Wrapper classes

The main objectives of wrapper classes are:

1. To wrap primitives into object form so that we can handle primitives also just like objects.

2. To define several utility functions which are required for the primitives.

Constructors :

1. All most all wrapper classes define the following 2 constructors one can take corresponding primitive as argument and the other can take String as argument.

2. Example:

3. 1) Integer i=new Integer(10);

4. 2) Integer i=new Integer("10");

5. If the String is not properly formatted i.e., if it is not representing number then

we will get runtime exception saying "NumberFormatException".

6. Example:

7. class WrapperClassDemo {

8. public static void main(String[] args)throws Exception {

9. Integer i=new Integer("ten");

10. System.out.println(i);//NumberFormatException

11. }

12. }

13. Float class defines 3 constructors with float, String and double arguments.

14. 1) Float f=new Float (10.5f);

15. 2) Float f=new Float ("10.5f");

16. 3) Float f=new Float(10.5);

17. 4) Float f=new Float ("10.5");

18. Character class defines only one constructor which can take char primitive as argument there is no String argument constructor.

19. Character ch=new Character('a');//valid

20. Character ch=new Character("a");//invalid

21. Boolean class defines 2 constructors with boolean primitive and String

arguments.

If we want to pass boolean primitive the only allowed values are true, false where case should be lower case.

22. Example:

23. Boolean b=new Boolean(true);

24. Boolean b=new Boolean(false);

25. //Boolean b1=new Boolean(True);//C.E

26. //Boolean b=new Boolean(False);//C.E

27. //Boolean b=new Boolean(TRUE);//C.E

28. If we are passing String argument then case is not important and content is not important. If the content is case insensitive String of true then it is treated as true in all other cases it is treated as false.

29. Example 1:

30. class WrapperClassDemo {

31. public static void main(String[] args)throws Exception {

32. Boolean b1=new Boolean("true");

33. Boolean b2=new Boolean("True");

```
Boolean b3=new Boolean("false");
Boolean b4=new Boolean("False");
Boolean b5=new Boolean("ashok");
Boolean b6=new Boolean("TRUE");
System.out.println(b1);//true
System.out.println(b2);//true
System.out.println(b3);//false
System.out.println(b4);//false
System.out.println(b5);//false
System.out.println(b6);//true
}
}
```

Example 2(for exam purpose):

```
class WrapperClassDemo {
public static void main(String[] args)throws Exception {
Boolean b1=new Boolean("yes");
Boolean b2=new Boolean("no");
System.out.println(b1);//false
System.out.println(b2);//false
System.out.println(b1.equals(b2));//true
System.out.println(b1==b2);//false
}
}
```

Wrapper class Constructor summery :

Wrapper class	Constructor summery
Byte	byte, String
Short	short, String
Integer	Int, String
Long	long, String
Float	float, String, double
Double	double, String
Character	char,~~String~~
Boolean	boolean, String

Note :

1) In all wrapper classes toString() method is overridden to return its content.

2) In all wrapper classes .equals() method is overridden for content compression.

Example :

```
Integer i1 = new Integer(10) ;

Integer i2 = new Integer(10);

System.out.println(i1); //10

System.out.println(i1.equals(i2)); //true
```

Utility methods :

1. valueOf() method.

2. XXXValue() method.

3. parseXxx() method.

4. toString() method.

valueOf() method :

We can use valueOf() method to create wrapper object for the given primitive or String this method is alternative to constructor.

Form 1:

Every wrapper class except Character class contains a static valueOf() method to create wrapper object for the given String.

public static Integer valueOf(String s, int $\underbrace{\text{radix)}}_{\to \text{base}}$

public static wrapper valueOf(String s);

Example:

```
class WrapperClassDemo {
public static void main(String[] args)throws Exception {
Integer i=Integer.valueOf("10");
Double d=Double.valueOf("10.5");
Boolean b=Boolean.valueOf("ashok");
System.out.println(i);//10
System.out.println(d);//10.5
System.out.println(b);//false
}
}
```

Form 2:

Every integral type wrapper class (Byte, Short, Integer, and Long) contains the

following valueOf() method to convert specified radix string to wrapper object.

public static wrapper valueOf(String s , int radix) ;

//radix means base

Note:

the allowed radix range is 2 to 36.

base 2	0 To 1
base 8	0 To 7
base 10	0 To 9
base 11	0 To 9,a
base 16	0 To 9,a To f
base 36	0 To 9,a to z

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
Integer i=Integer.valueOf("100",2);
System.out.println(i);//4
}
}
```

Analysis:

$$\begin{array}{ccc} 1 & 0 & 0 \\ | & | & | \\ 2^2 & 2^1 & 2^0 \end{array} \qquad \begin{array}{l} 2^0*0+2^1*0+2^2*1 \\ 1*0+2*0+4*1 \\ 0+0+4=4 \end{array}$$

Form 3 :

Every wrapper class including Character class defines valueOf() method to convert

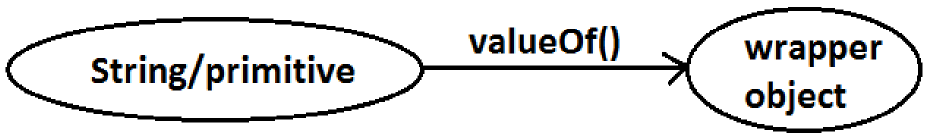
primitive to wrapper object.

public static wrapper valueOf(primitive p);

Example:

```
class WrapperClassDemo {
public static void main(String[] args)throws Exception {
Integer i=Integer.valueOf(10);
Double d=Double.valueOf(10.5);
Boolean b=Boolean.valueOf(true);
Character ch=Character.valueOf('a');
System.out.println(ch); //a
System.out.println(i);//10
System.out.println(d);//10.5
System.out.println(b);//true
}
}
```

Diagram:



xxxValue() method :

We can use xxxValue() methods to convert wrapper object to primitive.

Every number type wrapper class (Byte, Short, Integer, Long, Float, Double) contains

the following 6 xxxValue() methods to convert wrapper object to primitives.

1)public byte byteValue()

2)public short shortValue()

3)public int intValue()

4)public long longValue()

5)public float floatValue()

6)public double doubleValue();

Example:

```
class WrapperClassDemo {
                                        public static void main(String[]
args)throws Exception {
Integer i=new Integer(130);
System.out.println(i.byteValue());//-126
System.out.println(i.shortValue());//130
System.out.println(i.intValue());//130
System.out.println(i.longValue());//130
System.out.println(i.floatValue());//130.0
System.out.println(i.doubleValue());//130.0
}
}
```

charValue() method:

Character class contains charValue() method to convert Character object to char

primitive.

public char charValue();

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
Character ch=new Character(' char c=ch.charValue();
System.out.println(c);//a
}
}
```

booleanValue() method:

Boolean class contains booleanValue() method to convert Boolean object to boolean

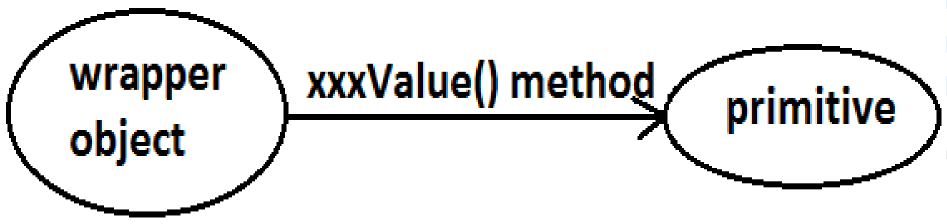
primitive.

public boolean booleanValue( );

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
Boolean b=new Boolean("ashok");
boolean b1=b.booleanValue();
System.out.println(b1);//false
}
}
```

Diagram :



In total there are 38(= 6*6+1+1) xxxValue() methods are possible.

parseXxx() method :

We can use this method to convert String to corresponding primitive.

Form1 :

Every wrapper class except Character class contains a static parseXxx() method to convert String to corresponding primitive.

public static primitive parseXxx(String s);

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
int i=Integer.parseInt("10");
boolean b=Boolean.parseBoolean("ashok");
double d=Double.parseDouble("10.5");
System.out.println(i);//10
System.out.println(b);//false
System.out.println(d);//10.5
}
}
```

Form 2:

integral type wrapper classes(Byte, Short, Integer, Long) contains the following

parseXxx() method to convert specified radix String form to corresponding primitive.

public static primitive parseXxx(String s,int radix);

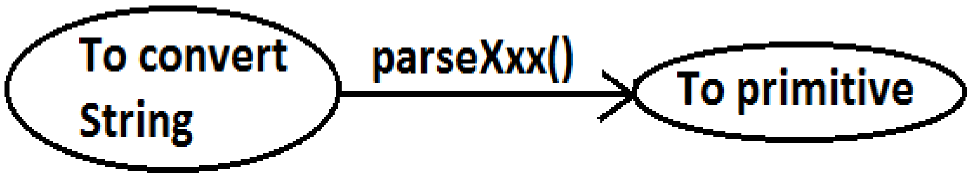
The allowed range of redix is : 2 to 36

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
int i=Integer.parseInt("100",2);
System.out.println(i);//4
}
```

}

Diagram :



toString() method :

We can use toString() method to convert wrapper object (or) primitive to String.

Form 1 :

public String toString();

1. Every wrapper class (including Character class) contains the above toString()

method to convert wrapper object to String.

2. It is the overriding version of Object class toString() method.

3. Whenever we are trying to print wrapper object reference internally this

toString() method only executed.

Example:

```
class WrapperClassDemo {

public static void main(String[] args) {

Integer i=Integer.valueOf("10");

System.out.println(i);//10

System.out.println(i.toString());//10

}

}
```

Form 2:

Every wrapper class contains a static toString() method to convert primitive to String.

```
public static String toString(primitive p);
```

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
String s1=Integer.toString(10);
String s2=Boolean.toString(true);
String s3=Character.toString('a');
System.out.println(s1); //10
System.out.println(s2); //true
System.out.println(s3); //a
}
}
```

Form 3:

Integer and Long classes contains the following static toString() method to convert the

primitive to specified radix String form.

```
public static String toString(primitive p, int radix);
```

Example:

```
class WrapperClassDemo {
public static void main(String[] args) {
String s1=Integer.toString(7,2);
String s2=Integer.toString(17,2);
System.out.println(s1);//111
System.out.println(s2);//10001
}
}
```

Form 4:

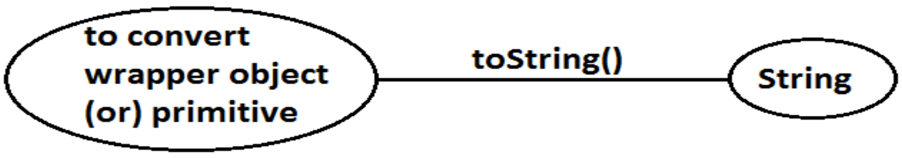
Integer and Long classes contains the following toXxxString() methods.

```
public static String toBinaryString(primitive p);
public static String toOctalString(primitive p);
public static String toHexString(primitive p);
```

Example:

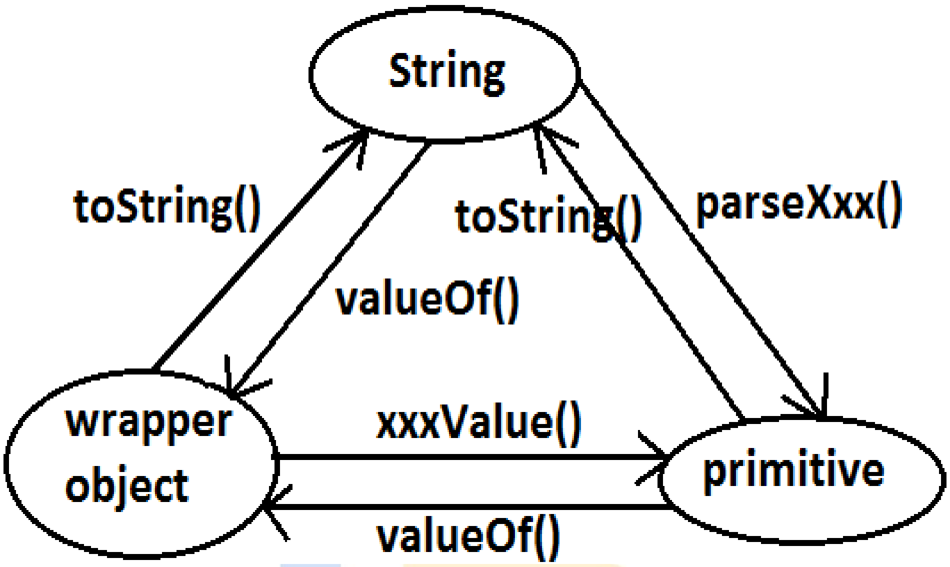
```
class WrapperClassDemo {
public static void main(String[] args) {
String s1=Integer.toBinaryString(7);
String s2=Integer.toOctalString(10);
String s3=Integer.toHexString(20);
String s4=Integer.toHexString(10);
System.out.println(s1);//111
System.out.println(s2);//12
System.out.println(s3);//14
System.out.println(s4);//a
}
}
```

Diagram:



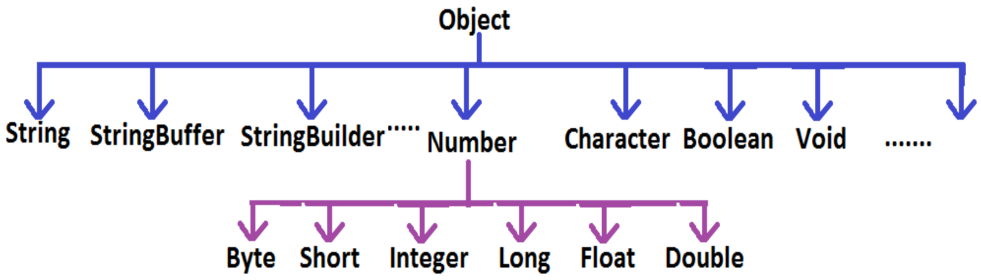
Dancing between String, wrapper object and primitive :

Diagram:



Partial Hierarchy of java.lang package :

Diagram :



1. String, StringBuffer, StringBuilder and all wrapper classes are final classes.

2. The wrapper classes which are not child class of Number are Boolean and

Character.

3. The wrapper classes which are not direct class of Object are Byte, Short, Integer,

Long, Float, Double.

4. Sometimes we can consider Void is also as wrapper class.

5. In addition to String objects , all wrapper class objects also immutable in java.

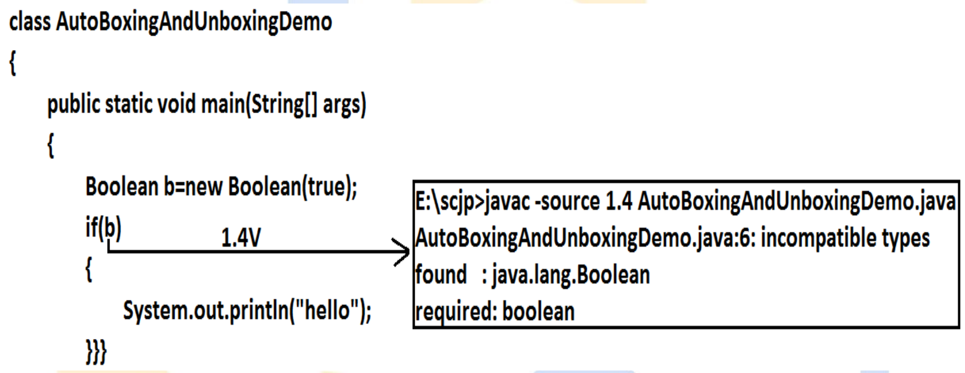
Until 1.4 version we can't provide wrapper object in the place of primitive and primitive

in the place of wrapper object all the required conversions should be performed

explicitly by the programmer.

Example 1 :

Program 1 :



```
class AutoBoxingAndUnboxingDemo
{
    public static void main(String[] args)
    {
        Boolean b=new Boolean(true);
        if(b)
        {
            System.out.println("hello");
        }}}
```

```
E:\scjp>javac -source 1.4 AutoBoxingAndUnboxingDemo.java
AutoBoxingAndUnboxingDemo.java:6: incompatible types
found   : java.lang.Boolean
required: boolean
```

Program 2:

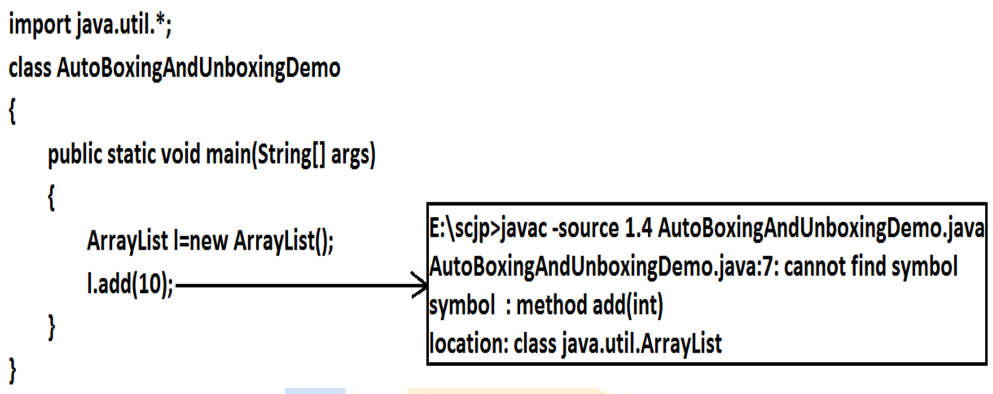
```
class AutoBoxingAndUnboxingDemo {
public static void main(String[] args) {
Boolean b=new Boolean(true);
if(b) {
System.out.println("hello");
}
}
}
```

Output:

Hello

Example 2:

Program 1:



```
import java.util.*;
class AutoBoxingAndUnboxingDemo
{
    public static void main(String[] args)
    {
        ArrayList l=new ArrayList();
        l.add(10);
    }
}
```

```
E:\scjp>javac -source 1.4 AutoBoxingAndUnboxingDemo.java
AutoBoxingAndUnboxingDemo.java:7: cannot find symbol
symbol  : method add(int)
location: class java.util.ArrayList
```

Program 2:

```
import java.util.*;
class AutoBoxingAndUnboxingDemo {
public static void main(String[] args) {
ArrayList l=new ArrayList();
Integer i=new Integer(10);
l.add(i);
}
}
```

But from 1.5 version onwards we can provide primitive value in the place of wrapper

and wrapper object in the place of primitive all required conversions will be performed

automatically by compiler. These automatic conversions are called Autoboxing and

Autounboxing.

Autoboxing :

Automatic conversion of primitive to wrapper object by compiler is called Autoboxing.

Example :

Integer i=10;

[compiler converts "int" to "Integer" automatically by Autoboxing]

After compilation the above line will become.

Integer i=Integer.valueOf(10);

That is internally Autoboxing concept is implemented by using valueOf() method.

Autounboxing :

automatic conversion of wrapper object to primitive by compiler is called

Autounboxing.

Example:

Integer I=new Integer(10);

Int i=I;

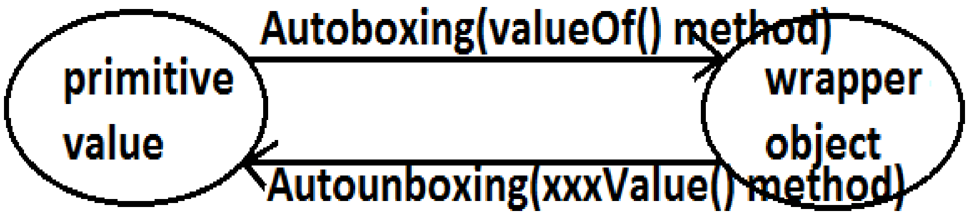
[compiler converts "Integer" to "int" automatically by Autounboxing]

After compilation the above line will become.

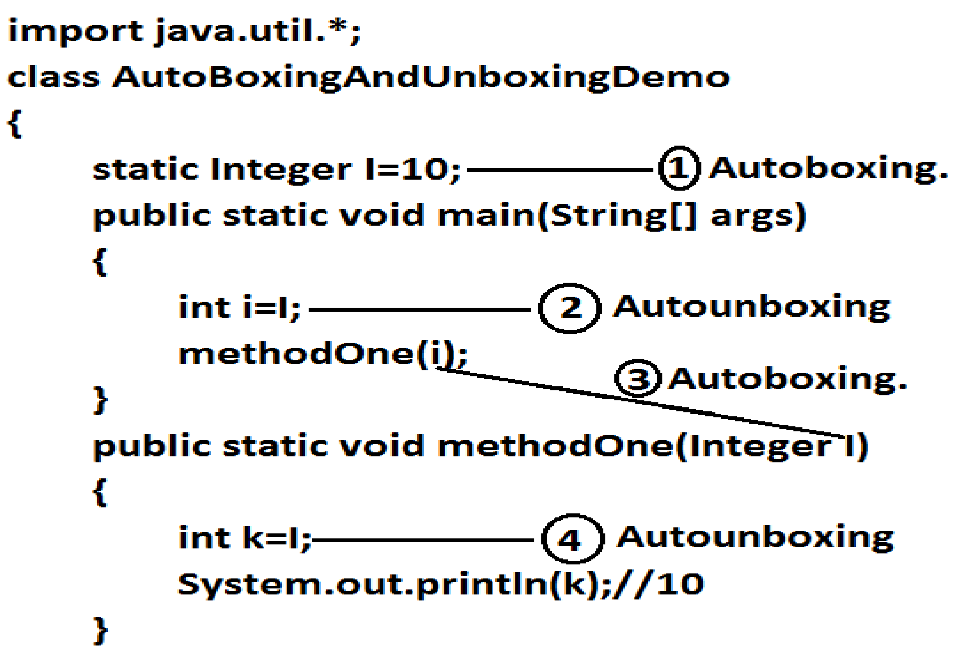
Int i=I.intValue();

That is Autounboxing concept is internally implemented by using xxxValue() method.

Diagram :



Example :



```
import java.util.*;
class AutoBoxingAndUnboxingDemo
{
	static Integer I=10;——————① Autoboxing.
	public static void main(String[] args)
	{
		int i=I;——————② Autounboxing
		methodOne(i);
		                    ③ Autoboxing.
	}
	public static void methodOne(Integer I)
	{
		int k=I;——————④ Autounboxing
		System.out.println(k);//10
	}
```

It is valid in 1.5 version but invalid in 1.4 version.

Note:

From 1.5 version onwards we can use primitives and wrapper objects interchangly the

required conversions will be performed automatically by compiler.

Example 1:

import java.util.*;

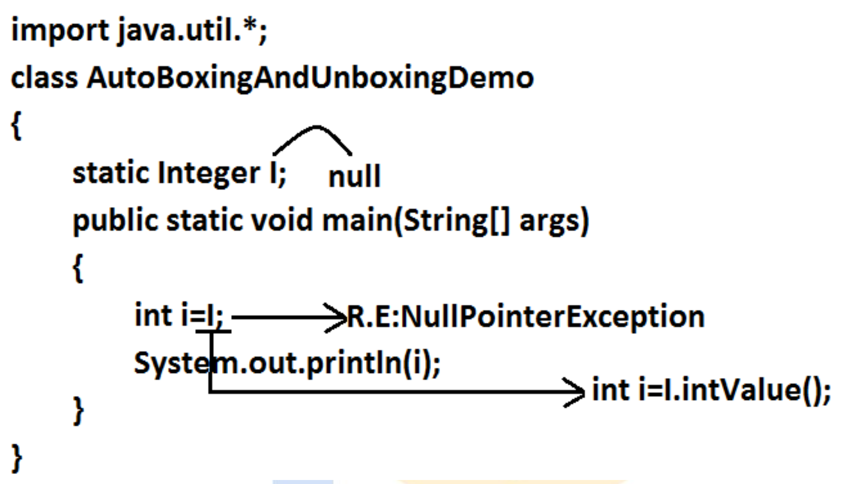
class AutoBoxingAndUnboxingDemo {

static Integer I=0;

public static void main(String[] args) {

```
int i=I;

System.out.println(i);//0

}
```

Example 2 :



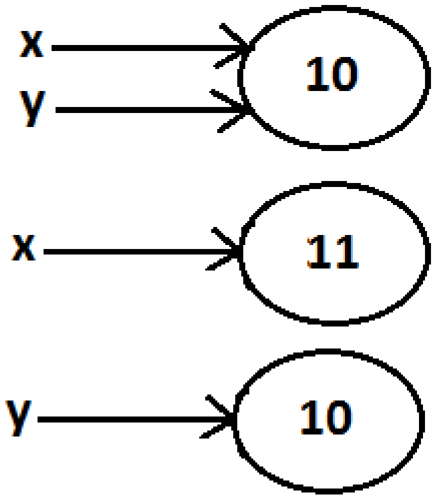
```
import java.util.*;
class AutoBoxingAndUnboxingDemo
{
	static Integer I;
	public static void main(String[] args)
	{
		int i=I;
		System.out.println(i);
	}
}
```

If we provide null reference for autounboxing , we will get NullPointerException

Example 3:

```
import java.util.*;

class AutoBoxingAndUnboxingDemo {

public static void main(String[] args) {

Integer x=10;

Integer y=x;

++x;

System.out.println(x);//11

System.out.println(y);//10

System.out.println(x==y);//false

}

}
```

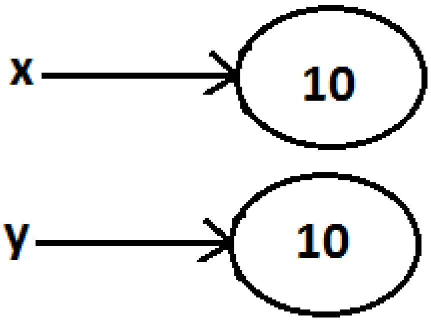
Diagram :



Example 2 :

```
import java.util.*;
class AutoBoxingAndUnboxingDemo {
public static void main(String[] args) {
Integer x=new Integer(10);
Integer y=10;
System.out.println(x==y);//false
}
}
```

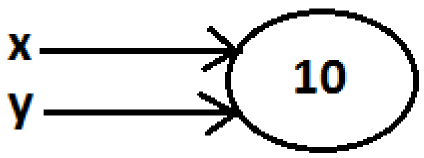
Diagram:



Example 3:

```
import java.util.*;
class AutoBoxingAndUnboxingDemo {
public static void main(String[] args) {
Integer x=new Integer(10);
Integer y=x;
System.out.println(x==y);//true
}
}
```

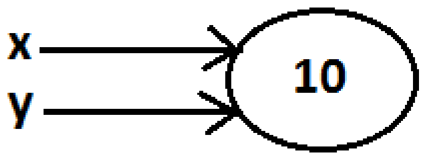
Diagram :



Example 4 :

```
import java.util.*;
class AutoBoxingAndUnboxingDemo {
public static void main(String[] args) {
Integer x=10;
Integer y=10;
System.out.println(x==y);//true
}
}
```

Diagram:



Example 5 :

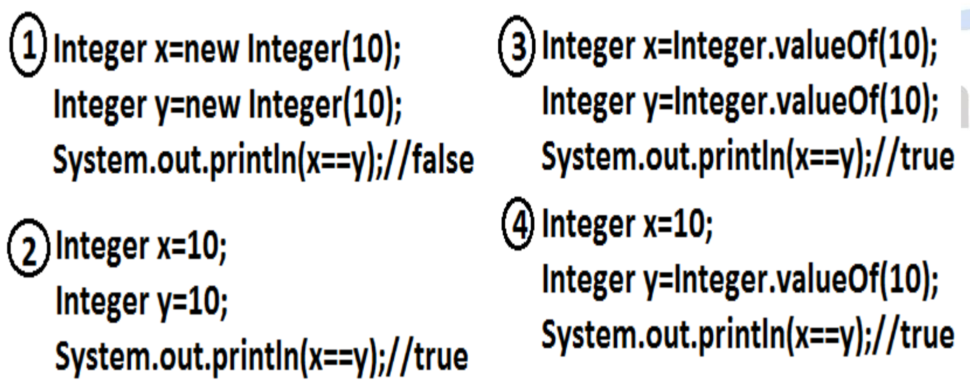
```
import java.util.*;

class AutoBoxingAndUnboxingDemo {

public static void main(String[] args) {

Integer x=100;

Integer y=100;

System.out.println(x==y);//true

}

}
```

Diagram :

Internally Autoboxing concept is implemented by using valueOf() method hence the

above rule applicable even for valueOf() method also.

Examples :



```
① Integer x=new Integer(10);
  Integer y=new Integer(10);
  System.out.println(x==y);//false

② Integer x=10;
  Integer y=10;
  System.out.println(x==y);//true

③ Integer x=Integer.valueOf(10);
  Integer y=Integer.valueOf(10);
  System.out.println(x==y);//true

④ Integer x=10;
  Integer y=Integer.valueOf(10);
  System.out.println(x==y);//true
```

Note:

When compared with constructors it is recommended to use valueOf() method to create

wrapper object.