Wrapper classes in java :
----
* In java we have 8 primitive data types i.e byte, short, int, long, float, double, char and boolean.

* Java is not a pure object Oriented language because It accepts primitive data types.

* Except these 8 primitive data types, Everything is object in java.

* The basic drawback with primitive data types are as follows :

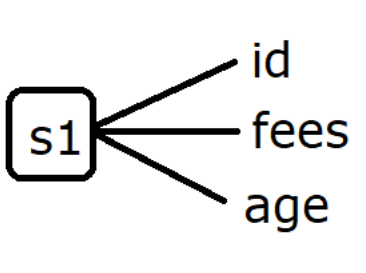
a) It cannot move in the network, only objects are moving in the network.

Example :

```
public class Student
{
    private int id;
    private double fees;
    private int age;
}

Student s1 = new Student();
```

->No built-in utility methods
->Cannot be used in Java Collections
->Cannot be treated polymorphically
->Always store values — cannot be null



Object can move in the network along with data.

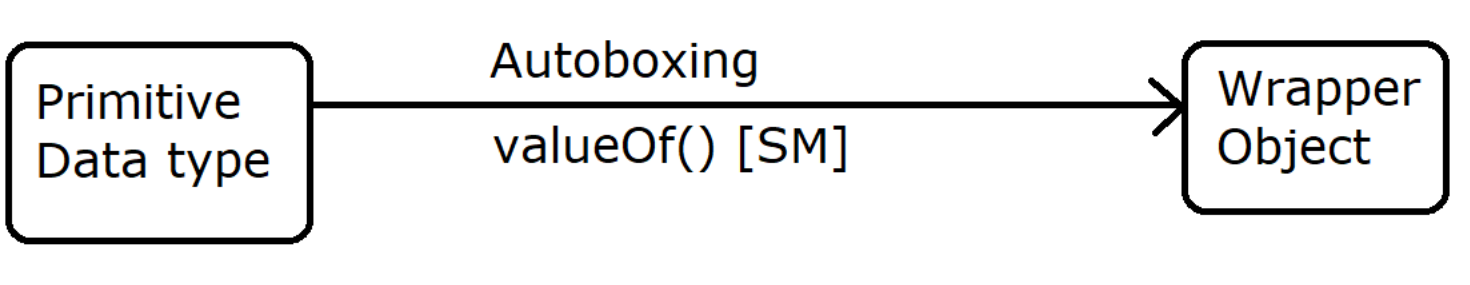
b) On primitive, We cannot assign null as well as we cannot call any method.

* To avoid the above said problem, Java software people has introduced Wrapper class concept.

* In JDK 1.5V, We have the following two concepts :

1) Autoboxing [Converting the primitive into corresponding Wrapper object]
2) Unboxing [Converting the Wrapper object back to primitive]

Autoboxing :
----
* It is an automtic technique to convert the primitive data type into corresponding Wrapper object.
* Autoboxing is available from JDK 1.5V
* Before Autoboxing, User was manually converting the primitive data into Wrapper object



Before Java 1.5 :

```
int i = 10;
Integer obj = Integer.valueOf(i);
System.out.println(obj); //10
```

From Java 5v

```
int i = 20;
Integer obj = i;
```

Java Compiler, Internally convert this code in the following

```
int i = 20;
Integer obj = Integer.valueOf(i);
```

Overloaded valueOf() method :
----
* We have following overloaded methods are available :
----
1) public static Integer valueOf(int x) :
----
* It will accept primitive and convert into Wrapper object.

2) public static Integer valueOf(String str);
----
* It will convert the String into Wrapper object. parseInt(String str) will convert the String into primitive type.

3) public static Integer valueOf(String str, int radix/base) :
It will convert the given String number into Integer object by using the specified radix or base.

Note :- We can pass base OR radix upto 36
i.e A to Z (26) + 0 to 9 (10) -> [26 + 10 = 36], It can be calculated by using Character.MAX_RADIX.

* We have 8 Wrapper classes , ALL THESE WRAPPER CLASSES ARE **IMMUTABLE AND hashCode() and equals(Object obj) both the methods are overridden in these Wrapper classe.**

Primitive		Wrapper Object
byte	:	Byte
short	:	Short
int	:	Integer
long	:	Long
float	:	Float
double	:	Double
char	:	Character
boolean	:	Boolean

WAP that shows wrapper classes are immutable :

```
public class Immutable
{
    public static void main(String[] args)
    {
      Integer i = new Integer(100);  //deprecated
      accept(i);
      System.out.println(i);
    }

    public static void accept(Integer j)
    {
        j = 999;
    }
}
```
Output is 100, that means Integer class is immutable class

```
//Integer.valueOf(int);
public class AutoBoxing1
{
    public static void main(String[] args)
    {
        int a = 12;
        Integer x = Integer.valueOf(a); //Upto 1.4 version
        System.out.println(x);


      int y = 15;
        Integer i = y;   //From 1.5 onwards compiler takes care
        System.out.println(i);
    }
}
```

```
public class AutoBoxing2
{
    public static void main(String args[])
    {
         byte b = 12;
        Byte b1 = Byte.valueOf(b);
        System.out.println("Byte Object :"+b1);

       short s = 17;
       Short s1 = Short.valueOf(s);
       System.out.println("Short Object :"+s1);

       int i = 90;
       Integer i1 = Integer.valueOf(i);
       System.out.println("Integer Object :"+i1);

         long g = 12;
         Long h =  Long.valueOf(g);
         System.out.println("Long Object :"+h);

         float f1 = 2.4f;
         Float f2 = Float.valueOf(f1);
         System.out.println("Float Object :"+f2);

         double k = 90.90;
         Double l = Double.valueOf(k);
         System.out.println("Double Object :"+l);

         char ch = 'A';
         Character ch1 = Character.valueOf(ch);
         System.out.println("Character Object :"+ch1);

         boolean x = true;
         Boolean x1 = Boolean.valueOf(x);
         System.out.println("Boolean Object :"+x1);

              }
}
```

```
//Integer.valueOf(String str)
//Integer.valueOf(String str, int radix/base)
public class AutoBoxing3
{
    public static void main(String[] args)
    {
            Integer a = Integer.valueOf(15);

            Integer b = Integer.valueOf("25");

        Integer c = Integer.valueOf("111",36); //Here Base we can take upto 36

            System.out.println(a);
            System.out.println(b);
            System.out.println(c);
    }
}
```

Whenever we create the Integer object then the range of **cache memory is upto 127 so if we use == operator to compare two Integer object and if the range is upto 127 then it will generate true, If the range is out of 127 (out of cache memory value) then it will generate false.**

It is strongly recommended to comapre Wrapper objects by using equals(Object obj) methods but not by using == operator.

equals(Object obj) is overridden in all the Wrapper classes for content comparison.

```
public class AutoBoxing4
{
    public static void main(String[] args)
    {
        Integer i1 = 127;
        Integer i2 = 127;
        System.out.println(i1==i2); //true
        System.out.println(i1.equals(i2)); //true

        System.out.println(".............");

        Integer i3 = 128;
        Integer i4 = 128;
        System.out.println(i3==i4); //false
        System.out.println(i1.equals(i2)); //true
    }
}
```

How to convert Integer to String :

Integer class has provided a static method toString(int x), by using this method we can convert the integer into String type.

```
//Converting integer value to String
public class AutoBoxing5
{
    public static void main(String[] args)
    {
        int x = 12;
        String str = Integer.toString(x);
        System.out.println(str+2);
    }
}
```

---

1) Object class and its methods
2) Nested classes in java
3) Enum in java
4) Java input ouput
----
* Polymorphism
* Abstraction (Abstract class and interface)
* Exception Handling
* Multithrading
* Collection Framework