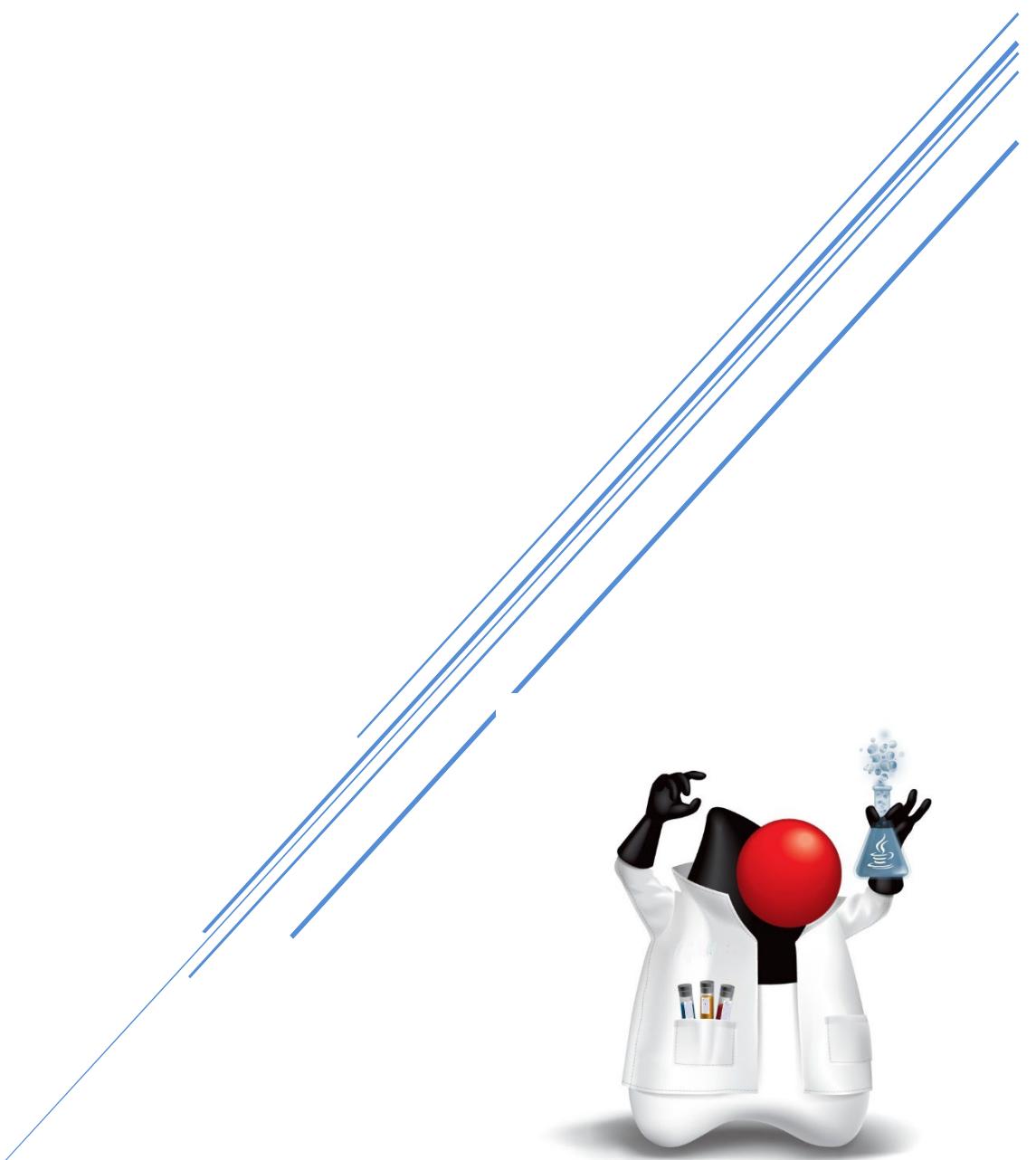


STRING

PROGRAMACIÓN



Contenido

| | |
|---|---|
| La clase <i>String</i> | 2 |
| Cómo se obtiene información acerca del string | 2 |
| Comparación de strings..... | 3 |
| Extraer un substring de un string..... | 4 |
| Convertir un número a string..... | 5 |
| Convertir un string en número..... | 5 |

La clase *String*

Dentro de un objeto de la clases *String* o *StringBuffer*, Java crea un array de caracteres de una forma similar a como lo hace el lenguaje C++. A este array se accede a través de las funciones miembro de la clase.

Los strings u objetos de la clase *String* se pueden crear explícitamente o implícitamente. Para crear un string implícitamente basta poner una cadena de caracteres entre comillas dobles. Por ejemplo, cuando se escribe, Java crea un objeto de la clase *String* automáticamente.

```
System.out.println("El primer programa");
```

Para crear un string explícitamente escribimos

```
String str=new String("El primer programa");
```

También se puede escribir, alternativamente

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

Para crear un string nulo se puede hacer de estas dos formas

```
String str="";
String str=new String();
```

Un string nulo es aquél que no contiene caracteres, pero es un objeto de la clase *String*. Sin embargo,

```
String str;
```

está declarando un objeto *str* de la clase *String*, pero aún no se ha creado ningún objeto de esta clase.

Cómo se obtiene información acerca del string

Una vez creado un objeto de la clase *String* podemos obtener información relevante acerca del objeto a través de las funciones miembro.

Para obtener la longitud, número de caracteres que guarda un string se llama a la función miembro ***length***.

```
String str="El primer programa";
int longitud=str.length();
```

Podemos conocer si un string comienza con un determinado prefijo, llamando al método ***startsWith***, que devuelve **true** o **false**, según que el string comience o no por dicho prefijo

```
String str="El primer programa";
boolean resultado=str.startsWith("El");
```

En este ejemplo la variable resultado tomará el valor **true**.

De modo similar, podemos saber si un string finaliza con un conjunto dado de caracteres, mediante la función miembro **endsWith**.

```
String str="El primer programa";
boolean resultado=str.endsWith("programa");
```

Si se quiere obtener la posición de la primera ocurrencia de la letra p, se usa la función **indexOf**.

```
String str="El primer programa";
int pos=str.indexOf('p');
```

Para obtener las sucesivas posiciones de la letra p, se llama a otra versión de la misma función

```
pos=str.indexOf('p', pos+1);
```

El segundo argumento le dice a la función **indexOf** que empiece a buscar la primera ocurrencia de la letra p a partir de la posición *pos+1*.

Otra versión de **indexOf** busca la primera ocurrencia de un substring dentro del string.

```
String str="El primer programa";
int pos=str.indexOf("pro");
```

Vemos que una clase puede definir varias funciones miembro con el mismo nombre pero que tienen distinto número de parámetros o de distinto tipo.

Comparación de strings

La comparación de strings nos da la oportunidad de distinguir entre el operador lógico **==** y la función miembro **equals** de la clase *String*. En el siguiente código

```
String str1="El lenguaje Java";
String str2=new String("El lenguaje Java");
if(str1==str2) {
    System.out.println("Los mismos objetos");
} else {
    System.out.println("Distintos objetos");
}
if(str1.equals(str2)) {
    System.out.println("El mismo contenido");
} else {
    System.out.println("Distinto contenido");
}
```

Esta porción de código devolverá que *str1* y *str2* son distintos objetos pero con el mismo contenido. *str1* y *str2* ocupan posiciones distintas en memoria pero guardan los mismos datos.

Cambiemos la segunda sentencia y escribamos

```
String str1="El lenguaje Java";
String str2=str1;
System.out.println("Son el mismo objeto "+(str1==str2));
```

Los objetos *str1* y *str2* guardan la misma referencia al objeto de la clase *String* creado. La expresión (*str1==str2*) devolverá **true**.

Así pues, el método **equals** compara un string con un objeto cualquiera que puede ser otro string, y devuelve **true** cuando dos strings son iguales o **false** si son distintos.

```
String str="El lenguaje Java";
boolean resultado=str.equals("El lenguaje Java");
```

La variable *resultado* tomará el valor **true**.

La función miembro **comapareTo** devuelve un entero menor que cero si el objeto string es menor (en orden alfabético) que el string dado, cero si son iguales, y mayor que cero si el objeto string es mayor que el string dado.

```
String str="Tomás";
int resultado=str.compareTo("Alberto");
```

La variable entera *resultado* tomará un valor mayor que cero, ya que Tomás está después de Alberto en orden alfabético.

```
String str="Alberto";
int resultado=str.compareTo("Tomás");
```

La variable entera *resultado* tomará un valor menor que cero, ya que Alberto está antes que Tomás en orden alfabético.

Extraer un substring de un string

En muchas ocasiones es necesario extraer una porción o substring de un string dado. Para este propósito hay una función miembro de la clase *String* denominada *substring*. Para extraer un substring desde una posición determinada hasta el final del string escribimos

```
String str="El lenguaje Java";
String subStr=str.substring(12);
```

Se obtendrá el substring "Java".

Una segunda versión de la función miembro *substring*, nos permite extraer un substring especificando la posición de comienzo y la el final.

```
String str="El lenguaje Java";
String subStr=str.substring(3, 11);
```

Se obtendrá el substring "lenguaje". Recuérdese, que las posiciones se empiezan a contar desde cero.

Convertir un número a string

Para convertir un número en string se emplea la [función miembro estática](#) `valueOf` (más adelante explicaremos este tipo de funciones).

```
int valor=10;
String str=String.valueOf(valor);
```

La clase `String` proporciona versiones de `valueOf` para convertir los datos primitivos: **int, long, float, double**.

Esta función se emplea mucho cuando programamos applets, por ejemplo, cuando queremos mostrar el resultado de un cálculo en el área de trabajo de la ventana o en un control de edición.

Convertir un string en número

Cuando introducimos caracteres en un control de edición a veces es inevitable que aparezcan espacios ya sea al comienzo o al final. Para eliminar estos espacios tenemos la función miembro `trim`

```
String str=" 12 ";
String str1=str.trim();
```

Para convertir un string en número entero, primero quitamos los espacios en blanco al principio y al final y luego, llamamos a la función miembro estática `parseInt` de la clase `Integer` (clase envolvente que describe los números enteros)

```
String str=" 12 ";
int numero=Integer.parseInt(str.trim());
```

Para convertir un string en número decimal (**double**) se requieren dos pasos: convertir el string en un objeto de la clase envolvente `Double`, mediante la función miembro estática `valueOf`, y a continuación convertir el objeto de la clase `Double` en un tipo primitivo **double** mediante la función `doubleValue`

```
String str="12.35 ";
double numero=Double.valueOf(str).doubleValue();
```

Se puede hacer el mismo procedimiento para convertir un string a número entero

```
String str="12";
int numero=Integer.valueOf(str).intValue();
```

Ejercicios:

Juego del Ahorcado: Hay que realizar la librería String, el juego del ahorcado, para ello:

Se debe solicitar por pantalla una palabra que posteriormente se deberá de averiguar introduciendo letra a letra.

No se deben usar arrays, todo el proceso se debe realizar con Strings

Como decoración, podemos insertar el siguiente código...

```
switch (iFallos) {  
    case 0:  
        System.out.println(" -----");  
        for (int j = 0; j < 15; j++) {  
            System.out.println(" | ");  
  
        }  
        System.out.println(" _____");  
        break;  
  
    case 1:  
        System.out.println(" -----");  
        System.out.println(" |      |");  
        System.out.println(" |      |");  
        System.out.println(" |      -----");  
        System.out.println(" |      | - - |");  
        System.out.println(" |      | o |");  
        System.out.println(" |      -----");  
        for (int j = 0; j < 10; j++) {  
            System.out.println(" | ");  
  
        }  
        System.out.println(" _____");  
        break;  
  
    case 2:  
        System.out.println(" -----");  
        System.out.println(" |      |");  
        System.out.println(" |      |");  
        System.out.println(" |      -----");  
        System.out.println(" |      | - - |");  
        System.out.println(" |      | o |");  
        System.out.println(" |      -----");  
        System.out.println(" |      |   ");  
        System.out.println(" |      |   ");  
        System.out.println(" |      |   ");
```

```
System.out.println(" |      |  ");
System.out.println(" |      |  ");
for (int j = 0; j < 5; j++) {
    System.out.println(" | |");

}

System.out.println(" _____");
break;
```

case 3:

```
System.out.println(" -----");
System.out.println(" |      | ");
System.out.println(" |      | ");
System.out.println(" |      -----");
System.out.println(" |      | - - | ");
System.out.println(" |      | o | ");
System.out.println(" |      -----");
System.out.println(" |      |   ");
System.out.println(" |      / |  ");
System.out.println(" |      / |  ");
System.out.println(" |      / |  ");
System.out.println(" |      |  ");
for (int j = 0; j < 5; j++) {
    System.out.println(" | |");
```

```


}
System.out.println(" _____");
break;
```

case 4:

```
System.out.println(" -----");
System.out.println(" |      | ");
System.out.println(" |      | ");
System.out.println(" |      -----");
System.out.println(" |      | - - | ");
System.out.println(" |      | o | ");
System.out.println(" |      -----");
System.out.println(" |      |   ");
System.out.println(" |      / | \\" );
System.out.println(" |      / | \\" );
System.out.println(" |      / | \\" );
System.out.println(" |      |  ");
for (int j = 0; j < 5; j++) {
    System.out.println(" | |");
```

```
    }
    System.out.println("_____");
    break;
```

case 5:

```
    System.out.println(" -----");
    System.out.println(" |      |");
    System.out.println(" |      |");
    System.out.println(" |      -----");
    System.out.println(" |      | - - |");
    System.out.println(" |      | o |");
    System.out.println(" |      -----");
    System.out.println(" |      |   ");
    System.out.println(" |      / | \\" );
    System.out.println(" |      / | \\" );
    System.out.println(" |      / | \\" );
    System.out.println(" |      |   ");
    System.out.println(" |      /   ");
    System.out.println(" |      /   ");
    System.out.println(" |      /   ");
    for (int j = 0; j < 2; j++) {
        System.out.println(" |");
    }
    System.out.println("_____");
    break;
```

case 6:

```
    System.out.println(" -----");
    System.out.println(" |      |");
    System.out.println(" |      |");
    System.out.println(" |      -----");
    System.out.println(" |      | X X |");
    System.out.println(" |      | o |");
    System.out.println(" |      -----");
    System.out.println(" |      |   ");
    System.out.println(" |      / | \\" );
    System.out.println(" |      / | \\" );
    System.out.println(" |      / | \\" );
    System.out.println(" |      |   ");
    System.out.println(" |      / \\" );
    System.out.println(" |      / \\" );
    System.out.println(" |      / \\" );
    for (int j = 0; j < 2; j++) {
        System.out.println(" |");
    }
```

```
    }
    System.out.println("_____ ");
    System.out.println("GAME OVER");
    break;
}
```