Los enumerados permiten la personalización de tipos de datos primitivos
Un tipo enumerado en Java es un tipo “especial” que en cierta medida puede usarse como una clase y admite ciertas posibilidades especiales.

```
public enum Equipo {
Madrid
Barca
Sevilla
}
```

Vectores

```
String[] palabras = new String[n_palabras];
int[] numeros = new int [4];
```

MATRIZ

```
int matriz[][] = new int[6][5];

for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[j].length; j++) {
        matriz[i][j] = ‘-’;


    }
}
```

Algoritmos de búsqueda.

Secuencial Lineal
Recorrer el array desde el inicio hasta encontrar el numero buscado.
Un for y un if

Binaria log2n
ha de cumplir la condición principal de que el vector se encuentre ordenado, se basa en ir “partiendo” el vector en 2 en cada iteración

```
int[] vector = {14, 25, 36, 47, 52, 53, 68, 76, 100, 112, 145,
178, 541};
int numero, inferior, superior, mitad;
boolean encontrado = false;
Scanner teclado = new Scanner(System.in);
System.out.println("Introduce el número a buscar");
numero = teclado.nextInt();
inferior = 0;
superior = vector.length-1;
System.out.println("Al iniciar inferior:"+inferior+"
superior:"+superior);
do {
```

```
mitad = (superior + inferior) / 2;
if (numero == vector[mitad]) {
encontrado = true;
} else {
if (numero < vector[mitad]) {
superior = mitad-1 ;
} else {
inferior = mitad+1 ;
}
}
System.out.println("Al salir vale mitad:"+mitad+"
inferior:"+inferior+" superior:"+superior);
} while (inferior <= superior && !encontrado);
if (encontrado) {
System.out.println("El número " + numero + "se encuentra en la
posición" + mitad);
} else {
System.out.println("El número " + numero + " NO se encuentra");
}
}
```

Algoritmos de ordenación.

Burbuja

Consiste en recorrer el array comparando elementos adyacente

```
int[] vector = {14, 25, 3, 7, 52, 5, 68, 6, 10, 12, 15, 78, 1};
int temporal;
for (int i = 0; i < vector.length; i++) {
for (int j = 0; j < vector.length; j++) {
if (vector[i] < vector[j]) {
temporal = vector[i];
vector[i] = vector[j];
vector[j] = temporal;
          }
     }
}
```

Selección

```
int[] vector = {14, 25, 3, 7, 52, 5, 68, 6, 10, 12, 15, 78, 1};
//indice sirve para guardar el menor de la parte desordenadad, temporal
//para el intercambio
int indice, temporal;
//indica la parte ordenada
int indice_ordenado = 0;
for (int i = indice_ordenado; i < vector.length; i++) {
indice = i;
for (int j = indice_ordenado ; j < vector.length; j++) {
//si se encuentra uno menor se guarda para intercambiarlo
if (vector[indice] > vector[j]) {
indice = j;
}
}
//se realiza el intercambio si se ha encontrado uno ordenado después
if (indice_ordenado != indice) {
temporal = vector[indice_ordenado];
vector[indice_ordenado] = vector[indice];
vector[indice] = temporal;
}
//se incrementa la parte ordenada en 1
indice_ordenado++;
}
```

Inserción

```
int[] vector = {14, 5, 3, 7, 52, 500, 68, 6, 10, 12, 15, 78, 1};
int indice, temporal;
int indice_ordenado = 0;
//se hace hasta la longitud -1 ya que en el último no se inserta
ninguno
for (int i = indice_ordenado; i < vector.length-1; i++) {
//se toma el siguiente
indice = indice_ordenado+1 ;
//se va de indice mayor a indice menor mientras no se llegue a 0 o
//el valor del anterior sea menor
for (int j = indice_ordenado; j >= 0 && vector[indice] < vector[j];
j--) {
//se realiza el intercambio
temporal = vector[indice];
vector[indice] = vector[j];
vector[j] = temporal;
//se actualiza el indice del que se tiene que insertar
indice=j;
}
//se incrementa la parte ordenada en 1
indice_ordenado++;
}
```

CADENAS

```
string cadena= new string();
```

equals

```
char cadena.charAt(int indice)
```