**java.lang package – Part-11**

* **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 using valueOf() methods.

* **Auto-Unboxing:**

Automatic conversion of Wrapper object to primitive by compiler is called auto-unboxing.

Integer I = new Integer(10);

int I = I;

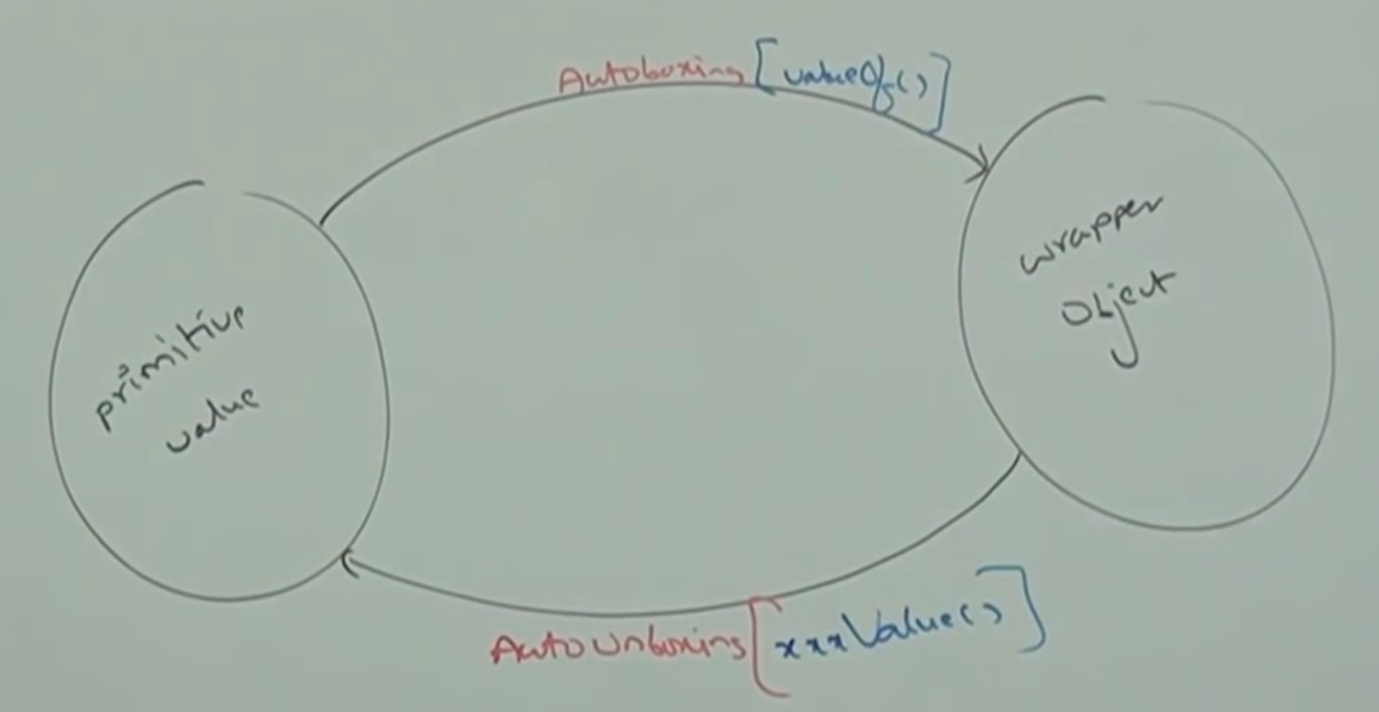
[Compiler converts Integer to int automatically by auto-unboxing]

After compilation the above line will become

int i = I.intValue();

That is, internally Auto-unboxing concept is implemented by using xxxValue() methods.

* **Diagrammatic Representation:**

****

* **Example\_01:**

class Test{

static Integer I = 10; // A.B

public static void main(String[] args){

int i = I; // A.U.B

m1(i); // A.B

}

public static void m1(Integer K){

int m = K;

System.out.println(m); // A. U. B

}

}

Output: 10

Note:

Above code is valid in 1.5 version, but invalid in 1.4 version.

Just because of autoboxing and auto-unboxing, we can use primitives and wrapper objects interchangeably from 1.5 version onwards.

* **How to compile a code with specific version through command prompt?**

javac -source Test.java

* **Example\_02:**

class Test{

static Integer I = 0;

public static void main(String[] args){

int m = I;

System.out.println(m);

}

}

Output: 0

class Test{

static Integer I;

public static void main(String[] args){

int m = I; // I.intValue 🡪 null.intValue();

System.out.println(m);

}

}

Output: RE: NullPointerException

Note:

On null reference if we are trying to perform auto-unboxing then we will get Runtime exception saying NullPointerException.

* **Exampe\_03:**

Integer X = 10;

Integer Y = X;

X++

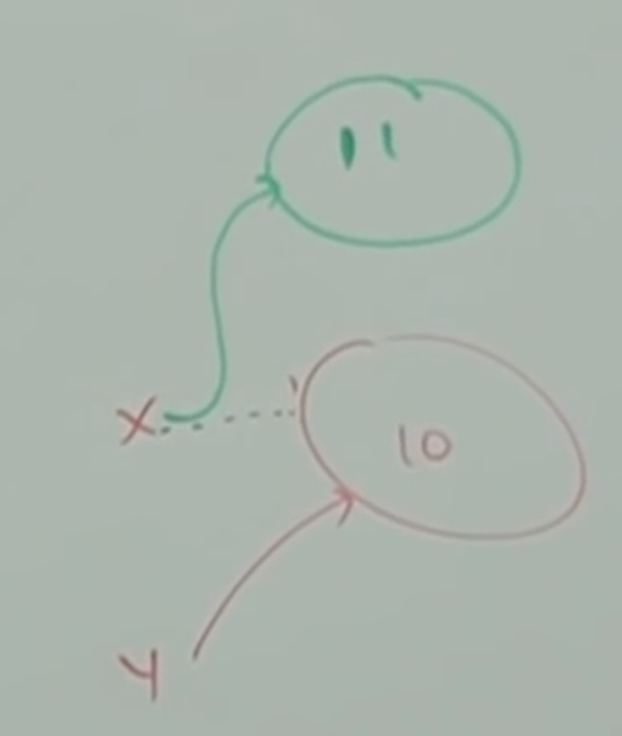
System.out.println(X); // 11

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

System.out.println(X == Y); //false

Note:

All Wrapper class objects are immutable, that is once we create Wrapper class object we can’t perform any changes in that object. If we are trying to perform any changes, with those changes a new object will be created.



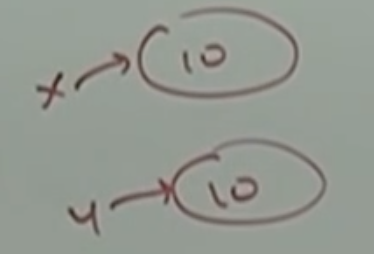
* **Example\_04:**

**Case\_01:**

Integer X = new Integer(10);

Integer Y = new Integer(10);

System.out.println(X == Y); // false

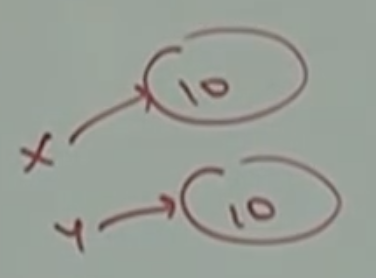


**Case\_02:**

Integer x = new Integer(10);

Integer y = 10;

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

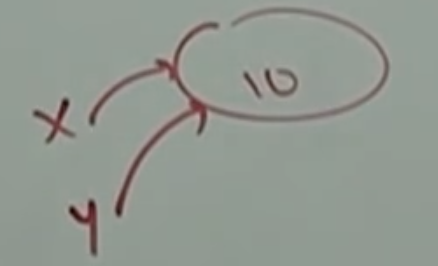


**Case\_03:**

Integer x = 10;

Integer y = 10;

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

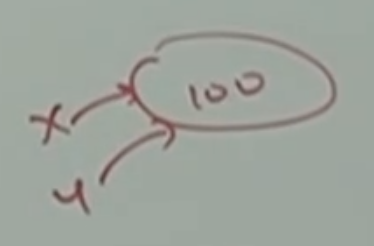


**Case\_04:**

Integer x = 100;

Integer y = 100;

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

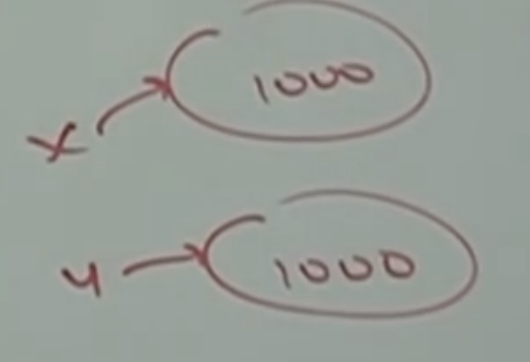


**Case\_05:**

Integer x = 1000;

Integer y = 1000;

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



Note:

Please read the below conclusion for getting false for the above case.

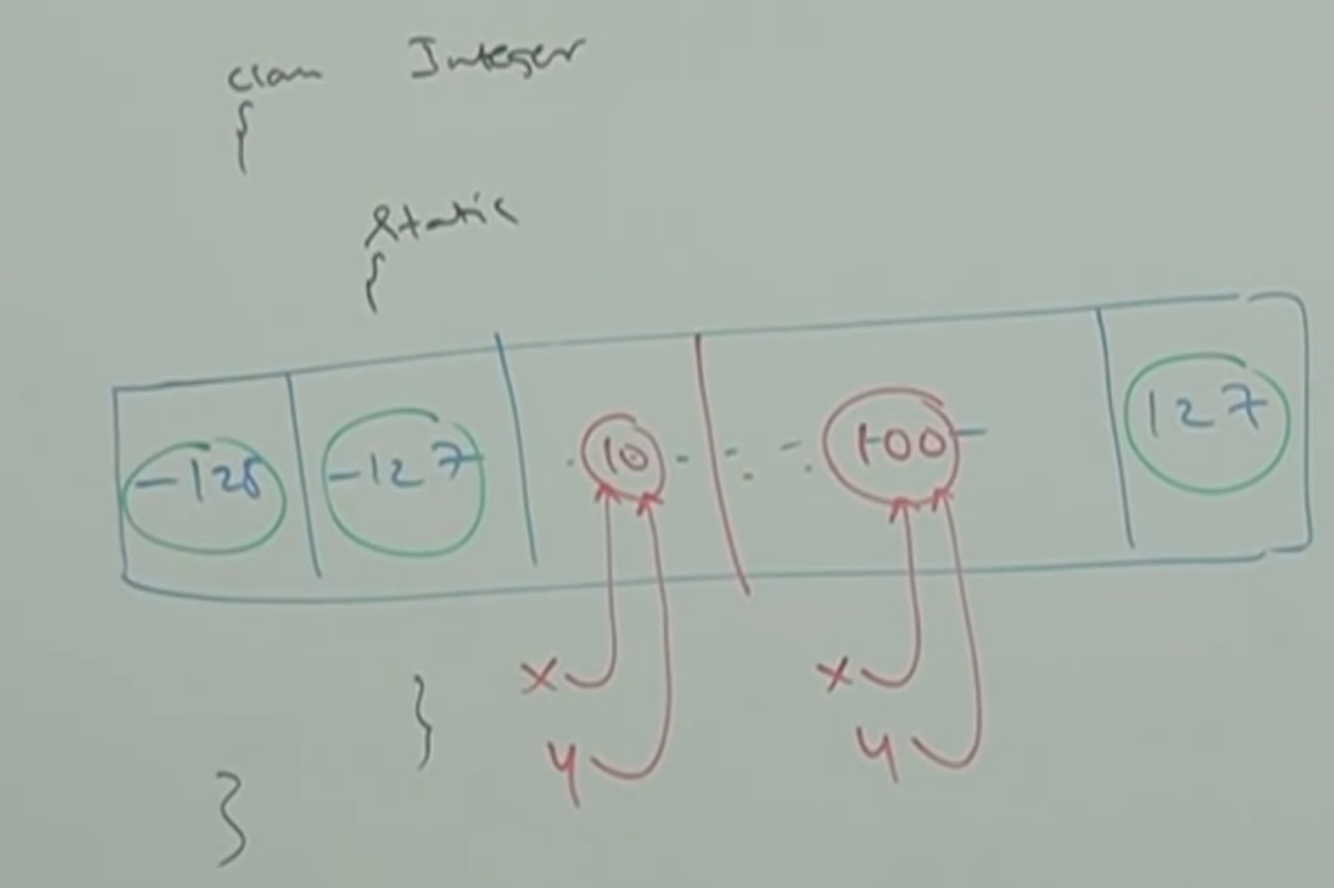
Conclusion:

Internally to provide support for Autoboxing, a buffer of Wrapper objects will be created at the time of Wrapper class loading.

By Autoboxing if an object is required to create, first JVM will check whether this object is present already in the buffer or not, if

It is already present in the buffer, then existing buffer object will

Be used. If it is not available in the buffer then JVM will create a new object.



But buffer concept is available only in the following ranges

Byte 🡪 Always

Short 🡪 -128 to 127

Integer 🡪 -128 to 127

Long 🡪 -128 to 127

Character 🡪 0 to 127

Boolean 🡪 Always

Except this range, in all remaining cases a new object will be created.

* **Example:**

Integer x = 127;

Integer y = 127;

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

Integer x = 128;

Integer y = 128;

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

Boolean x = false;

Boolean y = false;

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

Double x = 10.0;

Double y = 10.0;

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

Internally autoboxing concept is implemented by using valueOf() method. Hence, buffering concept is applicable for valueOf() method also.

Integer x = new Integer(10);

Integer y = new Integer(10);

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

Integer x = 10;

Integer y = 10;

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

Integer x = Integer.valueOf(10);

Integer y = Integer.valueOf(10);

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

Integer x = 10;

Integer y = Integer.valueOf(10);

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