Clases especiales en Java:

Estas clases son especiales porque siendo objetos, en ocasiones se trabaja con ellas como si fuesen tipos básicos. Las más utilizadas son Integer y String.

Empecemos por Integer:

Clase Integer en java

La clase **Integer** encierra un valor primitivo del tipo **int** en un objeto. Un único objeto de tipo entero contiene un único campo cuyo tipo es **int**. Además esta clase proporciona varios métodos para convertir un **int** en una cadena y una cadena en un **int** (ver casteos) así como otras constantes y métodos útiles para tratar un **int**.

Declaración de un Integer en Java

Para declarar un **Integer** en java es muy fácil, es el tipo de dato la variable.
Ejemplo

```
Integer i;
int j;
```

"Ambos tipos son soportados más halla que uno sea como objeto y otro sea primitivo".

Variables la Clase Integer en Java

Las variables static que contiene la clase **Integer** en Java son:

- **MAX_VALUE:** Es el máximo valor que puede contener un **int**.(Margen superior)
- **MIN_VALUE:** Es el mínimo valor que puede contener un **int**. (Margen inferior)
- **SIZE:** Es el número de bits usados para representar un valor **int**.

Prueba

```
System.out.println("valor maximo de un entero: "+Integer.MAX_VALUE);
System.out.println("numero de Bytes de un entero: "+ Integer.BYTES);
System.out.println("valor minimo de un entero"+ Integer.MIN_VALUE);
```

Constructores de la clase Integer en Java

- **Integer(int value)**: Esta constructor crea un nuevo objeto **Integer** con el especifico valor **int** pasado por parámetro.
- **Integer(String s)**:Esta constructor crea un nuevo objeto **Integer** con el especifico valor **String** pasado por parámetro. En el caso que el **String** no se permita pasar a **Integer**, como por ejemplo "R", lanza una excepción del tipo**NumberFormatException**.

Prueba a hacer

```
Integer i= new Integer("25");
Integer j= new Integer(25);
```

Métodos de la Clase Integer en Java

Estos son todos los métodos de la clase *Integer en Java, en amarillo los más usados.*

static int bitCount(int i)
Este método devuelve el número de uno -bits en la representación binaria complemento de dos del valor int especificado.
byte byteValue()
Este método devuelve el valor de este entero como un byte .
int compareTo(Integer anotherInteger)
Este método compara dos objetos Integer numéricamente.
static Integer decode(String nm)
Este método descodifica una cadena en un entero .
double doubleValue()
Este método devuelve el valor de este entero como un doble .
boolean equals(Object obj)
Este método compara este objeto para el objeto especificado .
float floatValue()
Este método devuelve el valor de este entero como un float .
static Integer getInteger(String nm)
Este método permite determinar el valor entero de la propiedad del sistema con el nombre especificado .
static Integer getInteger(String nm, int val)
Este método permite determinar el valor entero de la propiedad del sistema con el nombre especificado .
static Integer getInteger(String nm, Integer val)
Este método devuelve el valor entero de la propiedad del sistema con el nombre especificado.
int hashCode()
Este método devuelve un código hash de este objeto Integer.
static int highestOneBit(int i)
Este método devuelve un valor int con a lo sumo un solo de un bit , en la posición de la orden más alto (" más a la izquierda ") de un bit en el valor int especificado.
int intValue()
Este método devuelve el valor de este entero como un int .
long longValue()
Este método devuelve el valor de este entero como un largo .
static int lowestOneBit(int i)
Este método devuelve un valor int con un máximo de una sola de un bit , en la posición de la orden más bajo (" derecha ") de un bit en el valor int especificado.
static int numberOfLeadingZeros(int i)
Este método devuelve el número de bits cero anteriores a la orden más alta (" izquierda ") de un bit en la representación binaria complemento de dos del valor int especificado.
static int numberOfTrailingZeros(int i)

Este método devuelve el número de bits cero después de la orden más bajo (" derecha ") de un bit en la representación binaria complemento de dos del valor int especificado.

static int parseInt(String s)

Este método analiza el argumento de cadena como un entero decimal con signo .

static int parseInt(String s, int radix)

Este método analiza el argumento de cadena como un entero con signo en la base especificada por el segundo argumento .

static int reverse(int i)

Este método devuelve el valor obtenido invirtiendo el orden de los bits en la representación binaria el complemento a dos del valor int especificado.

static int reverseBytes(int i)

Este método devuelve el valor que se obtiene invirtiendo el orden de los bytes en representación de complemento a dos del valor int especificado.

static int rotateLeft(int i, int distance)

Este método devuelve el valor obtenido mediante la rotación de la representación binaria complemento de dos del valor int especificada dada por el número de bits especificado .

static int rotateRight(int i, int distance)

Este método devuelve el valor obtenido mediante la rotación de la representación binaria complemento de dos del valor int especificada derecha el número de bits especificado .

short shortValue()

Este método devuelve el valor de este entero como un corto .

static int signum(int i)

Este método devuelve la función signum del valor int especificado.

static String toBinaryString(int i)

Este método devuelve una representación de cadena del argumento entero como un entero sin signo en base 2.

static String toHexString(int i)

Este método devuelve una representación de cadena del argumento entero como un entero sin signo en base 16.

static String toOctalString(int i)

Este método devuelve una representación de cadena del argumento entero como un entero sin signo en base 8.

String toString ()

Este método devuelve un objeto String que representa el valor de este Integer.

static String toString (int i)

Este método devuelve un objeto String que representa el entero especificado .

static String toString (int i , int radix)

Este método devuelve una representación de cadena del primer argumento en la base especificada por el segundo argumento .

static Integer valueOf (int i)

Este método devuelve una instancia de tipo Integer que representa el valor int especificado.

static Integer valueOf (String s)

Este método devuelve un objeto Integer que contiene el valor de la cadena especificada.

static Integer valueOf (String s , int radix)

Este método devuelve un objeto Integer que contiene el valor extraído de la cadena especificada cuando se analiza con la base dada por el segundo argumento .

Operadores

Como clase especial, los objetos Integer pueden utilizarse con todos los operadores (+,-,*,/,%)

En las comparaciones de igualdad es importante tener en cuenta que objetos han hecho un new y cuales no. Si hacemos un new, se guardará memoria para el objeto y se guardará una referencia a este objeto, si no, simplemente se guardará el valor. Prueba lo siguiente:

```
Integer i= new Integer(25);
Integer j= new Integer(25);
System.out.println(i==j);
Integer x= 25;
Integer y=25;

System.out.println(x==y);
```

Esto mostrará un false y un true. (¿Que mostrará x==j?)

Clase String

Un String en Java representa una **cadena de caracteres no modificable**.

Todos los literales de la forma "*cualquier texto*", es decir, literales entre comillas dobles, que aparecen en un programa java se implementan como objetos de la clase String.

CREAR UN STRING

Se puede **crear un String** de varias formas, entre ellas:

- Utilizando una **cadena de caracteres** entre comillas:

  String s1 = "abcdef";

- Utilizando **operador de concatenación +**con dos o más objetos String:

  String s2 = s1 + "ghij"; //s2 contiene "abcdefghij"

  String s3 = s1 + s2 + "klm"; //s3 contiene " abcdefabcdefghijklm"

Además la clase String proporciona varios **constructores**, entre ellos:

CONSTRUCTOR	DESCRIPCIÓN
String()	Constructor por defecto. El nuevo String toma el valor "" String s = new String(); //crea el string s vacío.

	Equivale a: String s = "";
String(String s)	Crea un nuevo String, copiando el que recibe como parámetro. String s = "hola"; String s1 = new String(s); //crea el String s1 y le copia el contenido de s
String(char[] v)	Crea un String y le asigna como valor los caracteres contenidos en el array recibido como parámetro. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a); //crea String s con valor "abcde"
String(char[] v, int pos, int n)	Crea un String y le asigna como valor los n caracteres contenidos en el array recibido como parámetro, a partir de la posición pos. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a, 1, 3); //crea String s con valor "bcd";

MÉTODOS DE LA CLASE STRING

La clase String proporciona métodos para el tratamiento de las cadenas de caracteres: acceso a caracteres individuales, buscar y extraer una subcadena, copiar cadenas, convertir cadenas a mayúsculas o minúsculas, etc. Estos son los más utilizados.

MÉTODO	DESCRIPCIÓN
length()	Devuelve la longitud de la cadena
indexOf(‘caracter’)	Devuelve la posición de la primera aparición de carácter
lastIndexOf(‘caracter’)	Devuelve la posición de la última aparición de carácter

charAt(n)	Devuelve el carácter que está en la posición n
substring(n1,n2)	Devuelve la subcadena comprendida entre las posiciones n1 y n2-1
toUpperCase()	Devuelve la cadena convertida a mayúsculas
toLowerCase()	Devuelve la cadena convertida a minúsculas
equals("cad")	Compara dos cadenas y devuelve true si son iguales
equalsIgnoreCase("cad")	Igual que equals pero sin considerar mayúsculas y minúsculas
compareTo(OtroString)	Devuelve 0 si las dos cadenas son iguales. <0 si la primera es alfabéticamente menor que la segunda ó >0 si la primera es alfabéticamente mayor que la segunda.
compareToIgnoreCase(OtroString)	Igual que compareTo pero sin considerar mayúsculas y minúsculas.
valueOf(N)	Método estático. Convierte el valor N a String. N puede ser de cualquier tipo.

Los puedes consultar todos en la API de Java:

http://docs.oracle.com/javase/7/docs/api/index.html?java/lang/String.html

Debemos recordar que:

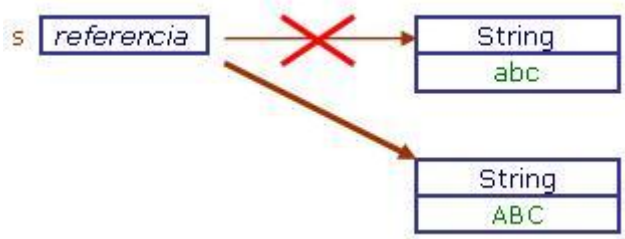
Los objetos String no son modificables.

Por lo tanto, los métodos que actúan sobre un String con la intención de modificarlo lo que hacen es crear un nuevo String a partir del original y devolverlo modificado.

Por ejemplo: Una operación como convertir a mayúsculas o minúsculas un String no lo modificará sino que creará y devolverá un nuevo String con el resultado de la operación.

String s = "abc";

s = s.toUpperCase(); //convertir a mayúsculas el contenido del String s



El **recolector de basura** es el encargado de eliminar de forma automática los objetos a los que ya no hace referencia ninguna variable.

EL OPERADOR DE CONCATENACIÓN +

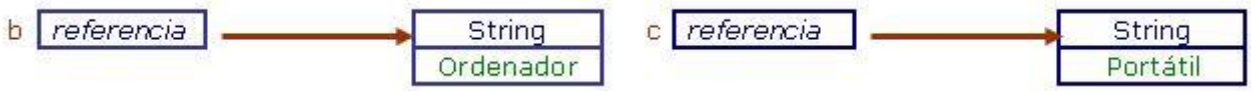
La clase proporciona el operador + (concatenación) para unir dos o más String.

El resultado de aplicar este operador es un nuevo String concatenación de los otros.

Por ejemplo:

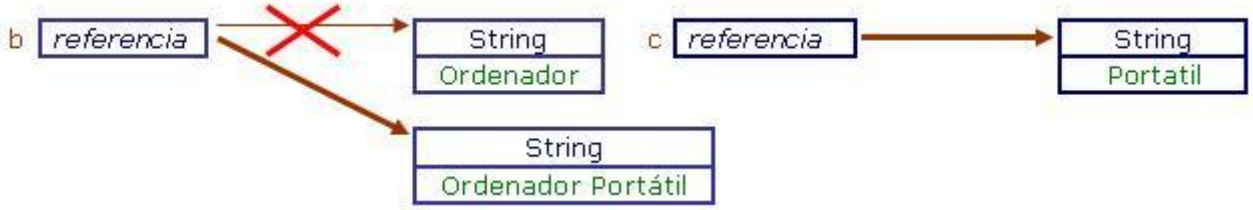
String b = "Ordenador";

String c = " Portátil";



La operación: b = b + c;

crea un nuevo String (b + c) y le asigna su dirección a b:



Operadores

En las comparaciones, existe el mismo problema que con Integer, depende de si has realizado un new o no, podrás comparar con el comparador ==. Por ello se recomienda el uso del método equals() y equalsIgnoreCase().

Prueba:

```
String cadena1="aa";
String cadena2= new String ("aa");
System.out.println(cadena1==cadena2);
System.out.println(cadena1.equals(cadena2));
```

El resultado será false y true