Data Type

Data Type

- Data Type – 자료형
  - ✓ 컴퓨터의 메모리는 유한하고 2진수로 저장해야 하는데 사람은 일반적으로 10진수를 사용하며 데이터의 종류도 정수, 실수, 문자 등으로 다양하므로 단순하게 1가지 저장 방법으로 여러 종류의 데이터를 사용할 수 없음
  - ✓ 데이터 타입은 데이터를 저장하고 읽을 때 어느 정도의 메모리를 할당 받아서 어떤 방식으로 저장하고 어떤 방식으로 해석할 것인지를 결정하는 것
  - ✓ Data Structure(자료 구조)라는 것은 여러 개의 데이터를 어떻게 저장하면 효율적으로 메모리 공간을 사용하고 이용할 것인가 하는 문제

Data Type

- ❖ 데이터의 분류
  - ✓ 변경 여부에 따른 분류로 mutable 과 immutable로 구분
    - Immutable Data
      - ◆ Constant: 변경할 수 없는 데이터
      - ◆ Literal: 프로그램 상에서 입력된 데이터
    - Mutable Data
      - ◆ Variable(변수): 데이터를 저장할 수 있는 메모리 공간에 붙여놓은 이름으로 데이터를 저장하고 있는 공간에 붙여놓은 이름
      - ◆ 변수의 선언(생성)

        DataType 변수이름;

        - Data Type(자료형)은 변수가 가리키는 공간에 저장할 수 있는 데이터의 종류를 의미
        - 변수 이름은 저장 공간을 구분하기 위한 이름으로 변수를 생성하면 프로그래밍 언어는 테이블을 만들어서 참조할 위치와 변수 이름을 연결해 두었다가 프로그래밍에서 변수 이름을 작성하면 그 저장 공간을 이용하도록 함
        - 변수의 이름은 영역 내에서 구분하기 위한 이름이므로 동일한 영역에 동일한 이름의 변수를 2번 이상 만들 수 없음
        - 자바에서 영역 – 클래스, 메서드, 제어문, 예외 처리 구문 등

Data Type

- 데이터의 분류
  - ✓ 저장되는 데이터의 종류에 따른 분류
    - Value Type - 데이터를 직접 가리키는 방식(일반적으로 하나의 데이터)
    - Reference Type – 참조형이라고 번역을 하는데 데이터가 위치 한 곳의 시작 지점을 저장하는 방법(일반적으로 0개 이상의 데이터)
  - ✓ 저장할 수 있는 데이터 개수에 따른 분류
    - Scala Type: 1개의 데이터를 저장하는 타입
    - Vector Type: 0개 이상의 데이터를 저장하고 그 데이터들의 시작 위치를 저장하는 타입으로 Collection 이라고도 함

Data Type

- 데이터의 분류
  - ✓ Data Type을 기재하는 방법에 따른 분류: 정적 타입(Static Type) 과 동적 타입(Dynamic Type)
    - 정적 타입
      - 변수를 선언할 때 Data Type을 결정하는 것으로 Java 나 C언어, Swift, Kotlin 등의 언어가 사용하는 방식
      - 컴파일 할 때 저장하려고 하는 데이터의 Type을 확인한 후 가리키고자 하는 변수의 Data Type 과 다르면 에러를 발생시키며 코딩 시에 모든 변수, 메서드의 매개변수, 리턴 값에 대한 타입을 지정해야 함
      - 실행하기 전에 컴파일을 수행하기 때문에 타입과 관련된 버그를 줄일 수 있음
    - 동적 타입
      - 변수를 선언할 때는 하나의 Data Type 으로 선언하고 대입할 때 Data Type 추론을 통해서 Data Type이 결정되는 형태로 Javascript 나 R, Python, Ruby 등의 언어가 사용하는 방식
      - 타입을 명시하지 않기 때문에 소스 코드의 양이 적을 가능성이 높고 자유도가 높다는 장점은 있지만 변수에 어떤 타입의 데이터가 저장되어 있는지 정해지지 않으므로 실행 시에 문제가 발생할 수 있고 IDE의 도움을 활용하기 어렵다는 단점이 있음

Data Type

- 자바의 자료형
  - ✓ 자바의 Data Type은 Primitive Type(기본형 - 8가지) 과 NonPrimitive Type 으로 나눔
  - ✓ 식별자에 데이터의 참조 저장

    식별자 = 데이터 또는 데이터를 저장한 다른 변수 또는 데이터를 리턴하는 메서드 호출;
  - ✓ 식별자에 데이터의 참조를 저장하는 작업을 초기화(Initialize) 라고 하기도 함
  - ✓ 식별자에 참조하는 데이터 사용: 변수의 이름만으로 사용이 가능한데 선언하고 데이터를 할당한 후에 사용하는 경우 Primitive Type은 데이터를 바로 접근할 수 있지만 NonPrimitive Type은 바로 접근하면 데이터의 참조(데이터가 저장되는 곳을 구분하기 위한 코드 – Hash Code)가 되고 인덱스 나 이름을 이용해서 세부 데이터에 접근
  - ✓ 식별자를 선언할 때 와 = 의 왼쪽에 식별자의 이름이 사용되면 식별자가 만들어진 메모리 공간(참조)을 의미하고 그 이외의 경우에는 식별자가 사용할 수 있는 데이터(값 이나 참조)를 의미

Data Type

- 식별자의 명명 규칙
  - ✓ 클래스 와 속성 또는 변수는 Camel Case 상수는 Snake Case를 권장
  - ✓ 자바의 예약어를 식별자 이름으로 사용하는 것은 안됨
  - ✓ 첫번째 글자는 문자(영문, 한글) 나 $, _로 시작
  - ✓ 영문 대소문자 구분
  - ✓ 변수와 메서드의 경우 첫번째 문자는 영문 소문자로 하는 것을 권장하고 이후에 다른 단어가 붙는 경우에는 대문자로 시작하며 클래스 이름은 첫 글자를 대문자로 시작 – Camel Case

    area, length, nArea
  - ✓ 식별자 이름은 한글로 선언하는 것이 가능하지만 실제 사용은 거의 하지 않음

    ```
    int 박문석;
    박문석 = 10;
    System.out.println("박문석=" + 박문석);
    ```
  - ✓ 식별자 이름은 이름을 보고 용도를 파악할 수 있도록 정하는 것을 권장
  - ✓ 변하지 않는 데이터(상수)는 모두 대문자로 하고 단어를 _로 구분 – Snake Case
  - ✓ 변수 나 상수는 명사 그리고 메서드는 동사로 명명하는 것을 권장

Data Type

- 변수의 유효 범위에 따른 분류
  - ✓ Local Variable - Stack이라는 일시적인 공간(자동으로 소멸되는 영역)에 생성
    - 메서드 내에서 선언된 변수
    - 만들어진 영역 내에서만 사용이 가능
    - 자동 초기화가 되지 않음
    - 데이터를 일시적으로 저장할 목적으로 생성
  - ✓ Member Variable – 개발자가 필요에 따라 삭제할 수 있는 영역에 만들어짐
    - 메서드 바깥에 있는 클래스 영역에 선언해서 인스턴스가 사용하는 변수
    - 자동 초기화
    - 다른 인스턴스와 구분해서 저장해야 하는 데이터를 저장하기 위한 목적으로 생성
  - ✓ Static Variable(Class Variable) – 삭제되지 않는 영역에 만들어짐
    - 메서드 바깥에 있는 클래스 영역에 static 키워드와 선언
    - 자동 초기화
    - 프로그램 종료 시까지 유지되는 변수
    - 데이터 공유를 목적으로 생성

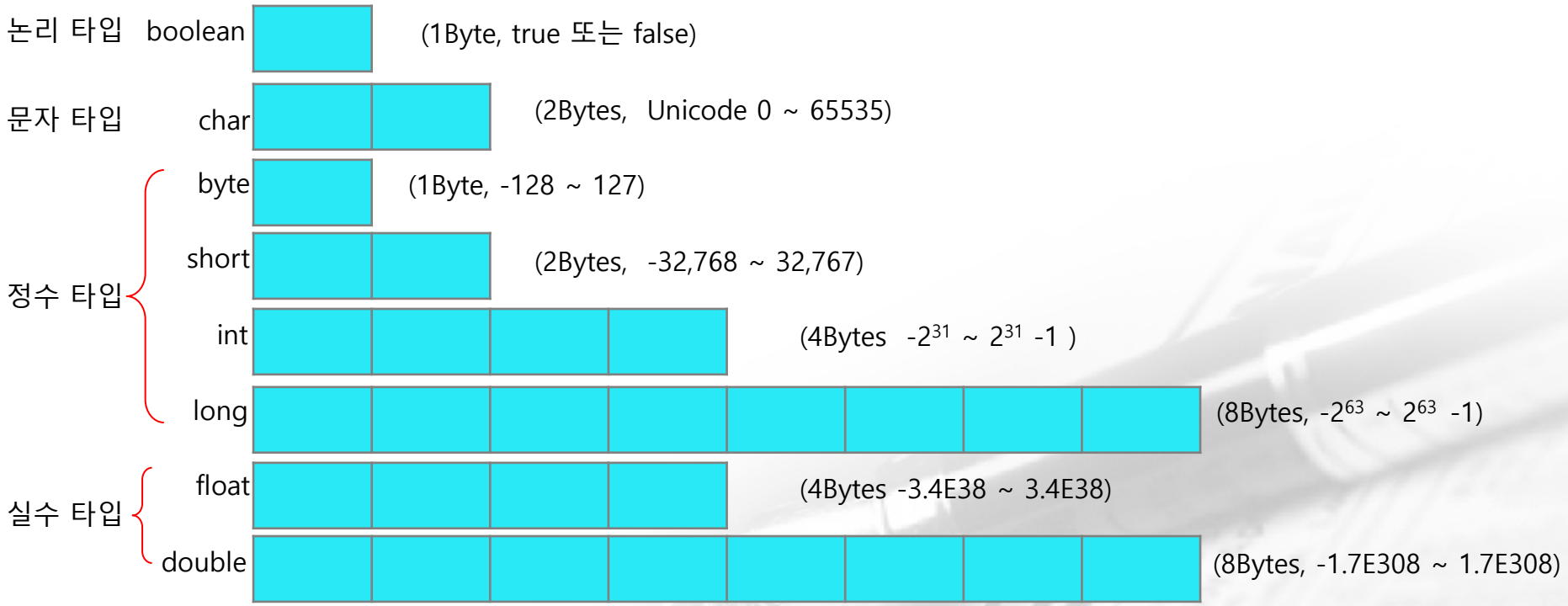
Data Type

- primitive type
  - ✓ primitive 타입은 실행의 효율성을 위해서 사용
  - ✓ 하나의 데이터를 저장하기 위한 Data Type
  - ✓ boolean, byte, short, char(2byte), int, long, float, double
  - ✓ boolean을 제외하고는 숫자를 저장
  - ✓ boolean을 제외하고는 서로 간에 Data Type을 변경해서 대입하는 것이 가능

구분	데이터형	크기(byte)	범위
정수	byte	1	-2^{7} ~ $2^{7}-1$
	short	2	-2^{15} ~ $2^{15}-1$
	int	4	-2^{31} ~ $2^{31}-1$
	long	8	-2^{63} ~ $2^{63}-1$
문자형	char	2	영문자 1자 ' a ' -> ASCII 0~65535 Java는 유니코드를 사용 함
실수형	float	4	유효자리 7자리(정밀도)
	double	8	유효자리 16자리(정밀도)
불형	boolean	1	true 또는 false

Data Type

- ❖ primitive type
  - ✓ 특징
    - 메모리 크기가 정해져 있음
    - 숫자의 경우 데이터의 표현 범위가 더 큰 데이터를 표현 범위가 작은 메모리 공간에 대입하는 것은 허용되지 않음



Data Type

- NonPrimitive – 데이터가 있는 곳을 대표하는 참조를 저장
  - ✓ 0개 이상의 데이터를 하나의 이름으로 묶어서 사용하기 위한 Data Type
  - ✓ 세부 데이터를 호출할 때 . 이나 [인덱스]를 이용해서 접근
  - ✓ Wrapper 클래스나 String, Date 클래스의 인스턴스는 참조형 이지만 하나의 데이터만 저장하는 것으로 간주해서 Primitive 타입처럼 toString 메서드를 호출하면 저장하고 있는 데이터를 문자열로 리턴
  - ✓ 사용자 메모리 영역을 할당받기 위해서는 new 라는 연산자를 이용

    new 생성자();
  - ✓ C언어에서의 포인터와 유사한 개념
  - ✓ 참조형 Literal로 null 이 존재하는데 null은 참조하는 데이터가 없다 라는 의미
  - ✓ 종류
    - Array
    - Class로 부터 만들어진 Instance, Interface
    - Enum
    - Annotation

Primitive Type

- 정수
  - ✓ 첫번째 비트를 부호로 해서 2진수로 변환해서 저장
  - ✓ 음수는 2의 보수 방법을 이용
  - ✓ 정수 Literal
    - 기준형(일반적인 정수를 기재할 때 만들어지는 자료형)은 int

      10
    - 뒤에 L이나 l을 입력하면 long형 정수 Literal

      10L
    - 천 단위 구분 기호인 ,는 _ 로 표현 – 1.7 버전 이상에서 가능

      123_456_789L
    - 8진수: 0으로 시작하는 숫자는 8진수

      015 // 10진수로 13
    - 16진수 : 0x로 시작하는 숫자는 16진수

      0x15 // 10진수로 21
    - 10진수 : 0이나 0x로 시작하지 않는 숫자는 10진수

      15, 3, 20, 55, 88

Primitive Type

- 정수
  - ✓ 정수의 표현

```
public class SystemWord
{
    public static void main(String args[])
    {
            //10진수 , 8진수, 16진수 표현
            System.out.println(12 + 017 + 0xb); //12 + 15 + 11
            int i = 020;
            int j = 0xf;
            System.out.println("i=" + i + '₩t' + "j=" +j);   // 16 15
    }
}
```

```
38
i=16j=15
```

Primitive Type

- 실수
  - ✓ 부호, 지수, 가수부를 일정한 크기로 분할해서 각각을 저장하고 해석

    3.14 => double
  - ✓ 실수 Literal은 기본적으로 double 인데 뒤에 f를 붙이면 float이고 d를 붙이면 double
  - ✓ 영문 소문자 e 나 대문자 E 와 함께 표현하는 것이 가능
  - ✓ 실제 저장이 될 때는 1(부호), 001(지수), 0.314(가수)의 형태로 저장

    3.14d =>+0.314E001

    3.14f
  - ✓ 부호 비트는 무조건 1비트이고 float은 지수가 7비트이고 double은 10비트를 사용하고 나머지는 가수를 저장

Primitive Type

- 실수
  - ✓ 정수 와 실수 변수 사용

```
public class Variable
{
    public static void main(String args[])
    {
            //정수를 저장할 수 있는 변수를 선언
            int a;
            //정수 변수에 값을 대입
            a = 10;
            System.out.println("a=" + a);

            //정수 변수에 값을 대입 - 기존의 값 대신에 새로운 값이 대입
            a = 20;
            System.out.println("a=" + a);

            //정수 변수 값을 더해서 출력
            System.out.println("a + a= " + (a+a));

            //선언과 동시에 초기화
            int b = 30;
            System.out.println("b=" + b);
```

Primitive Type

- 실수
  - ✓ 정수 와 실수 변수 사용

```
        //실수 변수를 생성하고 값을 저장
        float f;
        f = 0.123456789f;
        System.out.println("f = " + f);

        double d;
        d = 0.12345678901234567;
        System.out.println("d = " + d);
    }
}
```

```
a=10
a=20
a + a= 40
b=30
f = 0.12345679
d = 0.12345678901234566
```

Primitive Type

- 문자(Character)
  - ✓ Java는 하나의 문자를 2byte unicode를 사용해서 표현하며 문자의 코드를 저장하고 있다가 출력을 할 때 코드에 해당하는 문자를 출력
  - ✓ char 도 정수
  - ✓ 유니코드는 음수가 없으므로 char에 음수를 저장할 수 없음
  - ✓ 저장 방법
    - ASCII 코드 이용: 0 ~ 127
      - 숫자 0은 48 숫자 9는 57
      - 문자 A는 65, a는 97
    - 한글은 44032 ~ 55203
    - 문자 상수 직접 대입: '문자';
    - ₩다음에 숫자는 8진수로서 0 ~ 337사이의 8진수만 가능

      ₩102 -> 문자 'B'를 나타내는 8진수

      ₩337 -> 문자 'β'를 나타내는 8진수
    - ₩u다음에 4자리 16진수, 2 바이트의 유니코드(unicode)

      ₩u0041 -> 문자 'A'의 유니코드

      ₩uAC00 -> 한글 가의 유니코드

      ₩uae00 -> 한글 '글'의 유니코드(ae00)

Primitive Type

- ❖ 문자(Character)
  - ✓ 제어 문자(Control Character)
    - 자바가 제공하는 특별한 의미를 저장하고 있는 문자 Literal
    - 자바는 ₩ 다음에 하나의 문자가 오면 제어 문자로 판단

₩t: 탭	₩n: 줄 바꿈	₩r: 첫 칸으로 이동
₩b: back space	₩f: 페이지 바꿈	₩숫자: 8진수
₩u숫자: 16진수	₩₩: ₩문자	₩',₩": ' 또는 "

Primitive Type

- 문자(Character)
  - ✓ 문자 변수 와 제어 문자

```
public class Char{
    public static void main(String[] args){
        //문자 변수를 선언하고 값을 ASCII 코드 값으로 대입
        char ch = 97;
        System.out.println("ch = " + ch);
        //문자 Literal을 직접 대입
        ch = 'a';
        System.out.println("ch = " + ch);
        //유니코드를 대입
        ch = '₩u0061';
        System.out.println("ch = " + ch);
        //제어문자 사용
        System.out.print("여기는 첫째줄 ");
        System.out.print("입니다.₩n");
        System.out.println("여기는 둘째 줄 입니다.");
        System.out.print("국어₩t영어₩t수학₩n");
        System.out.println("---------------------");
        System.out.println("80₩t50₩t100");
    }
}
```

Primitive Type

- 문자(Character)
  - ✓ 문자 데이터를 메모리에 저장할 때 메모리에는 문자를 저장할 수 없기 때문에 코드 값을 저장
  - ✓ 문자 변수는 정수로 취급되므로 정수나 실수 데이터와 연산 가능
  - ✓ char ch = 'A'; 라는 문장이 있다면 ch에는 65라는 숫자 값이 저장되고 출력할 때 문자 변수이므로 65에 해당하는 A라는 문자를 출력
  - ✓ ch + 1은 'A' + 1이 아니고 65+1이 되서 66이라는 숫자
  - ✓ 정수를 문자 변수에 저장한 다음 출력하면 문자로 출력

Primitive Type

- 문자(Character)
  - ✓ 문자 데이터를 메모리에 저장할 때 메모리에는 문자를 저장할 수 없기 때문에 코드 값을 저장

```
class CharOp {
    public static void main(String args[]) {
        char ch = 'A';
        System.out.println("ch = " + ch);
        //문자 A에 1을 더해서 출력하므로 66
        System.out.println("ch + 1 = " + (ch + 1));

        //66을 문자 변수에 저장해서 출력하므로 B
        char ch1 = 'A' + 1;
        System.out.println("ch + 1 = " + ch1);

        //13은 Enter
        ch = 13;
        System.out.print("ASCII Code 13 = " + ch);
        System.out.print("한 줄이 떨어져서 출력될 것입니다");
    }
}
```

```
ch = A
ch + 1 = 66
ch + 1 = B
ASCII Code 13 =
한 줄이 떨어져서 출력될 것입니다
```

Primitive Type

- Overflow & Underflow
  - ✓ Overflow: 데이터 타입이 저장할 수 있는 값의 범위를 초과한 경우로 가장 작은 숫자로 돌아가서 다시 올라가게 됨
  - ✓ Underflow: 데이터 타입이 저장할 수 있는 값의 범위보다 작은 값을 저장 한 경우로 가장 큰 숫자로 돌아가서 다시 내려가게 됨

Primitive Type

- Primitive Type
  - ✓ Overflow & Underflow

```
class OverUnderFlow {
    public static void main(String args[]) {
            // int 가 저장할 수 있는 최대값을 저장
            int num1 = 2147483647;
            // 최대값보다 1이 더 큰 수를 강제 형 변환을 통해서 저장
            int num2 = (int) 2147483648L;
            System.out.println("num1 = " + num1);
            System.out.println("num2 = " + num2);


            System.out.println("==========================================
    =====");
            // int가 저장할 수 있는 최소값을 저장
            num1 = -2147483648;
            // 최소값보다 1이 더 작은 수를 강제 형 변환을 통해서 저장
            num2 = (int) -2147483649L;
            System.out.println("num1 = " + num1);
            System.out.println("num2 = " + num2);
    }
}
```

```
num1 = 2147483647
num2 = -2147483648
num1 = -2147483648
num2 = 2147483647
```

Primitive Type

- boolean
  - ✓ boolean은 true 또는 false 2가지 중 하나를 저장하는 Data Type 으로 1byte를 할당
  - ✓ Java API에서 제공하는 메서드의 이름이 is로 시작하면 메서드의 리턴 타입은 boolean
  - ✓ 다른 언어에서는 0이 아닌 숫자 나 데이터가 존재하는 데이터의 모임은 true로 간주하고 그렇지 않은 경우는 false로 간주함

Primitive Type

- boolean
  - ✓ boolean 변수은 true 또는 false 2가지 중 하나를 저장하는 Data Type 으로 1byte 를 할당

```
public class BooleanTest {
    public static void main(String args[]) {
            int num = 10 + 20;
            // boolean 타입의 변수를 선언
            boolean pan;
            // boolean 타입의 변수에 값 대입
            pan = num > 10;
            System.out.println("pan=" + pan);
    }
}
```

```
pan=true
```

String

- String 클래스
  - ✓ 자바에서 문자열을 저장할 때 사용하는 Data Type
  - ✓ 다른 클래스와 다르게 생성자를 사용하지 않고 문자열 Literal을 직접 대입해서 생성하는 것이 가능

    String str = "문자열";
  - ✓ 문자열 Literal은 "" 안에 기재
  - ✓ 출력하는 메서드에 Premitive Type 이외의 데이터의 참조를 저장한 변수의 이름을 대입하면 toString() 메서드의 결과를 출력

Literal

- Literal
  - ✓ Literal은 메모리 영역에 한 번만 저장
  - ✓ 동일한 Literal을 다음에 사용하게 되면 새로 저장하는 것이 아니고 이전에 저장된 Literal을 이용
  - ✓ 32라는 숫자를 2개의 int 변수에 저장하면 2개의 변수는 동일한 영역의 데이터를 가리킴
  - ✓ 32L 과 32는 다른 곳에 저장되는데 이유는 Data Type이 다르기 때문
  - ✓ System.identityHashCode( ) 메서드는 JVM 메모리 영역에 저장된 데이터의 위치를 반환하는 역할을 하는데 이때 반환되는 값이 해시 코드(HashCode): JVM 메모리의 참조명
  - ✓ 리턴되는 참조는 실제 메모리의 참조가 아니며 데이터를 저장할 때 구분하기 위한 코드

Literal

- Literal
  - ✓ Literal 저장

```
public class LiteralHashcode {

    public static void main(String[] args) {
        int a = 10;
        int b = 10;
        long c = 10L;

        System.out.println("a:" + System.identityHashCode(a));
        // 동일한 상수 값을 가리키므로 동일한 해시코드 출력
        System.out.println("b:" + System.identityHashCode(b));
        // 값은 같지만 데이터의 타입이 다르므로 다른 해시코드 출력
        System.out.println("c:" + System.identityHashCode(c));

    }
}
```

```
a:705927765
b:705927765
c:366712642
```

서식을 이용한 데이터 출력

- System.out.printf("format", argument);
  - ✓ format은 서식이고 argument는 서식이 적용되서 출력되는 데이터
  - ✓ format은 1번만 나와야 하고 argument는 생략이 가능하고 format에 나온 데이터 서식만큼 반복해서 나올 수 있음

코드	데이터 형식	사용 예제	출력 결과
%d	정수(10진수)	printf("%d",29);	29
%x	정수(16진수)	printf("%x %x", 29, 32);	1D 20
%o	정수(8진수)	printf("%o",29);	35
%f	실수	printf("%.2f",123.4567);	123.46
%e	지수형 실수	printf("%e", 874.9163);	8.749163e+02
%c	문자	printf("%c",'y');	y
%s	문자열	printf("%s","Hello");	Hello
%b	boolean		

  - ✓ %**숫자**서식: 숫자만큼 자릿수를 확보해서 출력하는데 자리가 남으면 빈칸으로 두고 부족하면 전체 출력
  - ✓ %0숫자d: 남는 자리는 0으로 채움
  - ✓ %.숫자f: 소수점 이하 자리를 숫자 만큼만 출력(반올림)

서식을 이용한 데이터 출력

- System.out.printf("format", argument);
  - ✓ 서식을 이용한 출력

```
public class FormatDisplay {
    public static void main(String[] args) {
            System.out.printf("%10d₩n", 24);
            System.out.printf("%010d₩n", 24);
            System.out.printf("%f₩n", 24.12345678);
            System.out.printf("%.3f₩n", 24.12345678);
    }
}
```

```
        24
0000000024
24.123457
24.123
```

연습문제

- primitive type 과 reference type의 차이는?
- primitive type 8개는?
- 다음 중 에러가 발생하는 경우는?

```
byte byteValue=10;
char charValue=10;

1)int intValue = byteValue;
2)int intValue = charValue;
3)short shortValue = charValue;
4)double doubleValue = byteValue;
```

- 자신의 번호와 이름과 나이 그리고 키를 저장할 수 있는 변수를 생성하고 값을 대입한 후 출력하는 프로그램을 작성하시오.
  - ✓ 번호는 21억이 되지 않는 정수
  - ✓ 이름은 문자열
  - ✓ 나이는 21억이 되지 않는 정수
  - ✓ 키는 소수 첫째자리까지 저장할 수 있는 실수