

Clase 8: Strings

Problema. Leer una lista de palabras y escribir la más larga (de más letras) y la mayor (alfabéticamente). El fin de la lista se indica con la palabra "fin"

Ejemplo:

```
Palabra(o fin)? gabriela
Palabra(o fin)? jose
Palabra(o fin)? rosa
Palabra(o fin)? matias
Palabra(o fin)? fin
Más larga = gabriela
Mayor = rosa
```

Algoritmo

```
masLarga = "", mayor="", //palabras de cero letras

while(true)
{
    obtener una palabra
    if( palabra == "fin" ) break;
    if( largo de palabra > largo de masLarga ) masLarga=palabra;
    if( palabra > mayor ) mayor = palabra;
}
escribir masLarga y mayor
```

Programa

```
String masLarga="", mayor="";
while(true)
{
    String palabra=U.readLine("palabra(o fin)?");

    if(palabra.equals("fin")) break;

    if(palabra.length() > masLarga.length())
        masLarga=palabra;

    if( palabra.compareTo(mayor) > 0)
        mayor=palabra;
}
U.println("mas larga=" + masLarga);
U.println("mayor=" + mayor);
```

Explicaciones

1. String masLarga="", mayor="";

- inicializa con strings vacíos (cero caracteres)
- equivalencia: String masLarga; masLarga="";
- String: clase predefinida (no es tipo, comienza con mayúscula)
- strings son objetos (no variables)
- operaciones a través de métodos (no operadores): objeto.metodo(argumentos)

2. U.readLine("\n")

- lee una línea (desde el teclado) entrega string que contiene todos los caracteres antes de tecla enter Ej: "la casa"

3. palabra.equals("fin")

- compara Strings palabra y "fin"
- devuelve true si son iguales, o false si son distintos
- resultado de tipo boolean

4. palabra.length()

- entrega cantidad de caracteres del string
- resultado de tipo int

5. palabra.compareTo(mayor)

- compara Strings palabra y mayor
- devuelve 0 si palabra=mayor, N°<0 si palabra<mayor, N°>0 si palabra>mayor
- resultado de tipo int
- comparacion lexicográfica extensión de comparación alfabética ("diccionario") para incluir otros caracteres
- orden: espacio<dígitos<mayúsculas<minúsculas
otros caracteres con representaciones arbitrarias

Clase 8: Strings

Tipo char

- para caracteres individuales (letra, dígito, signo especial)
- cada caracter se representa en 16 bits (2 bytes)
- convención UNICODE (extensión de ASCII)
- variables: ej: `char c;`
- constantes: 'un caracter' (entre apóstrofes)
ej: `' ', 'a', 'A', '8', '='`
- asignación: ej: `c='a';`
- comparación: caracter operador-relación caracter
orden entre caracteres:
`' ' < '0' < '1' < ... < '9' < 'A' < ... < 'Z' < 'a' < ... < 'z'`

Principales métodos de clase String

Sintaxis	Significado	tipo	Ej: <code>String s="casa"</code>
<code>x.length()</code>	Nº de caracteres	int	<code>s.length()</code> =4
<code>x.equals(y)</code>	¿x es igual a y?	boolean	<code>s.equals("casa")</code> =true <code>s.equals("Casa")</code> =false
<code>x.compareTo(y)</code>	0 si <code>x == y</code> Nº < 0 si <code>x < y</code> Nº > 0 si <code>x > y</code>	int	<code>s.compareTo("casa")</code> =0 <code>s.compareTo("casas")</code> <0 <code>s.compareTo("Casa")</code> >0
<code>x.charAt(i)</code>	carácter ubicado en el índice i (desde 0)	char	<code>s.charAt(2)</code> ='s' <code>s.charAt(0)</code> ='c' <code>s.charAt(4)</code> error
<code>x.indexOf(y)</code>	índice de primer y en x (-1 si no está)	int	<code>s.indexOf("as")</code> =1 <code>s.indexOf("a")</code> =1 <code>s.indexOf("hola")</code> =1
<code>x.indexOf(y,i)</code>	índice de y en x (a partir de i)	int	<code>s.indexOf("a",2)</code> =3

Prob : contar las apariciones de un carácter en un string

Ej: `cuenta('a', "abracadabra")` entrega 5

Solución

```
static public int cuenta(char x,String y)
{
    int n=0;
    int i=0;
    while(i<y.length()){
        if(y.charAt(i) == x) n=n+1;
        i=i+1;
    }
    return n;
}
```

Métodos que devuelven String (y no modifican string original)

<code>x.substring(i,j)</code>	string con caracteres entre índices i y j-1	<code>s.substring(1,3)</code>	"as"
<code>x.substring(i)</code>	<code>x.substring(i,x.length())</code>	<code>s.substring(1)</code>	"asa"
<code>x.concat(y)</code>	concatena x e y (añade y al final de x)	<code>s.concat("do")</code>	"casado"
<code>x.replace(y,z)</code>	reemplaza todos los caracteres y por z	<code>s.replace('a','e')</code>	"cese"
<code>x.toUpperCase()</code>	reemplaza minúsculas por mayúsculas	<code>s.toUpperCase()</code>	"CASA"
<code>x.toLowerCase()</code>	reemplaza mayúsculas por minúsculas	<code>s.toLowerCase()</code>	"casa"
<code>x.trim()</code>	elimina espacios al comienzo y fin	"a b ".trim()	"a b"

Problema. Obtener una respuesta "si" o "no"

Ejemplo de diálogo

me quieres? si o no? mmm...
me quieres? si o no? no sé
me quieres? si o no? si
yo no

Programa usando una función que obtenga una respuesta si o no

```
if(siOno("me quieres? ").equals("si"))
    U.println("yo no");
else
    U.println("adios mundo cruel");
```

Solución iterativa

```
static public String siOno(String x){
    String r; //respuesta
    while(true){
        r=U.readLine(x+" si o no?");//leer rpta
        r=r.trim(); //eliminar espacios
        r=r.toLowerCase();//a minusculas
        if(r.equals("si")||r.equals("no"))break;
    }
    return r;
}
```

Solución recursiva

```
String r=U.readLine(x+" si o no?")
    .trim().toLowerCase(); //encadenamiento
if(r.equals("si")||r.equals("no"))
    return r;
else
    return siOno(x);
```

Clase 8: Strings

Ejercicio

```
//repetir string x, y veces
//Ej:repetir("ja",3)="jajaja", repetir("ja",0)=""
static public String repetir(String x,int y){
```

```
...
}
//dibujar un cuadrado
static public void main(String[]x){
...
}
```

Diálogo:

lado cuadrado?4

```
* * * *
*      *
*      *
* * * *
```

Nota. Para conseguir un buen dibujo a cada * se añade un espacio

Solución iterativa

```
//repetir string x, y veces
```

```
static public String repetir(String x,int y)
```

```
{
    String s = "";
    //repetir y veces
    for(int i=1; i<=y; ++i){
        //añadir x a string con resultado
        s = s.concat(x);
    }
```

```
    return s;
```

Nota. s = s.concat(x) se puede escribir s=s+x o s+=x

Solución recursiva

```
//repetir string x, y veces
```

```
static public String repetir(String x,int y)
```

```
{
    if( y <= 0 )
        return "";
    else
        return x+repetir(x,y-1);
    //return x.concat(repetir(x,y-1));
}
```

usando operador de expression condicional:

```
return y<=0 ? "" : x+repetir(x,y-1);
```

```
//dibujar un cuadrado
```

```
static public void main(String[]x)
```

```
{
    //obtener longitud del lado
    int n=U.readInt("lado cuadrado?");

    //mostrar primera línea
    U.println( repetir(" " ,n) );

    //mostrar n-2 líneas intermedias
    for(int i=1; i<=n-2; ++i)
        U.println(" " + repetir(" ",n-2) + " ");

    //mostrar última línea
    U.println( repetir(" " ,n) );
}
```

Conversión de números a String

Ejemplos:

```
String s = ""+n; //si n=123, s="123"
```

```
String s = ""+x; //si x=4.5, s="4.5"
```

Conversión de String a n° entero

```
int n=Integer.parseInt(s);//s="123",n=123
```

```
class Integer{//clase predefinida
```

```
...
static public int parseInt(String x){...}
}
```

Conversion de String a n° real

```
double x=Double.parseDouble(s);//s="4.5",x=4.5
```

```
class Double{//clase predefinida
```

```
...
static public double parseDouble(String x){...}
}
```

Problemas propuestos

- String **inverso**(String x)//ej: inverso("roma")="amor"
- boolean **palindrome**(String x)//palabra capicúa?
ej: palindrome("reconocer")=true
- boolean **alfabetico**(String x)
ej: alfabetico("hola")=true, alfabetico("123")=false
- boolean **esVálido**(String x,String y)
ej: válido("123","0123456789")=true
- String **reemplazar**(String x,String y,String z)
ej: reemplazar("abcde","bc","BC")="aBCde"
- String **enPalabras**(int x)//de 3 dígitos
ej: enPalabras(666) entrega "seis cientos sesenta y seis"
- int **parseInt**(String x)