JAVA

- 자바의 다양한 기본 클래스

Wrapper 클래스

기본 자료형 데이터를 감싸는 Wrapper 클래스

wrapper클래스

- 기본형을 클래스로 정의한 것. 기본형 값도 객체로 다뤄져야 할 때가 있다.

기본형	래퍼클래스	생성자	활용예
boolean	Boolean	Boolean(boolean value) Boolean(String s)	Boolean b = new Boolean(true); Boolean b2 = new Boolean("true");
char	**Character**	Character(char value)	Character c = new Character('a');
byte	Byte	Byte(byte value) Byte(String s)	Byte b = new Byte(10); Byte b2 = new Byte("10");
short	Short	Short(short value) Short(String s)	Short s = new Short(10); Short s2 = new Short("10");
int	**Integer**	Integer(int value) Integer(String s)	Integer i = new Integer(100); Integer i2 = new Integer("100");
long	Long	Long(long value) Long(String s)	Long l = new Long(100); Long l2 = new Long("100");
float	Float	Float(double value) Float(float value) Float(String s)	Float f = new Float(1.0); Float f2 = new Float(1.0f); Float f3 = new Float("1.0f");
double	Double	Double(double value) Double(String s)	Double d = new Double(1.0); Double d2 = new Double("1.0");

기본 자료형 데이터를 감싸는 Wrapper 클래스

wrapper클래스

- 내부적으로 기본형(primitive type) 변수를 가지고 있다.

```
public final class Integer extends Number implements Comparable {
   ...
   private int value;
```

- 값을 비교하도록 equals()가 오버라이딩 되어 있다.

```
Integer i  = new Integer(100);
Integer i2 = new Integer(100);
System.out.println(i==i2);          // false
System.out.println(i.equals(i2));   // true
```

기본 자료형 데이터를 감싸는 Wrapper 클래스

```
class WrapperEx1
{
	public static void main(String[] args)
	{
		Integer i  = new Integer(100);
		Integer i2 = new Integer(100);

		System.out.println("i==i2?"+(i==i2));
		System.out.println("i.equals(i2)?"+i.equals(i2));
		System.out.println("i.toString()="+i.toString());

		System.out.println("MAX_VALUE="+Integer.MAX_VALUE);
		System.out.println("MIN_VALUE="+Integer.MIN_VALUE);
		System.out.println("SIZE="+Integer.SIZE+" bits");
		System.out.println("TYPE="+Integer.TYPE);
	}
}
```

기본 자료형 데이터를 감싸는 Wrapper 클래스

```
        int i  = new Integer("100").intValue();
        int i2 = Integer.parseInt("100");
        Integer  i3 = Integer.valueOf("100");
        int i4 = Integer.parseInt("100",2);
        int i5 = Integer.parseInt("100",8);
        int i6 = Integer.parseInt("100",16);
        int i7 = Integer.parseInt("FF", 16);
//      int i8 = Integer.parseInt("FF");  // NumberFormatException발생
        Integer i9 = Integer.valueOf("100",2);
        Integer i10 = Integer.valueOf("100",8);
        Integer i11 = Integer.valueOf("100",16);
        Integer i12 = Integer.valueOf("FF",16);
//      Integer i13 = Integer.valueOf("FF"); // NumberFormatException발생
        System.out.println(i);
        System.out.println(i2);
        System.out.println(i3);
        System.out.println("100(2) -> "+i4);
        System.out.println("100(8) -> "+i5);
        System.out.println("100(16)-> "+i6);
        System.out.println("FF(16) -> "+i7);
        System.out.println("100(2) -> "+i9);
        System.out.println("100(8) -> "+i10);
        System.out.println("100(16)-> "+i11);
        System.out.println("FF(16) -> "+i12);
```

자바에서 제공하는 Wrapper 클래스

Boolean	Boolean(boolean value)
Character	Character(char value)
Byte	Byte(byte value)
Short	Short(short value)
Integer	Integer(int value)
Long	Long(long value)
Float	Float(float value)
	Float(double value)

순전히 기본 자료형 데이터의 표현이 목적 이라면, 별도의 클래스 정의 없이 제공되는 Wrapper 클래스를 사용하면 된다!

위의 클래스 이외에도 문자열 기반으로 정의된 Wrapper 클래스도 존재하기 때문에 다음과 같이 인스턴스 생성도 가능.
단,Character 클래스제외!

Integer num1=new Integer("240")
Double num2=new Double("12.257")

```
public static void main(String[] args)
{
    Integer intInst=new Integer(3);
    showData(intInst);
    showData(new Integer(7));
}
```

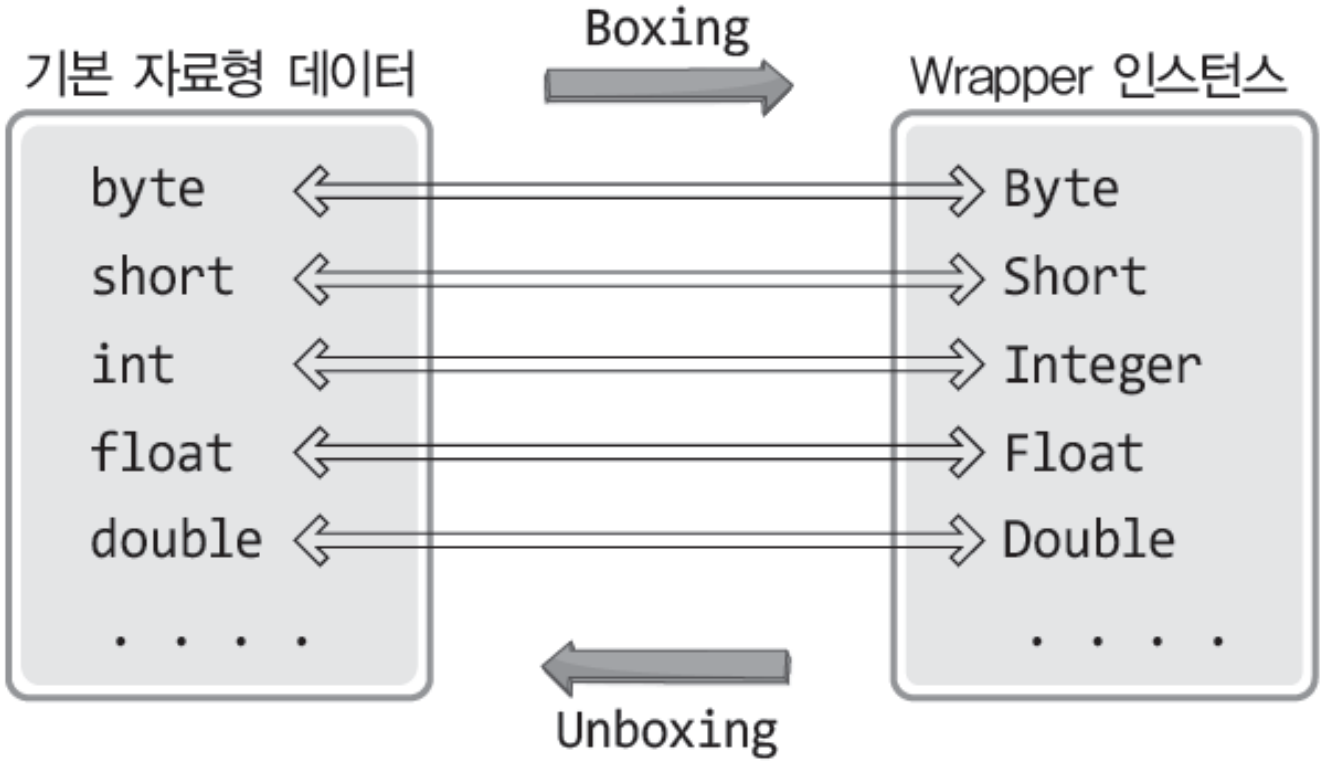
자바에서 제공하는 Wrapper 클래스

Boxing

*** 기본 자료형 데이터를 Wrapper 인스턴스로 감싸는 것!**

UnBoxing

*** Wrapper 인스턴스에 저장된 데이터를 꺼내는 것!**



자바에서 제공하는 Wrapper 클래스

```
class BoxingUnboxing
{
	public static void main(String[] args)
	{
		Integer iValue=new Integer(10);
		Double dValue=new Double(3.14);

		System.out.println(iValue);
		System.out.println(dValue);

		iValue=new Integer(iValue.intValue()+10);
		dValue=new Double(dValue.doubleValue()+1.2);

		System.out.println(iValue);
		System.out.println(dValue);
	}
}
```

본래 Wrapper 클래스는 산술연산을 고려해서 정의되는 클래스가 아니다.

따라서 Wrapper 인스턴스를 대상으로 산술연산을 할 경우에는 코드가 복잡해진다.

Auto Boxing & Auto Unboxing

Auto Boxing

*** 기본 자료형 데이터가 자동으로 Wrapper 인스턴스로 감싸지는 것!**

Auto UnBoxing

Wrapper 인스턴스에 저장된 데이터가 자동으로 꺼내지는 것!

기본 자료형 데이터가 와야 하는데, Wrapper 인스턴스가 있다면,

Auto Unboxing!

인스턴스가 와야하는데, 기본 자료형 데이터가 있다면,

Auto Boxing!

Auto Boxing & Auto Unboxing

```
class AutoBoxingUnboxing
{
	public static void main(String[] args)
	{
		Integer iValue=10;
		Double dValue=3.14;

		System.out.println(iValue);
		System.out.println(dValue);

		int num1=iValue;
		double num2=dValue;
		System.out.println(num1);
		System.out.println(num2);
	}
}
```

BigInteger클래스와 BigDecimal클래스

매우 큰 정수의 표현을 위한 BigInteger 클래스

```
class SoBigInteger
{
	public static void main(String[] args)
	{
		System.out.println("최대 정수: " + Long.MAX_VALUE);
		System.out.println("최소 정수: " + Long.MIN_VALUE);

		BigInteger bigValue1=new BigInteger("10000000000000000000");
		BigInteger bigValue2=new BigInteger("-99999999999999999999");

		BigInteger addResult=bigValue1.add(bigValue2);
		BigInteger mulResult=bigValue1.multiply(bigValue2);

		System.out.println("큰 수의 덧셈결과: "+addResult);
		System.out.println("큰 수의 곱셈결과: "+mulResult);
	}
}
```

큰 정수를 문자열로 표현한 이유는 숫자로 표현이 불가능하기 때문이다!

기본 자료형의 범위를 넘어서는 크기의 정수는 숫자로 표현 불가능하다!

매우 큰 정수의 표현을 위한 BigInteger 클래스

```
import java.math.*;

class SoBigInteger
{
    public static void main(String[] args)
    {
        System.out.println("최대 정수: " + Long.MAX_VALUE);
        System.out.println("최소 정수: " + Long.MIN_VALUE);

        BigInteger bigValue1=new BigInteger("10000000000000000000");
        BigInteger bigValue2=new BigInteger("-99999999999999999999");

        BigInteger addResult=bigValue1.add(bigValue2);
        BigInteger mulResult=bigValue1.multiply(bigValue2);

        System.out.println("큰 수의 덧셈결과: "+addResult);
        System.out.println("큰 수의 곱셈결과: "+mulResult);
    }
}
```

오차 없는 실수의 표현을 위한 BigDecimal 클래스

```
class WowDoubleError
{
	public static void main(String[] args)
	{
		BigDecimal e1=new BigDecimal(1.6);
		BigDecimal e2=new BigDecimal(0.1);

		System.out.println("두 실수의 덧셈결과: "+ e1.add(e2));
		System.out.println("두 실수의 곱셈결과: "+ e1.multiply(e2));
	}
}
```

```
class NoErrorBigDecimal
{
	public static void main(String[] args)
	{
		BigDecimal e1=new BigDecimal("1.6");
		BigDecimal e2=new BigDecimal("0.1");

		System.out.println("두 실수의 덧셈결과: "+ e1.add(e2));
		System.out.println("두 실수의 곱셈결과: "+ e1.multiply(e2));
	}
}
```

오차 없는 실수의 표현을 위한 BigDecimal 클래스

```
import java.math.*;

class NoErrorBigDecimal
{
	public static void main(String[] args)
	{
		BigDecimal e1=new BigDecimal("1.6");
		BigDecimal e2=new BigDecimal("0.1");

		System.out.println("두 실수의 덧셈결과: "+ e1.add(e2));
		System.out.println("두 실수의 곱셈결과: "+ e1.multiply(e2));
	}
}
```

난수의 생성, 문자열 토큰의 구분

난수(Random Number)의 생성

Random rand=new Random();

blooean nextBoolean()	boolean형 난수 반환
int nextInt()	int형 난수 반환
long nextLong()	long형 난수 반환
int nextInt(int n)	0이상 n미만의 범위 내에 있는 int형 난수 반환
float nextFloat()	0.0이상 1.0 미만의 float형 난수 반환
double nextDouble()	0.0이상 1.0 미만의 double형 난수 반환

```
import java.util.Random;
class RandomNumberGenerator
{
        public static void main(String[] args)
        {
                Random rand=new Random();
                for(int i=0; i<100; i++)
                        System.out.println(rand.nextInt(1000));
        }
}
```

씨드(Seed) 기반의 난수 생성

```
import java.util.Random;

class PseudoRandom
{
	public static void main(String[] args)
	{
		Random rand=new Random(12);
		for(int i=0; i<100; i++)
			System.out.println(rand.nextInt(1000));
	}
}
```

```
public static void main(String[] args)
{
	Random rand=new Random(12);
	rand.setSeed(System.currentTimeMillis());

	for(int i=0; i<100; i++)
		System.out.println(rand.nextInt(1000));
}
```

씨앗!이 동일하면 생성되는 난수의 패턴은 100% 동일!
이렇듯 컴퓨터의 난수는 씨앗을 기반으로 생성되기 때문에 가짜 난수(Pseudo-random number)라 불린다.

매 실행 시마다 다른 유형의 난수를 발생시키는 방법

Random 클래스의 디폴트 생성자 내에서는 왼편의 코드와 같이 씨드를 설정한다

문자열 토큰(Token)의 구분

“08:45”

“11:24”

콜론 :을 기준으로 문자열을 나눈다고 할 때,

→ 08, 45, 11, 24는 토큰(token)

→ 콜론 :는 구분자(delimeter)

```
import java.util.StringTokenizer;

class TokenizeString {
        public static void main(String[] args)     {
                String strData="11:22:33:44:55";
                StringTokenizer st=new StringTokenizer(strData, ":");
                while(st.hasMoreTokens())
                        System.out.println(st.nextToken());
        }
}
```

구분자 정보는 둘 이상 담을 수 있다. 하나의 문자열 안에 둘 이상을 담을 수 있다.

문자열 토큰(Token)의 구분

```
import java.util.StringTokenizer;

class TokenizeString2
{
	public static void main(String[] args)
	{
		String phoneNum="TEL 82-02-997-2059";		//국제 전화번호
		String javaCode="num+=1";

		System.out.println("First Result.....");
		StringTokenizer st1=new StringTokenizer(phoneNum);
		while(st1.hasMoreTokens())
			System.out.println(st1.nextToken());
		System.out.println("₩nSecond Result.....");
		StringTokenizer st2=new StringTokenizer(phoneNum, " -");
		while(st2.hasMoreTokens())
			System.out.println(st2.nextToken());

		System.out.println("₩nThird Result.....");
		StringTokenizer st3=new StringTokenizer(javaCode, "+=", true);
		while(st3.hasMoreTokens())
			System.out.println(st3.nextToken());
	}
}
```

정리합시다

- Wrapper 클래스
- Auto Boxing 과 Auto UnBoxing
- BigInteger클래스와 BigDecimal클래스
- Random 클래스
- StringTokenizer 클래스