기본 API

▶ Object 클래스 - 모든 클래스의 최상위 클래스

✓ java.lang 패키지

- 프로그래밍시 import 하지 않아도 자동으로 imort됨
- import.java.lang.*;
- 많이 사용하는 기본 클래스들이 속한 패키지
- String, Integer, System...

✓ 모든 클래스는 Object 클래스를 상속 받는다

- java.lang.Object 클래스
- 모든 클래스의 최상위 클래스는
- 모든 클래스는 Object에서 상속받고, Object 클래스의 메서드 중 일부는 재정의해서 사용할 수 있음
- 컴파일러가 extends Object를 추가함

  class Student => class Student extends Object

▶ Object 클래스 - 모든 클래스의 최상위 클래스

✓ toString() 메서드

- 객체의 정보를 String으로 바꾸어서 사용할 때 쓰임
- String이나 Integer 클래스는 이미 재정의 되어 있음
- toString()메서드 재정의 예

✓ equals() 메서드

두 인스턴스의 주소 값을 비교하여 true/false를 반환

- 재정의 하여 두 인스턴스가 논리적으로 동일함의 여부를 구현함
- 인스턴스가 다르더라도 논리적으로 동일한 경우 true를 반환하도록 재정의 할 수 있음
  (같은 학번, 같은 사번, 같은 아이디의 회원...)

▶ Object 클래스 - 모든 클래스의 최상위 클래스

✓ clone() 메서드

- 객체의 원본을 복제하는데 사용하는 메서드
- 생성과정의 복잡한 과정을 반복하지 않고 복제 할 수 있음
- clone()메서드를 사용하면 객체의 정보(멤버 변수 값등...)가 동일한 또 다른 인스턴스가 생성되는 것이므로, 객체 지향 프로그램에서의 정보 은닉, 객체 보호의 관점에서 위배될 수 있음
- 해당 클래스의 clone() 메서드의 사용을 허용한다는 의미로 cloneable 인터페이스를 명시해 줌

▶ String 관련 클래스

✓ String 클래스

String str1 = new String("abc");

String str2 = "abc";

- 힙 메모리는 생성될때마다 다른 주소 값을 가지지만, 상수 풀의 문자열은 모두 같은 주소 값을 가짐, 따라서 equls 비교를 해줘야 한다.
- 문자열 값 수정 불가능, immutable(불변)
- 수정 시 수정된 문자열이 새로 할당 되어 새 주소를 넘김(메모리 낭비가 발생할 가능성)

▶ String 관련 클래스

✓ StringBuffer 클래스

- 내부적으로 가변적인 char[]를 멤버 변수로 가짐
- 문자열 값 수정 가능, mutable(가변)
- 수정, 삭제 등이 기존 문자열에 수정되어 적용
- 기본 16문자 크기로 지정된 버퍼를 이용하며 크기 증가 가능
- 쓰레드 safe기능 제공(성능 저하 요인)

✓ StringBuilder 클래스

- StringBuffer와 동일하나 쓰레드 safe기능을 제공하지 않음
- 단인 쓰레드 프로그램에서는 StringBuilder 사용을 권장

▶ String 관련 클래스

✓ text block 사용하기 (java 13)

- 문자열을 """ """ 사이에 이어서 만들 수 있음
- html, json 문자열을 만드는데 유용하게 사용할 수 있음

▶ Wrapper 클래스

Primitive Data Type을 객체화 해주는 클래스

Primitive Data Type	Wrapper Class
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

기본 자료형의 값을 감싸는 래퍼 클래스

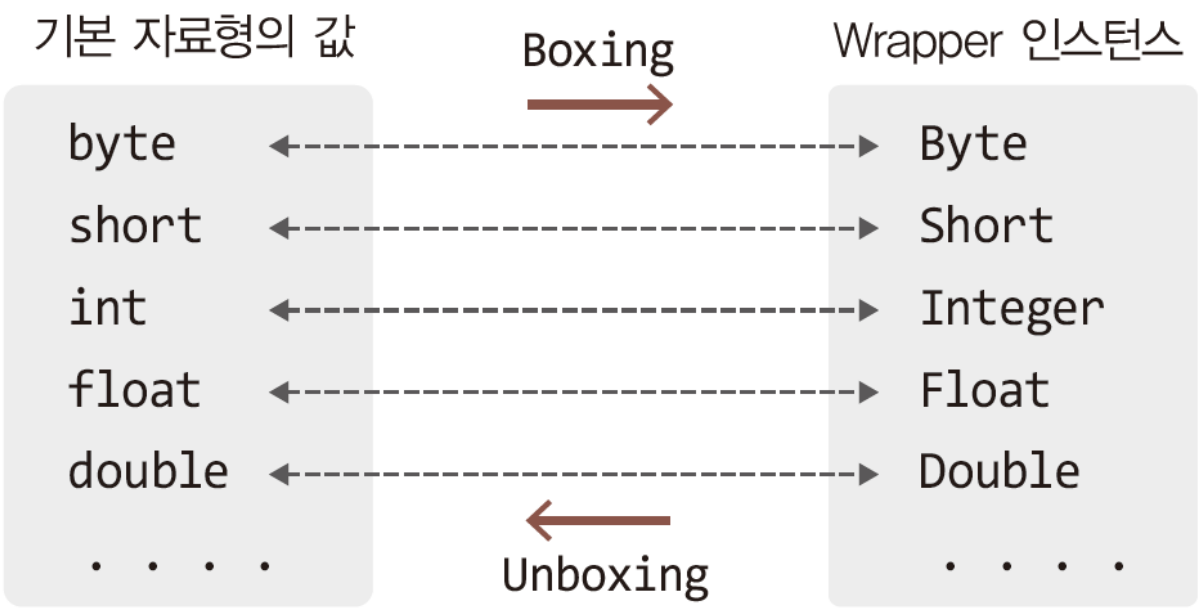
```
class UseWrapperClass {
   public static void showData(Object obj) {
      System.out.println(obj);
   }

   public static void main(String[] args) {
      Integer iInst = new Integer(3);
      showData(iInst);
      showData(new Double(7.15));
   }
}
```

인스턴스를 요구하는 메소드

이 메소드를 통해서 정수나 실수를 출력하려면 해당 값을 인스턴스화 해야 한다.

이렇듯 기본 자료형의 값을 인스턴스로 감싸는 목적의 클래스를 가리켜 래퍼 클래스라 한다.

래퍼 클래스의 두 가지 기능



박싱과 언박싱 예

```
public static void main(String[] args) {
   Integer iObj = new Integer(10);    // 박싱
   Double dObj = new Double(3.14);    // 박싱
   . . . .

   int num1 = iObj.intValue();           // 언박싱
   double num2 = dObj.doubleValue();     // 언박싱
   . . . .

   // 래퍼 인스턴스 값의 증가 방법
   iObj = new Integer(iObj.intValue() + 10);
   dObj = new Double(dObj.doubleValue() + 1.2);
   . . . .
}
```

언박싱 메소드의 이름

Boolean	public boolean booleanValue()
Character	public char charValue()
Integer	public int intValue()
Long	public long longValue()
Double	public double doubleValue()

오토 박싱과 오토 언박싱

```
class AutoBoxingUnboxing {
   public static void main(String[] args) {
      Integer iObj = 10;      // 오토 박싱 진행
      Double dObj = 3.14;      // 오토 박싱 진행
      . . . .   인스턴스가 와야 할 위치에 기본 자료형 값이 오면 오토 박싱 진행

      int num1 = iObj;      // 오토 언박싱 진행
      double num2 = dObj;      // 오토 언박싱 진행
      . . . .   기본 자료형 값이 와야 할 위치에 인스턴스가 오면 오토 언박싱 진행
   }
}
```

오토 박싱, 오토 언박싱의 또 다른 예

```
public static void main(String[] args) {
   Integer num = 10;
   num++;     // new Integer(num.intValue() + 1);
   . . . .       오토 박싱과 오토 언박싱 동시에 진행!

   num += 3;     // new Integer(num.intValue() + 3);
   . . . .         오토 박싱과 오토 언박싱 동시에 진행!

   int r = num + 5;   // 오토 언박싱 진행
   Integer rObj = num - 5;   // 오토 언박싱 진행
   . . . .
}
```

▶ Wrapper 클래스

✓ String을 기본 자료형으로 바꾸기

```
byte b = Byte.parseByte("1");
short s = Short.parseShort("2");
int i = Integer.parseInt("3");
long l = Long.parseLong("4");
float f = Float.parseFloat("0.1");
double d = Double.parseDouble("0.2");
boolean bool = Boolean.parseBoolean("true");


char c = "abc".charAt(0);
```

▶ Wrapper 클래스

✓ 기본 자료형을 String으로 바꾸기

```
String b = Byte.valueOf((byte)1).toString();
String s = Short.valueOf((short)2).toString();
String i = Integer.valueOf(3).toString();
String l = Long.valueOf(4L).toString();
String f = Float.valueOf(0.1f).toString();
String d = Double.valueOf(0.2).toString();
String bool = Boolean.valueOf(true).toString();
String ch = Character.valueOf('a').toString();
```

▶ 날짜 관련 클래스

✓ Date 클래스

시스템으로부터 현재 날짜, 시간 정보를 가져와서 다룰 수 있게 만들어진 클래스

생성자 2개만 사용 가능하고 나머지는 모두 deprecated

✓ 예시

```
Date today = new Date();
//시스템으로부터 현재 날짜, 시간 정보를 가져와 기본 값으로 사용

Date when = new Date(123456798L);
//long형 정수 값을 가지고 날짜 시간 계산
//1970년 1월 1일 0시 0분 0초를 기준으로 함

System.out.println (today);
//현재시간을 보여준다.
```