La principal diferencia entre un tipo primitivo y un wrapper es que éste último es una clase. Cuando trabajamos con wrappers estamos trabajando con objetos mientras que cuando trabajamos con un tipo primitivo obviamente no.

A simple vista un wrapper siendo un objeto, puede aportar muchas ventajas pero, ¿qué problemas nos puede plantear utilizar objetos en vez de tipos primitivos? El problema que nos podemos encontrar es que cuando le pasamos una variable a un método como argumento y esta es de un tipo primitivo se le pasa siempre por valor mientras que cuando pasamos un wrapper(objeto) se lo estamos pasando por referencia.

Una de las grandes ventajas que tienen estos wrappers es la facilidad de conversión entre tipos primitivos y cadenas de caracteres en ambos sentidos. Existen wrappers de todos los tipos primitivos numéricos. La siguiente tabla muestra los tipos primitivos y sus wrappers asociados:

Tipo primitivo	Wrapper asociado
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Estas clases tienen como padre común Number excepto Character y Boolean.

CLASE WRAPPER INTEGER

El wrapper Integer tiene dos constructores:

1. Integer(int)
2. Integer(String)

En la siguiente tabla se muestra un resumen de algunos de los métodos del wrapper Integer:

Método	Descripción
Integer(int) Integer(String)	Constructores
byteValue() shortValue() intValue() longValue() doubleValue() floatValue()	Funciones de conversión con datos primitivos
Integer decode(String) Integer parseInt(String) Integer parseInt(String,int) Integer valueOf(String)	Conversión de String

String toString()	
String toBinaryString(int) String toHexString(int) String toOctalString(int)	Conversión a otros sistemas de numeración
MAX_VALUE,MIN_VALUE,TYPE	Constantes

El método parseInt(String,int) permite introducir como segundo parámetro la base en la que está codificado el número del primer parámetro.

La creación de un objeto Integer es la misma que para cualquier otro tipo de objeto:

Integer i1 = new Integer(“7”);
Integer i2 = new Integer(5);

Una vez asignado un valor al objeto Integer no puede cambiarse. Si queremos utilizar otro valor deberemos crear otro objeto Integer.

En el siguiente programa se muestra la utilización de algunos de los métodos expuestos en la tabla anterior:

```
public class Wrappers {
   public static void main(String[] args) {
      Integer i1,i2,i3,i4,i5;
      String s1;
      int i6;
      i1 = new Integer(5);
      i2 = new Integer("7");
      s1 = i1.toString();
      System.out.println(s1);//Muestra 5 por pantalla
      i3=Integer.parseInt("10",10);
      i4=Integer.parseInt("10",8);
      i5=Integer.parseInt("BABA",16);
      System.out.println(i3);//Muestra 10 por pantalla
      System.out.println(i4);//Muestra 8 por pantalla
      System.out.println(i5);/*Muestra 47.802 por pantalla, que equivale en
                              decimal al número BABA en hexadecimal*/
      System.out.println(Integer.toOctalString(i4));//Muestra 10 por pantalla
      System.out.println(Integer.toHexString(i5));//Muestra baba por pantalla
      i6=Integer.valueOf("22").intValue();
      System.out.println(i6);//Muestra 22 por pantalla
      }
}
```

Los wrappers para los demás tipos primitivos tienen una funcionalidad y modo de utilización similar al wrapper Integer.