Wrappers

Hay estructuras de datos, como las colecciones, que no trabajan con tipos primitivos, sino con objetos. Entre otras cosas, esto permite que las variables puedan tener un valor *null*. Por tanto, no se pueden insertar valores de tipo primitivo, como int o double. Para sustituirlos, Java ha implementado unas clases que "envuelven" datos primitivos dentro de un objeto llamado envoltorio o *wrapper*. Estos, al ser objetos, se pueden insertar en una lista o un conjunto, o bien pasarlo a una función donde se espera un objeto como argumento.

Para cada tipo primitivo hay definida una clase envoltorio (véase Tabla B.1).

Tabla B.1. Tipos primitivos y clases envoltorio

Tipo primitivo	Clase envoltorio
byte	
short	
int	Integer
long	
boolean	Boolean
float	
double	
char	Character

Todas las clases envoltorio, salvo Character, heredan de la clase abstracta Number.

La forma de construir un wrapper a partir de un valor primitivo, es decir, de "envolverlo", es pasándole el valor como argumento a un método estático de los llamados factoría,

```
Integer x = Integer.valueOf(3);
```

La variable x referencia un objeto Integer que guarda un 3 en su interior. El uso del constructor new se desaconseja por obsoleto.

Los wrappers se pueden manipular como si fueran valores primitivos,

```
Integer y = Integer.valueOf(5);
Integer z = x + y;
System.out.println(z);
```

mostrará en pantalla un 8, ya que la suma entre valores Integer se realiza como si fueran de tipo int.

En realidad, Java se suele encargar de envolver automáticamente los valores primitivos cuando es necesario. Podríamos haber inicializado,

```
Integer x = 3, y = 5;
```

como si fueran del tipo int. Java envuelve el 3 y el 5 antes de asignarlo a x e y respectivamente. Este mecanismo se llama *autoboxing*.

Del mismo modo, los desenvuelve (*unboxing*) cuando hace falta,

```
int w = x;
```

Se desenvuelve el 3 de su envoltorio antes de asignarlo a la variable w.

Algo parecido ocurre cuando queremos realizar operaciones que involucran tipos primitivos y wrappers, como int con Integer, mezclados en la misma expresión,

```
Integer v = x + 9;
```

Java desenvuelve x, lo suma con el 9 y envuelve el resultado para asignarlo a v. Sin embargo, no se aconseja confiar en el autoboxing dentro de un bucle.

Todo lo dicho de int e Integer, puede decirse de las otras parejas primitivo-wrapper.

Con todas las clases envoltorio que heredan de Number se pueden realizar las mismas operaciones que con sus tipos primitivos. Además, son aplicables los operadores relacionales,

```
Double x1 = 1.23, x2 = 4.21;
System.out.println(x1 > x2);
```

muestra por pantalla false, como haría con tipos double. Una excepción es el operador de igualdad. Con las clases envoltorio hay que usar equals() en lugar de ==, ya que se están comparando objetos.

En general, Java envuelve y desenvuelve, según las necesidades, para colocar un tipo primitivo donde se espere primitivo y un wrapper donde se espere un wrapper. Por ejemplo, una función como,

```
void funcion(Integer x) {
        ...
}
```

que tiene un parámetro de tipo Integer, puede ser llamada tanto con un valor de tipo Integer,

```
funcion(Integer.valueOf(5));
```

como int,

```
funcion(5);
```

En este último caso, Java se encarga de envolver el 5 antes de pasarlo a la función.

Las clases envoltorio disponen de una serie de métodos. Especialmente útiles son los que interpretan cadenas de caracteres (a menudo leídas del teclado), que representan valores, y los convierten. Por ejemplo, la clase Double dispone del método:

```
static double parseDouble(String cadena)
```

al que se le pasa una cadena que representa un número real y lo convierte en un valor double,

```
String cad = "23.546";
double t = Double.parseDouble(cad);
System.out.println(t);
```

La variable real t contiene el valor decimal 23.546.

El resto de los wrappers disponen de métodos análogos.