



UNIVERSIDAD DE ALMERÍA

Grado en Ingeniería Informática

Introducción a la Programación

2016-2017



Tema 3. Librerías básicas

- La clase String
- La clase StringBuffer
- Envoltorios de los tipos básicos. Wrappers
- La clase Math
- Las clases BigInteger y BigDecimal

Tema 3. Librerías básicas

La clase **String**

Un *string* es una secuencia de caracteres. En la mayoría de los lenguajes de programación son tratados como array de caracteres, pero en Java un *string* es un objeto. La clase **String** tiene 11 constructores y más de 40 métodos para manipular cadenas.

Se encuentra en el paquete `java.lang`

Está orientada a manejar cadenas de caracteres constantes, es decir, que yo no voy a poder cambiar. Se dice que son **inmutables**.

La clase **StringBuffer** está orientada a manejar cadenas de caracteres variables ó modificables.

Tema 3. Librerías básicas

La clase String

Construcción de un objeto de la clase String

- `String mensaje = new String("Bienvenido a Java");`
- `String mensaje = "Bienvenido a Java";`

Tema 3. Librerías básicas

➡ La clase String

Comparación de cadenas

java.lang.String

+equals(s1: String): boolean

+equalsIgnoreCase(s1: String):
boolean

+compareTo(s1: String): int

+compareToIgnoreCase(s1: String):
int

+startsWith(prefix: String): boolean

+endsWith(suffix: String): boolean

Devuelve true si el string (objeto) es igual a s1.

Igual al anterior pero ignorando las mayúsculas.

Devuelve un entero >0, =0 o <0 para indicar que el string(objeto) es mayor, igual o menor que s1.

Igual al anterior pero ignorando las mayúsculas.

Devuelve true si el string comienza con el prefijo especificado.

Devuelve true si el string termina con el prefijo especificado.

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

```
String s1 = new String("Welcome to Java");  
String s2 = "Welcome to Java";
```

```
if (s1.equals(s2)) //true  => compara contenidos  
    System.out.println("s1 y s2 tienen el mismo contenido");  
else  
    System.out.println("s1 y s2 no tienen el mismo contenido");  
  
if (s1 == s2) //false  => compara referencias  
    System.out.println("s1 y s2 son los mismos objetos");  
else  
    System.out.println("s1 y s2 no son los mismos objetos");
```

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

```
String s1 = "Tomas";  
String s2 = "Alberto";
```

```
if (s1.compareTo(s2) > 0)  
    System.out.println("s1 es mayor que s2 "); ←  
else if (s1.compareTo(s2) < 0)  
    System.out.println("s1 es menor que s2 ");  
else  
    System.out.println("s1 y s2 son iguales");
```

```
String str = " El primer programa ";
```

```
boolean resultado = str.startsWith("El"); → true  
boolean resultado = str.endsWith("programa"); → true
```

Tema 3. Librerías básicas

La clase String

Longitud del string, acceso a caracteres individuales y combinación de string

`java.lang.String`

```
+length(): int  
+charAt(index: int): char  
+concat(s1: String): String
```

Devuelve el número de caracteres del string.

Devuelve el carácter cuya posición se especifica en index

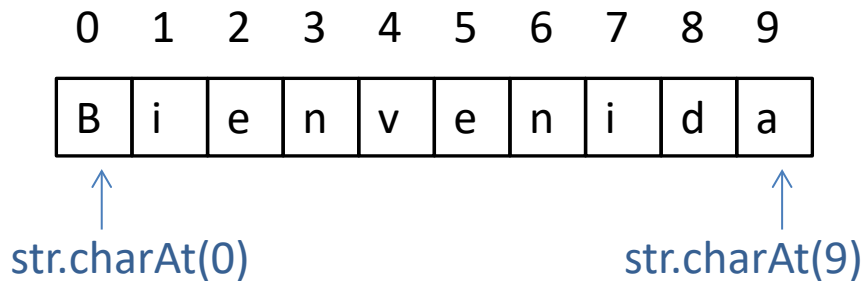
Devuelve un string nuevo concatenando el string con s1

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

`int longitud = str.length();` → 10



```
String str = " Bienvenida";
```

`String s3 = s1.concat(s2)` equivalente a

`String s3 = s1 + s2`

```
String s1 = " Hola";  
String s2 = " que tal";
```

Tema 3. Librerías básicas

La clase String

Obtener substring (subcadenas)

java.lang.String

+substring(beginIndex: int): String

+substring(beginIndex: int, endIndex: int): String

Devuelve una subcadena que comienza con el carácter cuya posición se especifica en beginIndex hasta el final del string. Devuelve una subcadena que comienza en el carácter cuya posición se especifica en beginIndex y termina en el carácter cuya posición es endIndex -1.

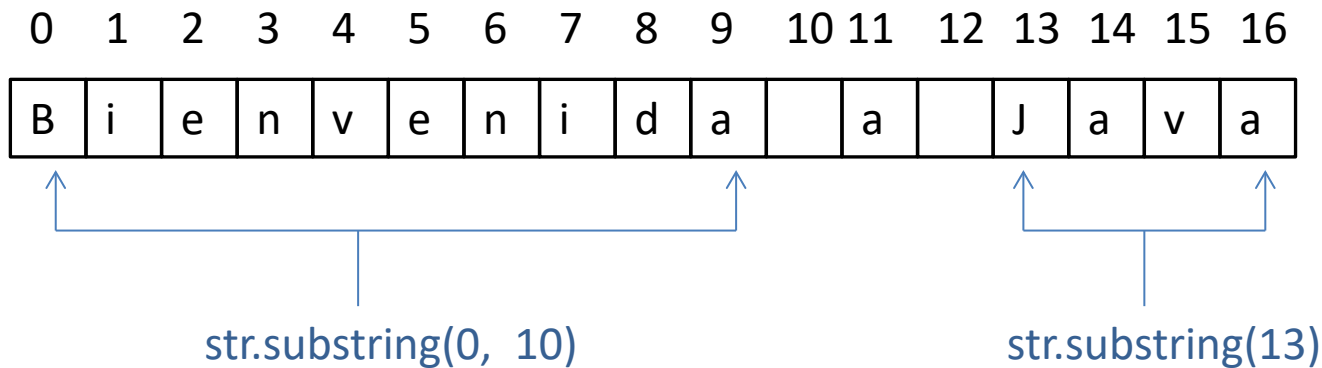
Métodos sobrecargados

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

```
String str = " Bienvenida a Java";
```



Tema 3. Librerías básicas

La clase String

Convertir string

`java.lang.String`

```
+toLowerCase(): String  
+toUpperCase(): String  
+trim(): String
```

Devuelve un nuevo string con los caracteres en minúscula.

Devuelve un nuevo string con los caracteres en mayúscula.

Devuelve un nuevo string eliminando los caracteres blancos al principio y al final.

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

"**Hola**". toLowerCase()

devuelve un nuevo string, **hola**.

"**Hola**". toUpperCase ()

devuelve un nuevo string, **HOLA**.

" **Hola** ".trim()

devuelve un nuevo string, **Hola**. (sin blancos)

Tema 3. Librerías básicas

La clase String

Encontrar un carácter o un substring en un string

java.lang.String

`+indexOf(ch: char): int`

`+indexOf(ch: char, fromIndex: int): int`

`+indexOf(s: String): int`

`+indexOf(s: String, fromIndex: int): int`

Devuelve el índice (posición) de la 1ª ocurrencia de ch en el string.
-1 si no está.

Devuelve el índice de la 1ª ocurrencia de ch después de fromIndex.
-1 si no está.

Devuelve el índice de la 1ª ocurrencia del string s en el string.
-1 si no está.

Devuelve el índice de la 1ª ocurrencia del string s en el string después de fromIndex. -1 si no está.

Métodos sobrecargados

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

"Bienvenida a Java".indexOf('B')	devuelve 0.
"Bienvenida a Java".indexOf('e')	devuelve 2.
"Bienvenida a Java".indexOf('e', 2)	devuelve 5.
"Bienvenida a Java".indexOf("nida")	devuelve 6.
"Bienvenida a Java".indexOf("Java", 5)	devuelve 12.
"Bienvenida a Java".indexOf("java", 5)	devuelve -1.

Tema 3. Librerías básicas

La clase String

Convertir caracteres y valores numéricos en string

java.lang.String

+valueOf(c: char): String
+valueOf(data: char[]): String
+valueOf(d: double): String
+valueOf(f: float): String
+valueOf(i: int): String
+valueOf(l: long): String
+valueOf(b: boolean): String

Devuelve un string que contiene el carácter c.

Devuelve un string que contiene los caracteres del array data.

Devuelve un string que tiene el valor d double.

Devuelve un string que tiene el valor f float.

Devuelve un string que tiene el valor i int.

Devuelve un string que tiene el valor l long.

Devuelve un string que tiene el valor b boolean

¡OJO!

→ Métodos de clase (sobrecargados)

Tema 3. Librerías básicas

La clase String

Ejemplos de uso

`String.valueOf(5.44)` devuelve un string que contiene " 5.44 "

`String.valueOf(10)` devuelve un string que contiene " 10 "

Tema 3. Librerías básicas

La clase **StringBuffer**

La clase **StringBuffer** es una alternativa a la clase **String**, siendo la primera más flexible ya que se puede añadir, insertar, es decir modificar el contenido de un objeto de esa clase. Los objetos **StringBuffer** son cadenas *modificables* a diferencia de los objetos de **String** que son *inmutables*

Constructores

1) **StringBuffer s1 = new StringBuffer ();**

Crea un objeto **StringBuffer** sin caracteres y con una capacidad para 16 caracteres.

2) **StringBuffer s2 = new StringBuffer (10);**

Recibe un argumento entero y crea un objeto que no contiene caracteres y tiene una capacidad inicial que es la indicada en el parámetro.

Tema 3. Librerías básicas

La clase **StringBuffer**

3) **StringBuffer** s3 = new **StringBuffer** (" hola ");

Crea un objeto que contiene los caracteres que hemos pasado como parámetro y con una capacidad inicial igual al número de caracteres del argumento +16.

StringBuffer and **StringBuilder** son más flexibles que **String**. La clase **StringBuilder** es similar a **StringBuffer** excepto que los métodos para modificar el buffer en **StringBuffer** están *sincronizados*, lo que significa que sólo una tarea es permitida para ejecutar los métodos. Utilizaremos **StringBuffer** si la clase puede ser accedida por múltiples tareas al mismo tiempo. El uso de **StringBuilder** es más eficiente si se accede por una sola tarea, ya que no se necesita la sincronización en este caso.

java.lang.StringBuilder

```
+StringBuilder()  
+StringBuilder(capacity: int)  
+StringBuilder(s: String)
```

Tema 3. Librerías básicas

La clase StringBuffer

Modificar strings con StringBuffer

`java.lang.StringBuilder`

```
+append(v: aPrimitiveType): StringBuilder  
+append(s: String): StringBuilder  
+delete(startIndex: int, endIndex: int):  
    StringBuilder  
+deleteCharAt(index: int): StringBuilder  
+insert(offset: int, b: aPrimitiveType):  
    StringBuilder  
+insert(offset: int, s: String): StringBuilder  
+replace(startIndex: int, endIndex: int, s:  
    String): StringBuilder  
+reverse(): StringBuilder  
+setCharAt(index: int, ch: char): void
```

Añade un valor de tipo primitivo como un string.

Añade un string al objeto de la clase StringBuffer.

Borra caracteres desde la posición **startIndex** a **endIndex -1**.

Borra el carácter que está en la posición que indica **index**.

Inserta un valor convertido a string en la posición **offset**.

Inserta un string en la posición **offset**.

Sustituye los caracteres desde **startIndex** hasta **endIndex -1** por el string **s**.

Invierte los caracteres.

Pone el nuevo carácter (**ch**) en la posición indicada por **index**.

Tema 3. Librerías básicas

La clase StringBuffer

Ejemplos de uso

```
StringBuffer stringBuffer = new StringBuffer();  
stringBuffer.append("Bienvenida");  
stringBuffer.append(' ');  
stringBuffer.append("a");  
stringBuffer.append(' ');  
stringBuffer.append("Java");  
stringBuffer.insert(13, "HTML y ");  
stringBuffer.delete(13, 20);  
stringBuffer.reverse();  
stringBuffer.setCharAt(0, 'b');
```

stringBuffer contiene **"Bienvenida a Java"**
stringBuffer contiene **"Bienvenida a HTML y Java"**
stringBuffer contiene **"Bienvenida a Java"**
stringBuffer contiene **"avaJ a adinevneiB "**
stringBuffer contiene **" bvaJ a adinevneiB "**

Tema 3. Librerías básicas



Clases Envoltorios de los tipos básicos. Wrappers

Los **Wrappers** (*envoltorios*) son clases diseñadas para ser un complemento de los **tipos primitivos**. Dichos tipos, son los únicos elementos de Java que no son objetos, ello supone un inconveniente ya que en su inmensa mayoría, los métodos de Java requieren el uso de parámetros o argumentos que son objetos. Para evitar ello, Java ofrece una manera de *envolver* el tipo primitivo y convertirlo en objeto utilizando para cada tipo una clase:

int la clase envoltorio **Integer**

double la clase envoltorio **Double**

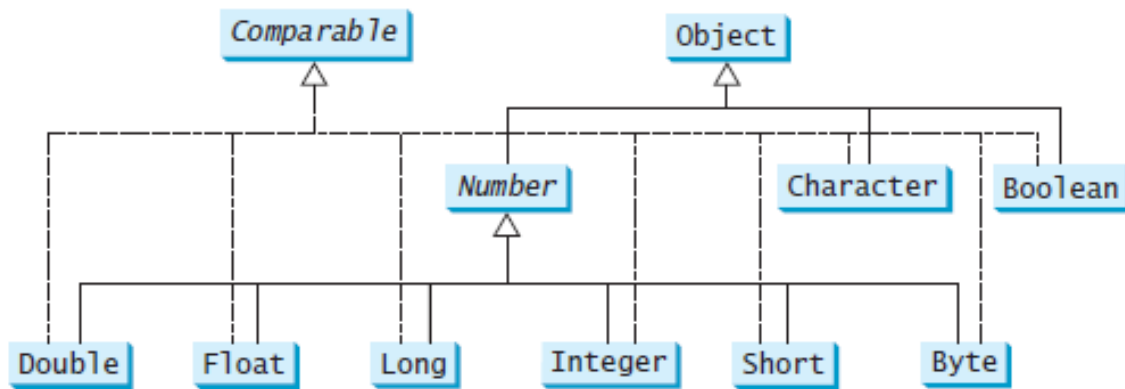
byte la clase envoltorio **Byte**

char la clase envoltorio **Character** etc.

Tema 3. Librerías básicas

➡ Clases Envoltorios de los tipos básicos. Wrappers

Jerarquía de clases



La clase **Number** es una superclase abstracta para **Double**, **Float**, **Long**, **Integer**, **Short** y **Byte**. (estas clases son extensiones, es decir heredan de **Number** los métodos abstractos)

Además las clases envoltorios implementan la interface **Comparable**

Tema 3. Librerías básicas

La clase Character.

Es la clase envoltorio para el tipo primitivo **char**.

Podemos crear un objeto de la clase **Character** a partir de un valor de tipo **char**, utilizando el constructor:

```
Character character = new Character('a'); // el objeto character contiene 'a'.
```


Tema 3. Librerías básicas

La clase Character

java.lang.Character

```
+Character(value: char)
+charValue(): char
+compareTo(anotherCharacter: Character): int
+equals(anotherCharacter: Character): boolean
+isDigit(ch: char): boolean
+isLetter(ch: char): boolean
+isLetterOrDigit(ch: char): boolean
+isLowerCase(ch: char): boolean
+isUpperCase(ch: char): boolean
+toLowerCase(ch: char): char
+toUpperCase(ch: char): char
```

Constructor de un objeto carácter con un valor de tipo char.

Devuelve el valor de tipo char de ese objeto.

Compara un carácter con otro.

Devuelve true si un carácter es igual a otro.

Devuelve true si el carácter ch es un dígito.

Devuelve true si el carácter ch es una letra.

Devuelve true si el carácter ch es un dígito o una letra.

Devuelve true si el carácter ch está en minúscula.

Devuelve true si el carácter ch está en mayúscula.

Devuelve la minúscula del carácter especificado.

Devuelve la mayúscula del carácter especificado.

Tema 3. Librerías básicas

La clase Integer.

Es la clase envoltorio para el tipo primitivo **int**.

Podemos crear un objeto de la clase **Integer** a partir de un valor de tipo **int** utilizando el constructor:

```
Integer entero1= new Integer(25);      // el objeto entero1 contiene 25  
Integer entero2 = new Integer("456" ); // el objeto entero2 contiene 456
```

Tema 3. Librerías básicas

La clase Double.

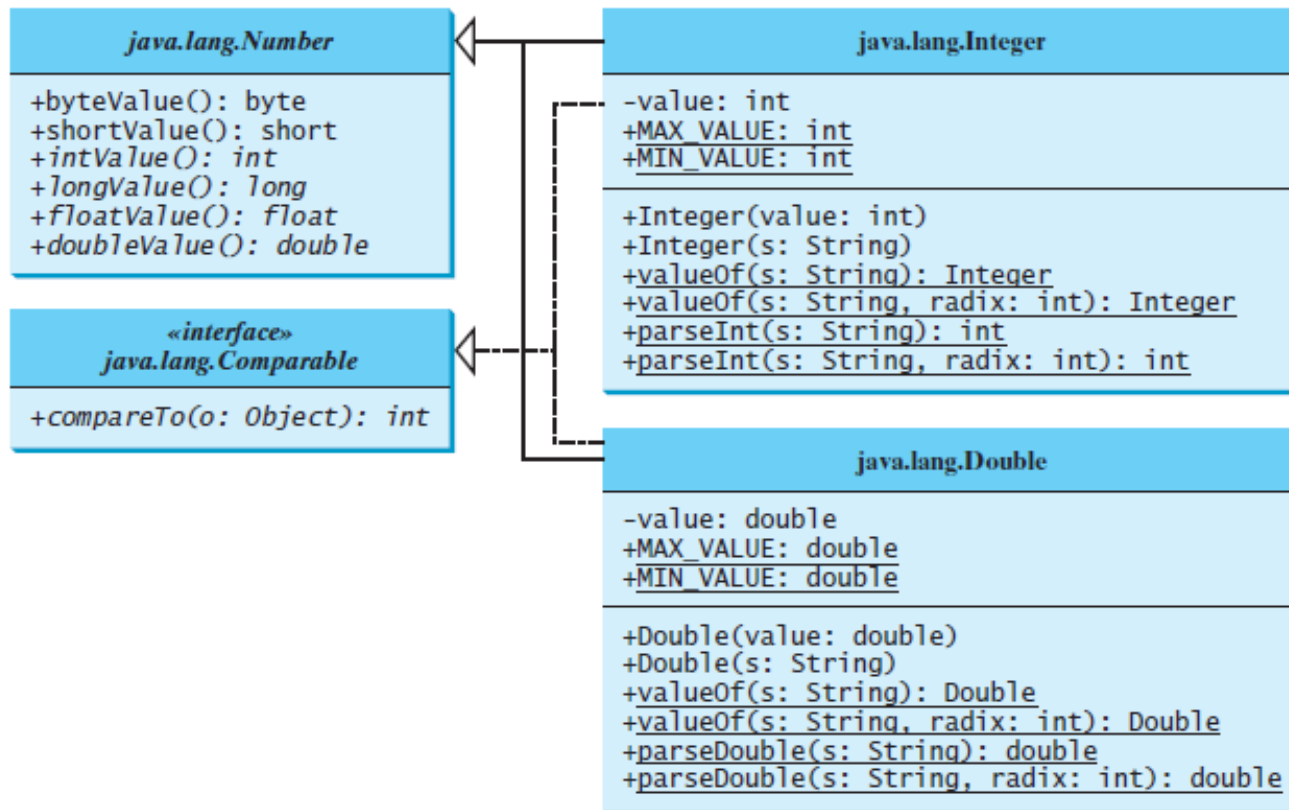
Es la clase envoltorio para el tipo primitivo **double**.

Podemos crear un objeto de la clase **Double** a partir de un valor de tipo **double** utilizando el constructor:

```
Double real1= new Double(25.0);           // el objeto real1 contiene 25.0  
Double real2 = new Double("456.75" );    // el objeto real2 contiene 456.75
```

Tema 3. Librerías básicas

➡ La clase Integer y Double.



Ejemplo de uso

```

1 package org.ip.tema03;
2
3 public class EjemplosEnvoltorios {
4
5     public static void main(String[] args) {
6         Character character = new Character('a');
7         boolean digito = Character.isDigit('5');    // metodo de clase
8         System.out.println(digito);
9         int valor = character.compareTo('j');    // metodo de objeto, interfaz Comparable
10        System.out.println(valor);
11        char letra = Character.toUpperCase('a');    // metodo de clase
12        System.out.println(letra);
13        Integer entero1 = new Integer(45);
14        Integer entero2 = new Integer("456");
15        System.out.println("El máximo valor entero es " + Integer.MAX_VALUE);
16        System.out.println("El mínimo valor real es " + Double.MIN_VALUE);
17        int valor1 = entero1.intValue();    // metodo de Number y de objeto
18        System.out.println(valor1);
19        int valor2 = entero2.intValue();    // metodo de Number y de objeto
20        System.out.println(valor2);
21        Double real1 = new Double(25.0);
22        Double real2 = new Double("456.75");
23        double valor3 = real1.doubleValue();    // metodo de Number y de objeto
24        System.out.println(valor3);
25        double valor4 = real2.doubleValue();    // metodo de Number y de objeto
26        System.out.println(valor4);
27        double valor5 = Double.parseDouble("788.90");    // metodo de clase
28        System.out.println(valor5);
29    }
30 }

```

Salida

```

true
-9
A
El máximo valor entero es 2147483647
El mínimo valor real es 4.9E-324
45
456
25.0
456.75
788.9

```

Ejemplo de uso

```
1 package org.ip.tema03;
2
3 public class ConvertirBinarioADecimal {
4
5     public static void main(String[] args) {
6         String strNumeroBinario = "111000";
7
8         // Convierte un número binario a decimal utilizando
9         // el método de clase parseInt de la clase Integer
10        // El segundo argumento indica la base
11
12        int numeroDecimal = Integer.parseInt(strNumeroBinario, 2);
13        System.out.println("El número binario " + strNumeroBinario
14            + " convertido a número decimal es " + numeroDecimal);
15    }
16 }
```

Salida El número binario 111000 convertido a número decimal es 56

Tema 3. Librerías básicas

La clase Math

Es una clase que tiene:

- métodos de clase
- constantes E y PI

Ejemplos de uso:

```
double area = Math.PI * radio * radio;  
double raiz = Math.sqrt(25);
```

java.lang.Math
<ul style="list-style-type: none">• <u>E: double</u>• <u>PI: double</u>
<ul style="list-style-type: none">• <u>sin(arg0: double): double</u>• <u>cos(arg0: double): double</u>• <u>tan(arg0: double): double</u>• <u>asin(arg0: double): double</u>• <u>acos(arg0: double): double</u>• <u>atan(arg0: double): double</u>• <u>toRadians(arg0: double): double</u>• <u>toDegrees(arg0: double): double</u>• <u>exp(arg0: double): double</u>• <u>log(arg0: double): double</u>• <u>log10(arg0: double): double</u>• <u>sqrt(arg0: double): double</u>• <u>ceil(arg0: double): double</u>• <u>floor(arg0: double): double</u>• <u>atan2(arg0: double, arg1: double): double</u>• <u>pow(arg0: double, arg1: double): double</u>• <u>round(arg0: float): int</u>• <u>round(arg0: double): long</u>• <u>random(): double</u>• <u>abs(arg0: int): int</u>• <u>abs(arg0: long): long</u>• <u>abs(arg0: float): float</u>

Tema 3. Librerías básicas

Las clases *BigInteger* y *BigDecimal*

Si necesitamos utilizar enteros muy grandes o reales con muy alta precisión, debemos usar las clases *BigInteger* y *BigDecimal* del paquete **java.math**. Ambas son *inmutables*, heredan de la clase *Number* e implementan la interface *Comparable*. El mayor entero de tipo long es Long.MAX_VALUE (9223372036854775807). Una instancia o ejemplar u objeto de *BigInteger* puede representar un entero de cualquier tamaño.

Usaremos: `new BigInteger(String)` y `new BigDecimal(String)` para crear una instancia de *BigInteger* y *BigDecimal*.

Usaremos: **add**, **subtract**, **multiple**, **divide** para realizar operaciones aritméticas y **compareTo** para comparar dos números grandes.

Tema 3. Librerías básicas

Las clases BigInteger y BigDecimal

Ejemplo de uso:

```
BigInteger a = new BigInteger("9223372036854775807");  
BigInteger b = new BigInteger("2");  
BigInteger c = a.multiply(b);    // 9223372036854775807 * 2  
System.out.println(c);
```

Salida: 18446744073709551614

Tema 3. Librerías básicas

Las clases BigInteger y BigDecimal

Ejemplo de uso:

```
BigDecimal a = new BigDecimal(1.0);
```

```
BigDecimal b = new BigDecimal(3);
```

```
BigDecimal c = a.divide(b, 20, BigDecimal.ROUND_UP);
```

```
System.out.println(c);
```

Salida: 0.3333333333333333333333334.

Ejemplo de uso:

```
1 package org.ip.tema03;
2
3 import java.math.BigInteger;
4
5 public class FactorialGrande {
6
7     public static BigInteger factorial(int n) {
8         BigInteger resultado = BigInteger.ONE; // equivalente a new BigInteger("1")
9         for (int i = 1; i <= n; i++)
10             resultado = resultado.multiply(new BigInteger(i + ""));
11         return resultado;
12     }
13
14     public static void main(String[] args) {
15         System.out.println("50! = " + factorial(50));
16     }
17 }
```

Salida 50! = 30414093201713378043612608166064768844377641568960512000000000000



¡MUCHAS GRACIAS!



UNIVERSIDAD DE ALMERÍA

