Java.lang - Wrapper

우리는 음식을 보관할 때, 랩에 감싸서 보관한다. 자바에서도 감싸주는 클래스가 있으니, 바로 Wrapper Class가 있다. Wrapper Class는 제네릭과 컬렉션즈 프레임워크 이해에 상당한 도움이 되므로 눈여겨 볼 필요가 있다.

Wrapper Class

- **기본 자료형 데이터를 감싸는(Wrap) 클래스**

- `public static void showData(Object obj){`

  // print() 호출 시 obj의 toString() 메소드를 호출하여 인스턴스 obj가 담고있는 데이터정보 출력

  `System.out.println(obj);`

  `}`

  메소드가 인스턴스를 인자로 요구할 수 있는데, 기본자료형(int, double...) 형 데이터를 인자로 전달할 수가 없다. 그러므로, 기본자료형 데이터를 인스턴스화 할 경우에 클래스 형태로 Wrapping 해서 인스턴스로 전달해야 한다.

Wrapper Class 정의해보기

```
class IntWrapper {
     private int num;

     public IntWapper(int data){
          num = data;
     }

     @Override
     public String toString(){
          retrun ""+num;
     }
}
```

Java에서 기본적으로 제공하는 Wrapper Class

class	Constructor
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)
Double	Double(double value)

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

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

```
}
```

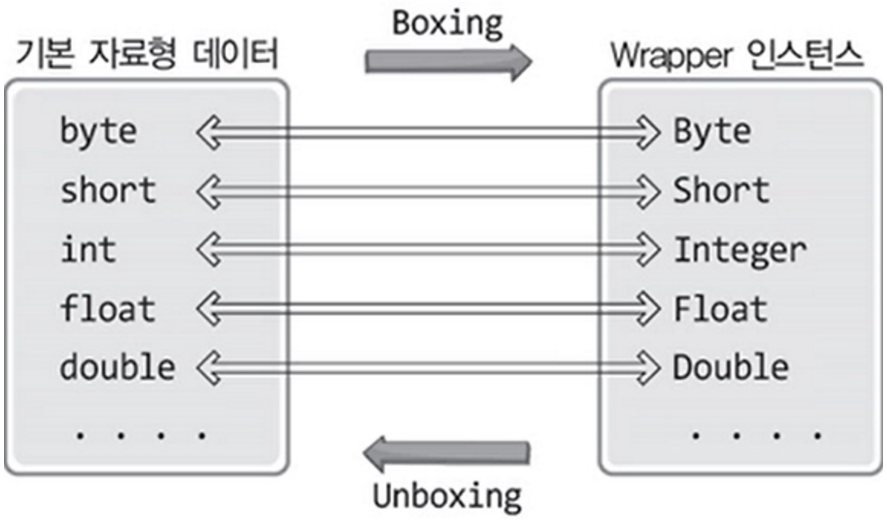
참고!
각 Wrapper Class들은 문자열을 인자로 받을 수 있는 생성자도 따로 마련이 되어있다. (character class 제외)

- `Integer intInst = new Integer("3");`
- `Float floatInst = new Float("3.1415");`

Wrapper Class의 두가지 기능

- Boxing
  기본 자료형 데이터를 Wrapper 인스턴스로 감싸는 것

- UnBoxing
  Wrapper 인스턴스에 저장된 기본 자료형 데이터를 꺼내는 것 (인스턴스 소멸 X)




```
class BoxingUnBoxing {
     public static void(String[] args){

          // Boxing
          Integer iValue = new Integer(10);
          Double dValue = new Double(3.14);

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

          // UnBoxing & Boxing
          iValue = new Integer(iValue.intValue()+10);
          dValue = new Double(dValue.doubleValue()+1.2);

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

     }
}
```

- **인스턴스 생성 : Boxing**
- **Wrapper Class의 toString(), ~Value() Method : 대표적인 Unboxing Method**

Auto-Boxing & Auto-UnBoxing

i. **기본 자료형 데이터가 Wrapper Class의 인스턴스로 자동 Boxing으로 감싼다.**
ii. **Wrapper Class의 인스턴스의 기본 자료형 데이터를 자동으로 UnBoxing해서 감싸준다.**

```
class AutoBox {
     public static void main(String[] args){

          // Auto-Boxing
          Integer iValue = 10;
          Double dValue = 3.14;

          // Integer ivalue = new Integer(10);
          // Double dValue = new Double(3.14);

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

          // Auto UnBoxing
          int num1 = iValue;
          double num2 = dValue;

          // int num1 = iValue.intValue();
          // double num2 = dValue.doubleValue();

          System.out.println(num1);
          System.out.println(num2);
     }
}
```

Wrapper Instance의 연산 - Auto-Boxing & Auto-UnBoxing

```
class AutoBoxOperator {
     public static void main(String[] args){

          Integer num1 = 10;
          Integer num2 = 20;

          num1++;
          // num1 = new Integer(num1.intValue()+1);
```

Q. 왜 num1에 저장된 값을 1 증가 시키면 되는데, 새로 인스턴스를 생성할까?

A. **String Class의 인스턴스처럼 내부에 저장된 데이터를 변경할 수 없다! 그러므로 새로운 인스턴스가 필요하다.**

```
          System.out.println(num1);

          num2 += 3;
          // num2 = new Integer(num2.intValue()+3);

          System.out.println(num2);

          int addResult = num1+num2;
          // int addResult = ( new Integer(num1.intValue()+num2.intValue()) ).intValue();

          System.out.println(addResult);

          int minResult = num1-num2;
          // int minResult = ( new Integer(num1.intValue()-num2.intValue()) ).intValue();

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

문자열을 기본 자료형으로 변환해주는 parse~() Method

- 인자로 받은 문자열을 해당 클래스의 타입에 맞게 기본 자료형으로 변환해주는 static Method

class	Use
Boolean	`boolean bool = Boolean.parseBoolean("true");`
Byte	`byte num = Byte.parseByte("10");`
Short	`short num = Short.parseShort("100");`
Integer	`int num = Integer.parseInt("1000");`
Long	`long num = Long.parseLong("10000");`
Float	`float num = Float.parseFloat("2.5F");`
Double	`double num = Double.parseDouble("3.5");`
