WRAPPER CLASS

Java is an Object-oriented programming language so represent everything in the form of the object, but java supports 8 primitive data types these all are not part of object.

To represent 8 primitive data types in the form of object form we required 8 java classes these classes are called wrapper classes.

The main objective of Wrapper class is wrap primitives into Object.

All wrapper classes present in the java.lang package and these all classes are immutable classes.

Wrapper classes hierarchy:-

- Object
    1. Number
        - A. Byte
        - B. Short
        - C. Integer
        - D. Long
        - E. Float
        - F. Double
    2. Character
    3. Boolean

There are three approach to convert primitive to Wrapper.

By Using Parameterized Constructors from Wrapper Classes.

Example:-1

```
int i = 10;
Integer in1 = new Integer(i);
Integer in2 = new Integer(10);
Integer in3 = new Integer("10");

Double D1 = new Double(10.5);
Double D2 = new Double("10.5");

Float F1 = new Float(10.5f);
Float F2 = new Float("10.5f");
```

```
Float F3 = new Float(10.5);
Float F4 = new Float("10.5");

Character ch1 = new Character('a');
```

DATATYPES	WRAPPER-CLASS	CONSTRUCTORS
byte	Byte	byte, String
short	Short	short, String
int	Integer	int, String
long	Long	long, String
float	Float	double, float, String
double	Double	double, String
char	Character	char
boolean	Boolean	boolean, String

B) By Using valueOf(-) Method Provided By Wrapper Classes:

```
public static XXX valueOf(xxx value)
XXX ----> Wrapper classes
xxx ----> primitive data types
```

EXAMPLE:-

```
int i = 10;
Integer in = Intreger.valueOf(i);
System.out.println(i+" "+in);
```

OUTPUT: 10 10

C) By Using Auto-Boxing Approach:

Auto-Boxing concept came in JDK5.0 version, in this approach we have not need to use pre-defined methods and constructor to convert primitive datatype to Object, simply, we have to assign primitive variable to wrapper class reference variable.

EXAMPLE:-

```
int i = 10;
Integer in = i;
System.out.println(i+" "+in);
```

OUTPUT: 10 10

Conversions From Object Type To Primitive Types:

a) By Using xxxValue() Method From Wrapper Classes:

```
public xxx xxxValue()
xxx ----> primitive data types
```

EXAMPLE:-

```
Integer in = new Integer(10);
int i = in.intValue();
System.out.println(in+" "+i);
```

OUTPUT: 10 10

b) By using Auto-Unboxing:

Auto-Unboxing was provided by JDK 5.0 version, in this approach no need to use any predefined methods, simply assign wrapper class reference variables to the respective primitive variables.

EXAMPLE:-

```
Integer in = new Integer(10);
int i = in;
System.out.println(in+" "+i);
```

OUTPUT: 10 10

Conversions from String Type to Wrapper type Object:

By Using String Parameterized Constructors From Wrapper Classes:

EXAMPLE:-

```
String data = "10";
Integer in = new Integer(data);
System.out.println(data+" "+in);
```

OUTPUT: 10 10

By Using Static valueOf(-) Method From Wrapper Classes:

```
public static XXX valueOf(String data)
```

EXAMPLE:-

```
String data = "10";
Integer in = Integer.valueOf(data);
System.out.println(data+" "+in);
```

OUTPUT: 10 10

Conversions From Object Type To String Type:

By using toString() Method from Wrapper Classes:

public String toString()

EXAMPLE:-

```
Integer in = new Integer(10);
String data = in.toString();
System.out.println(in+" "+data);
```

OUTPUT: 10 10

Conversions From String Type To Primitive Type:-

a) By using parseXXX() Method from Wrapper Classes:

public static xxx parseXxx(String data)

EXAMPLE:-

```
String data = "10";
int i = Integer.parseInt(data);
System.out.println(data+" "+i);
```

OUTPUT: 10 10

Conversion between Wrapper Object, Primitive and String