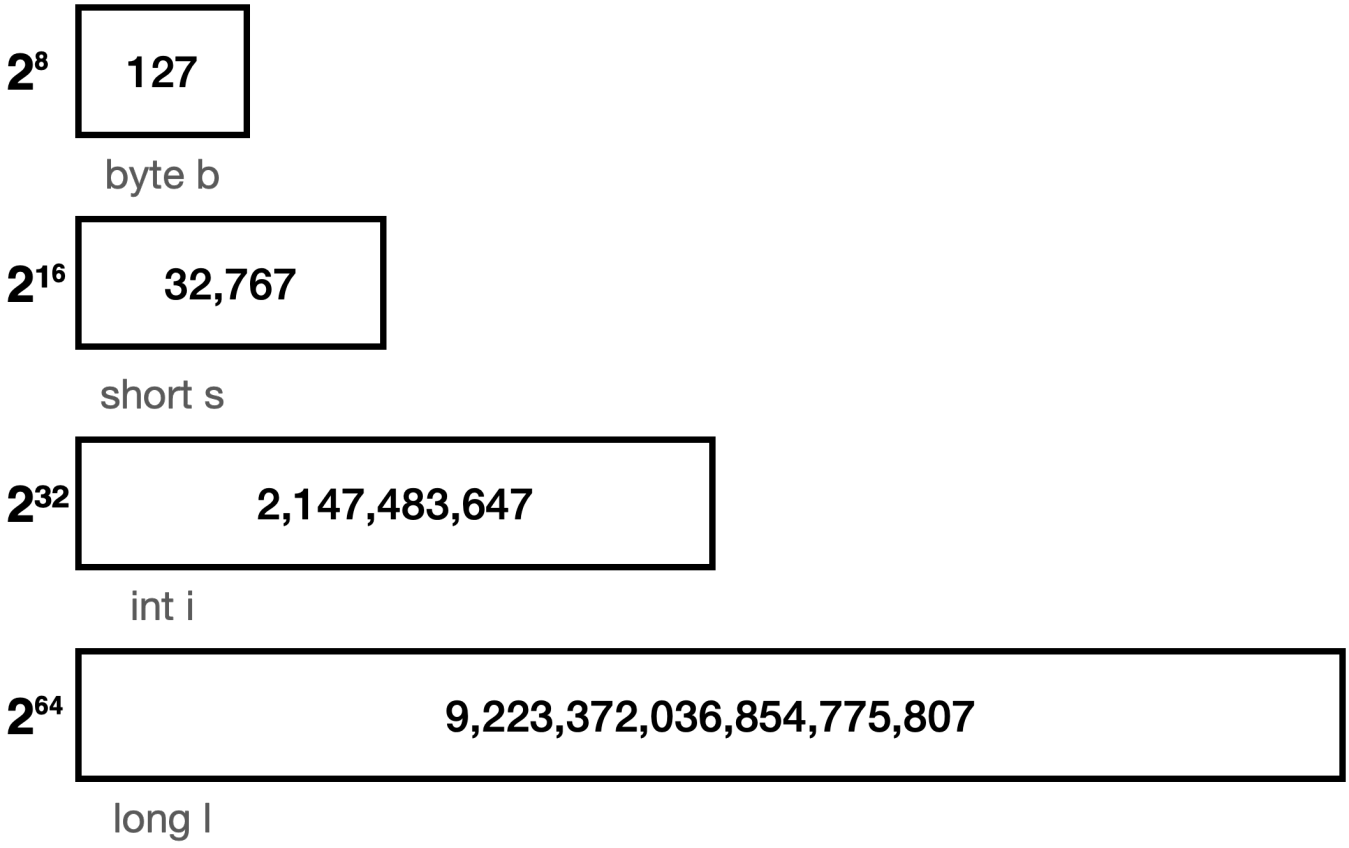
Var8

```
package variable;

public class Var8 {
```

```
    public static void main(String[] args) {
        //정수
        byte b = 127; //-128 ~ 127
        short s = 32767; // -32,768 ~ 32,767
        int i = 2147483647; //-2,147,483,648 ~ 2,147,483,647 (약 20억)

        //-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
        long l = 9223372036854775807L;

        //실수
        float f = 10.0f;
        double d = 10.0;
    }
}
```

메모리를 적게 사용하면 작은 숫자를 표현할 수 있고, 메모리를 많이 사용하면 큰 숫자를 표현할 수 있다.
변수를 선언하면 표현 범위에 따라 메모리 공간을 차지한다. 그래서 필요에 맞도록 다양한 타입을 제공한다.

변수와 메모리 공간 크기



표현할 수 있는 숫자의 범위와 차지하는 메모리 공간은 다음과 같다. 기타 타입도 함께 정리해두었다.

- 정수형

    - `byte` : -128 ~ 127 (1byte, 2^8)
    - `short` : -32,768 ~ 32,767 (2byte, 2^{16})
    - `int` : -2,147,483,648 ~ 2,147,483,647 (약 20억) (4byte, 2^{32})
    - `long` : -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 (8byte, 2^{64})
- 실수형
    - `float` : 대략 -3.4E38 ~ 3.4E38, 7자리 정밀도 (4byte, 2^{32})
    - `double` : 대략 -1.7E308 ~ 1.7E308, 15자리 정밀도 (8byte, 2^{64})
- 기타
    - `boolean` : `true` , `false` (1byte)
    - `char` : 문자 하나(1byte)
    - `String` : 문자열을 표현한다. 메모리 사용량은 문자 길이에 따라 동적으로 달라진다. (특별한 타입이다. 자세한 내용은 뒤에서 학습한다)

리터럴 타입 지정

- 정수 리터럴은 `int` 를 기본으로 사용한다. 따라서 `int` 범위까지 표현할 수 있다. 숫자가 `int` 범위인 약 20억을 넘어가면 `L` 을 붙여서 정수 리터럴을 `long` 으로 변경해야 한다. (대문자 `L` , 소문자 `l` 모두 가능하다 그런데 소문자 `l` 은 숫자 1과 착각할 수 있어서 권장하지 않는다.)
- 실수 리터럴은 기본이 `double` 형을 사용한다. `float` 형을 사용하려면 `f` 를 붙여서 `float` 형으로 지정해야 한다.

변수 타입 정리

이렇게 많은 변수들을 실제로 다 외우고 사용해야 할까?

다음 타입은 실무에서 거의 사용하지 않는다.

- `byte` : 표현 길이가 너무 작다. 또 자바는 기본으로 4byte(`int`)를 효율적으로 계산하도록 설계되어 있다. `int` 를 사용하자.
    - `byte` 타입을 직접 선언하고 여기에 숫자 값을 대입해서 계산하는 일은 거의 없다.
    - 대신에 파일을 바이트 단위로 다루기 때문에 파일 전송, 파일 복사 등에 주로 사용된다.
- `short` : 표현 길이가 너무 작다. 또 자바는 기본으로 4byte(`int`)를 효율적으로 계산하도록 설계되어 있다. `int` 를 사용하자
- `float` : 표현 길이와 정밀도가 낮다. 실수형은 `double` 을 사용하자.
- `char` : 문자 하나를 표현하는 일은 거의 없다. 문자 하나를 표현할 때도 문자열을 사용할 수 있다.
    - 예를 들어 `String a = "b"` 와 같이 사용하면 된다.

> 참고: 메모리 용량은 매우 저렴하다. 따라서 메모리 용량을 약간 절약하기 보다는 개발 속도나 효율에 초점을 맞추는 것이 더 효과적이다.

자주 사용하는 타입

실무에서 자주 사용하는 타입은 다음과 같다.

- 정수 - `int`, `long`: 자바는 정수에 기본으로 `int`를 사용한다. 만약 20억이 넘을 것 같으면 `long`을 쓰면 된다.
  - 파일을 다룰 때는 `byte`를 사용한다.
- 실수 - `double`: 실수는 고민하지 말고 `double`을 쓰면 된다.
- 불린형 - `boolean`: `true`, `false` 참 거짓을 표현한다. 이후 조건문에서 자주 사용된다.
- 문자열 - `String`: 문자를 다룰 때는 문자 하나든 문자열이든 모두 `String`을 사용하는 것이 편리하다.

자주 사용하는 타입을 제외하고 실무에서 나머지를 사용하는 경우는 거의 없다. 그나마 파일 전송시에 `byte`를 사용하는 것 정도이다.
따라서 자주 사용하는 타입만 이해하고 나머지는 이런게 있구나 하고 넘어가도 충분하다.

변수 명명 규칙

자바에서 변수의 이름을 짓는데는 규칙과 관례가 있다.
규칙은 필수이다. 규칙을 지키지 않으면 컴파일 오류가 발생한다.
관례는 필수는 아니지만 전세계 개발자가 해당 관례를 따르기 때문에 사실상 규칙이라고 생각해도 된다.

규칙

- 변수 이름은 숫자로 시작할 수 없다. (예: `1num`, `1st`)
  - 그러나 숫자를 이름에 포함하는 것은 가능하다 (예: `myVar1`, `num1`).
- 이름에는 공백이 들어갈 수 없다.
- 자바의 예약어를 변수 이름으로 사용할 수 없다. (예: `int`, `class`, `public`)
- 변수 이름에는 영문자(`a-z`, `A-Z`), 숫자(`0-9`), 달러 기호(`$`) 또는 밑줄(`_`)만 사용할 수 있다.

관례

- 소문자로 시작하는 낙타 표기법
  - 변수 이름은 소문자로 시작하는 것이 일반적이다. 여러 단어로 이루어진 변수 이름의 경우, 첫 번째 단어는 소문자로 시작하고 그 이후의 각 단어는 대문자로 시작하는 낙타 표기법(camel case)를 사용한다. (예: `orderDetail`, `myAccount`)

> **낙타표기법**
> 낙타표기법(Camel Case)은 프로그래밍에서 변수, 함수, 클래스 등의 이름을 지을 때 많이 사용하는 표기법 중 하나이다. 이 표기법의 이름은 작성한 이름이 여러 단어로 구성되어 있을 때, 각 단어의 첫 글자가 대문자로 시작하고, 이 대문자들이 낙타의 등봉처럼 보이는 것에서 유래했다. 낙타표기법을 사용하면 이름에 공백을 넣지 않고도 여러 단어를 쉽게 구분할 수 있으므로, 변수의 이름을 이해하기 쉽게 만들어준다. 또한, 대부분의 프로그래밍 언어에서는 이름에 공백을 포함할 수 없기 때문에, 낙타표기법은 이런 제한을 우회하는 좋은 방법이다.

자바 언어의 관례 한번에 정리

클래스는 대문자로 시작, 나머지는 소문자로 시작

- 자바에서 클래스 이름의 첫 글자는 대문자로 시작한다. 그리고 나머지는 모두 첫 글자를 소문자로 시작한다. 여기에 낙타 표기법을 적용하면 된다. 이렇게 하면 모든 자바 관례를 다 외울 수 있다!
- 예시: 클래스는 첫 글자 대문자, 나머지는 모두 첫 글자 소문자로 시작 + 낙타 표기법
  - 클래스: `Person`, `OrderDetail`
  - 변수를 포함한 나머지: `firstName`, `userAccount`
- 여기에 예외가 딱 2개 있다.
  - 상수는 모두 대문자를 사용하고 언더바로 구분한다. (상수는 뒤에서 학습)
    - `USER_LIMIT`
  - 패키지는 모두 소문자를 사용한다. (패키지는 뒤에서 학습)
    - `org.spring.boot`

참고: 변수 이름은 의미있고, 그 용도를 명확하게 설명해야 한다.

- `a`, `b`: 이런 변수는 용도를 설명하지 않는다. 단순한 예제에서만 사용하는 것이 좋다.
- `studentCount`, `maxScore`, `userAccount`, `orderCount`: 용도를 명확하게 설명한다.

문제와 풀이

- 문제를 스스로 풀어보세요.

> **백문이 불여일타!**
> 프로그래밍은 수영이나 자전거 타기와 비슷하다. 아무리 책을 보고 강의 영상을 봐도 직접 물속에 들어가거나 자

전거를 타봐야 이해할 수 있다. 프로그래밍도 마찬가지이다. 이해하고 외우는 것도 중요하지만 무엇보다 직접 코딩을 해보는 것이 더욱 중요하다. 읽어서 이해가 잘 안되더라도 직접 코딩을 해보면 자연스럽게 이해가 되는 경우도 많다. 백번 읽는 것 보다 한번 직접 코딩해서 결과를 보는 것이 좋은 개발자로 빠르게 성장할 수 있는 지름길이다.

코딩이 처음이라면 필독!
프로그래밍이 처음이라면 아직 코딩 자체가 익숙하지 않기 때문에 문제와 풀이에 상당히 많은 시간을 쓰게 될 수 있다. 강의를 들을 때는 다 이해가 되는 것 같았는데, 막상 혼자 생각해서 코딩을 하려니 잘 안되는 것이다. 이것은 아직 코딩이 익숙하지 않기 때문인데, 처음 코딩을 하는 사람이라면 누구나 겪는 자연스러운 현상이다.
문제를 스스로 풀기 어려운 경우, 너무 고민하기 보다는 먼저 **강의 영상의 문제 풀이 과정을 코드로 따라하면서 이해하자. 반드시 코드로 따라해야 한다.** 그래야 코딩하는 것에 조금씩 익숙해질 수 있다. 그런 다음에 정답을 지우고 스스로 문제를 풀어보면 된다. 참고로 강의를 듣는 시간만큼 문제와 풀이에도 많은 시간을 들어야 제대로 성장할 수 있다!

문제1

다음 코드에 반복해서 나오는 숫자 `4`, `3`을 다른 숫자로 한번에 변경할 수 있도록 다음을 변수 `num1`, `num2`를 사용하도록 변경해보세요.

VarEx1

```
package variable.ex;

public class VarEx1Question {
    public static void main(String[] args) {
        System.out.println(4 + 3);
        System.out.println(4 - 3);
        System.out.println(4 * 3);
    }
}
```

문제1 - 정답

VarEx1

```
package variable.ex;

public class VarEx1 {
    public static void main(String[] args) {
        int num1 = 4;
        int num2 = 3;
```

```
        System.out.println(num1 + num2);
        System.out.println(num1 - num2);
        System.out.println(num1 * num2);
    }
}
```

실행 결과

```
7
1
12
```

문제2

다음과 같은 작업을 수행하는 프로그램을 작성하세요.

클래스 이름은 `VarEx2`라고 적어주세요.

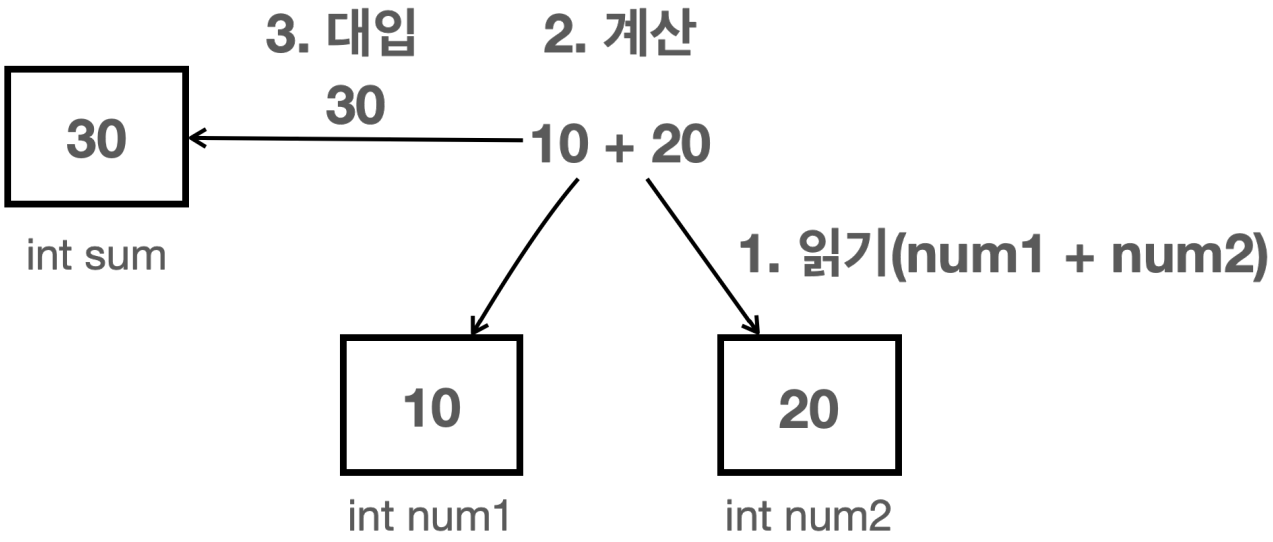
1. 변수 `num1`을 선언하고, 이에 10을 할당하세요.
2. 변수 `num2`를 선언하고, 이에 20을 할당하세요.
3. 두 변수의 합을 구하고, 그 결과를 새로운 변수 `sum`에 저장하세요.
4. `sum`변수의 값을 출력하세요.

문제2 - 정답

VarEx2

```
package variable.ex;

public class VarEx2 {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;
        int sum = num1 + num2;
        System.out.println(sum);
    }
}
```

`int sum = num1 + num2` **계산 과정**




```
30
```

문제3 - long, boolean 데이터 타입

클래스 이름: `VarEx3`

`long` 타입의 변수를 선언하고, 그 변수를 10000000000(백억)으로 초기화한 후 출력하는 프로그램을 작성하세요.

`boolean` 타입의 변수를 선언하고, 그 변수를 `true`로 초기화한 후 출력하는 프로그램을 작성하세요.

정답 - long, boolean 데이터 타입

```
package variable.ex;

public class VarEx3 {
    public static void main(String[] args) {
        long longVar = 10000000000L;
        System.out.println(longVar);

        boolean booleanVar = true;
        System.out.println(booleanVar);
    }
}
```

실행 결과

```
10000000000
true
```

정리