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

Ejemplo:

```
Palabra(o fin) ? calculo
Palabra(o fin)? algebra
Palabra(o fin)? fisica
Palabra(o fin)? computacion
Palabra(o fin) ? fin
Más larga = computacion
Mayor = fisica
```

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("…")`**

- lee una línea (desde el teclado) entrega string que contiene todos los caracteres antes de tecla enter Ej: "la casa"
- si se detecta el fin de los datos entrega el valor **null** (no "")

3. **`palabra.equals("fin")`**

- compara Strings palabra y "fin"
- devuelve true sin 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,
  Nº<0 si palabra<mayor,
  Nº>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

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'`
- operaciones aritméticas (entre representaciones internas)
  ej: `'c'-'a'` entrega 2 (tipo int)
  `(char)('a'+2)` entrega `'c'`

Principales métodos de clase String

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

Métodos que devuelven String (y no modifican string original)

x.substring(i,j)	string con caracteres entre índices i y j-1	s.substring(1,3) s.substring(2,3)	"as" "s"
x.substring(i)	x.substring(i,x.length())	s.substring(1)	"asa"
x.concat(y)	concatena x e y (añade y al final de x)	s.concat("do") s + "do"	"casado"
x.replace(y,z)	reemplaza todos los caracteres y por z	s.replace('a','e')	"cese"
x.toUpperCase()	reemplaza minúsculas por mayúsculas	s.toUpperCase()	"CASA"
x.toLowerCase()	reemplaza mayúsculas por minúsculas	s.toLowerCase()	"casa"
x.trim()	elimina espacios al comienzo y fin	" a b ".trim()	"a b"

Problema. Función que obtenga una respuesta "si" o "no"

```
static public String siOno(String x){
  String r; //respuesta
  while(true){
    //r=U.readLine(…).trim().toLowerCase();
    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;
}
```

Uso

```
if(siOno("otro?").equals("si"))...else...
```

alternativamente

```
switch(siOno("otro?").charAt(0)){//admite char
case 's': ...
case 'n': ...
}
```

Prob : contar las apariciones de un string en otro
Ej: `cuenta("abra","abracadabra") entrega 2`

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

iterativa

```
int n=0, l=x.length();
//repetir mientras se encuentre x
for(int i=0; (i=y.indexOf(x,i))>=0; i+=l)
  ++n;
return n;
```

recursiva

```
int i=y.indexOf(x);
if(i<0) return 0;
return 1+cuenta(x,y.substring(i+x.length()));
```

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 = “”;
  for(int i=1; i<=y; ++i)//repetir y veces
    s = s.concat(x);
  return s;
}
```

Nota. `s = s.concat(x)` se puede escribir

```
1. s = s + x
2. 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));
}
```

con operador de exp 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 nº entero

```
int n=Integer.parseInt(s);//s=”123”,n=123
class Integer{//clase predefinida
…
static public int parseInteger(String x){…}
}
```

Conversion de String a nº 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”
- String **siOno**()//recursivo
- 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 **parseInteger**(String x)