

# CURSO DE PROGRAMADOR EN



## Relación de Ejercicios I

Documento realizado por:

José Antonio Márquez Paredes  
jamp@jamp.es  
<http://www.jamp.es>

Julio 2007

## INDICE

1.- Introducción al lenguaje Java	4
2.- Estructuras Condicionales e Iterativas	11
3.- Arrays Unidimensionales y Multidimensionales	14
4.- Métodos y Clases	17
5.- Diseño UML	22
6.- Herencia	24
7- Interfaces	29
8.- Excepciones	33
9.- Ficheros	34
10.- Interfaces Graficas	35
11.- Applets	37
12.- Sockets	39
13.- Proyecto	40

## Consideraciones Previas

Cada uno de los ejercicios de esta relación ha de desarrollarse en un archivo independiente cuyo nombre ha de ser de la forma "Ejercicio??.java", donde '??' es el número del ejercicio.

El desarrollo de los ejercicios ha de basarse en la siguiente estructura:

```
Ejercicio??.java
    class Ejercicio?? {
        static void main ( String [] args ) {
            <Secuencia de instrucciones>
        }
    }
```

donde **<Secuencia de instrucciones>** es la secuencia de instrucciones que implementa el programa descrito en el enunciado de cada ejercicio.

## 1.- INTRODUCCION AL LENGUAJE JAVA

### Ejercicio 1:

Desarrollar un programa que escriba en pantalla la frase "Mi nombre es xxx".

### Ejercicio 2:

Desarrollar un programa que escriba en pantalla varias líneas de texto, combinando los métodos System.out.print y System.out.println. La salida podría ser la siguiente:

```
Con diez cañones por banda,  
viento en popa, a toda vela,  
no corta el mar, sino vuela,  
un velero vergantín.
```

### Ejercicio 3:

Escribe el siguiente programa, compílalo y corrige los errores que salgan

```
public class JavaDivertido;{  
    public static void main ( String [] args ){  
        System.out.println ('Java es muy divertido')  
    }  
}
```

### Ejercicio 4:

Desarrollar un programa en el que se declare e inicialice una variable de cada uno de los tipos primitivos y escriba en pantalla frases del tipo:

**El valor de la variable '<tipo>' es: <???'>**

### Ejercicio 5:

Haz un programa que asigne 10 a una variable llamada número, la muestre por pantalla, asigne 7 a un variable llamada numero2, la muestre por pantalla y luego muestre la suma de ambas variables sin utilizar ninguna nueva variable

### Ejercicio 6:

A partir de una variable "num1" con el valor 12 y una variable "num2" con el valor 4, utiliza nuevas variables en las cuales se almacene la suma, resta, división y multiplicación de num1 y num2 y muéstralas en pantalla

### Ejercicio 7:

Escribe un programa que calcule la operación  $(7+14) / 7$ . Escríbela con y sin paréntesis y observa el resultado. ¿Qué conclusiones sacas?

### Ejercicio 8:

Escribe un programa que asigne 7 a una variable llamada numero, muéstrala por pantalla luego suma 1 a dicha variable, vuelve a mostrarla por pantalla, vuelve a restarle uno y vuelve a mostrar su valor por pantalla.

**Ejercicio 9:**

Realiza el ejercicio anterior pero con un sistema distinto de sumar y restar uno a la variable

**Ejercicio 10:**

Escribe un programa que calcule la operación  $2 + 9 - 6 / 3 * 2$ . Escribe paréntesis hasta en tres sitios distintos y observa y explica los resultados que se obtienen en todos los casos

**Ejercicio 11:**

Define dos variables primerNumero y segundoNumero y haz un pequeño programa que asigne un valor a cada una y obtenga el mayor de los dos, mostrándolo en un mensaje en pantalla. Para ello utiliza la construcción if preguntando por el valor de dichas variables

**Ejercicio 12:**

Intercambia el valor de dos variables llamadas "var1" y "var2" que tenga inicialmente (antes del intercambio) los valores 2 y 5 respectivamente. Muestra mensajes en pantalla con el valor que tiene cada variable antes y después del intercambio. Para realizar este intercambio de variables no se pueden hacer asignaciones directas de valores, si hace falta utiliza una nueva variable.

**Ejercicio 13:**

Realiza un programa que pida dos números enteros por pantalla y diga si el primero es divisible por el segundo. Para ello utilizar el operador % que devuelve el resto de la división.

**Ejercicio 14:**

Realiza un programa que lea una línea por teclado y la vuelva a escribir en la pantalla

**Ejercicio 15:**

Realiza un programa que lea un entero por teclado (es decir lo meta en una variable de tipo entero) y lo vuelva a escribir en pantalla

**Ejercicio 16:**

Realiza un programa que lleva a cabo las siguientes operaciones a partir de 3 variables enteras que lea por teclado, mostrando al final, por pantalla los resultados de las operaciones

```
(a + b) * c
(a * b) + c
(a / b) + c
(a + b) / c
(a * b) / c
(a / b) * c
```

**Ejercicio 17:**

Desarrollar un programa en el que se pida al usuario que introduzca un dato de tipo entero y otro de tipo real y, una vez hecho esto, se escriba en pantalla los valores introducidos, tal y como se indica en el ejemplo:

```
Introduzca un número entero: 7
Introduzca un número real: 3.45
El número entero proporcionado es 7
El número real proporcionado es 3.45
```

Nota: Para la realización de este ejercicio será necesario utilizar la librería Teclado.

**Ejercicio 18:**

¿Cuál es el resultado del siguiente programa?

```
class Ejemplo {
    public static void main(String [] args) {
        int a=1, b=4, c=2, d=1;
        int x=a+b/c+d;
        System.out.print("x =" + x);
    }
}
```

**Ejercicio 19:**

Suponga que b es una variable lógica (boolean). ¿Cuál es el resultado de las siguientes expresiones?

- a) b==true
- b) b=true

**Ejercicio 20:**

¿Cuál es el resultado del siguiente programa?

```
class Alcance {
    public static void main(String [] args) {
        int i=3;
        {
            int j=4;
        }
        System.out.println("j: "+j);
        System.out.println("i: "+i);
    }
}
```

**Ejercicio 21:**

Indique cuál es la salida del siguiente programa

```
class Ejercicio {
    public static void main(String[] args) {
        char probador;
        probador='c';
        System.out.println("probador:" + probador);
        ++probador;
        System.out.println("probador:"+probador);
        System.out.println("probador:"+ probador++ + probador+probador-- + probador);
    } //del main
} // de la clase
```

**Ejercicio 22:**

Proponga un programa que muestre un overflow con enteros

**Ejercicio 23:**

Indique cuál es la salida del siguiente programa

```
class Ejercicio {
    public static void main(String[] args) {
        int suma=30;
        System.out.println (suma++ + " " + ++suma + " " + suma + " " + suma--);
        System.out.println(suma);
    } //del main
} // de la clase
```

**Ejercicio 24:**

¿Cuál es el resultado del siguiente programa?

```
class Ejercicio {
    public static void main(String [] args) {
        int a=1, b=2, c=3, d=1;
        float r, s=(float)3.0;
        r=a+b/c+d/a;
        s=r-s;
        r=(long) s;
        r=++r;
        System.out.println(r);
    } //fin main
} // fin Ejercicio
```

**Ejercicio 25:**

¿Cuál es el resultado del siguiente programa?

```
class Ejercicio{
    public static void main(String [ ] args) {
        int var=1;
        boolean r,s,t,v;
        r=(var>1) && (var++ <100);
        s=(100 < var) && ( 150 > var++);
        t=(100 == var) || (200 > var++);
        v=(100 == var) || (200 > var++);
        System.out.println(r + " " + s + " " + t + " " + v);
    } //Fin del main
} //Fin de la clase
```

**Ejercicio 26:**

¿Cuál es el resultado de evaluar las siguientes expresiones si suponemos que, inicialmente, x vale 1?

- a. (x > 1) & (x++ < 10)
- b. (1 > x) && ( 1 > x++)
- c. (1 == x) | (10 > x++)
- d. (1 == x) || (10 > x++)
- e. (++x) + x;
- f. x + (++x)

**Ejercicio 27:**

Desarrollar un programa que, dadas las variables enteras a, b y c, cuyos valores iniciales se piden al usuario, imprima en pantalla el valor de las tres variables después de la ejecución de cada una de las siguientes expresiones:

`c += b, a -= b, b *= a -= c y c /= b.`

Por ejemplo:

	a	b	c
	10	5	-5
<code>c += b =&gt;</code>	10	5	0
<code>a -= b =&gt;</code>	5	5	0
<code>b *= a -= c =&gt;</code>	5	25	0
<code>c /= b =&gt;</code>	5	25	0

**Ejercicio 28:**

Desarrollar un programa que, dadas las variables enteras a y b, cuyos valores iniciales se piden al usuario, imprima en pantalla el valor de las tres variables después de la ejecución de cada una de las siguientes expresiones:

`c = ++a + ++b`  
`c = ++a + b++`  
`c = a++ + ++b`  
`c = a++ + b++.`

**Ejercicio 29:**

Desarrollar un programa que, dadas las variables enteras a y b, cuyos valores iniciales se piden al usuario, imprima en pantalla el valor de las tres variables después de la ejecución de cada una de las siguientes expresiones:

`c = --a + --b`  
`c = --a + b --`  
`c = a- - + -b`  
`c = a- - + b --`

**Ejercicio 30:**

Desarrollar un programa en el que se pidan al usuario dos datos de tipo real y escriba en pantalla su media.

**Ejercicio 31:**

Desarrollar un programa que sirva para comprobar que la tabla de los operadores lógicos vista en clase es correcta. Para ello se han de evaluar las operaciones lógicas sobre dos variables de tipo booleano x e y, a las que se les va asignando cada vez un valor lógico distinto.

**Ejercicio 32:**

Desarrollar un programa en el que se pidan al usuario dos datos de tipo entero y escriba en pantalla el cociente y el resto de la división entera entre ambos.

**Ejercicio 33:**

Desarrollar un programa en el que se pidan al usuario tres datos de tipo entero, a, b y c, y escriba en pantalla la ecuación de segundo grado.

**Ejercicio 34:**

Desarrollar un programa en el que se pida al usuario un valor real en pesetas y se escriba en pantalla el correspondiente valor en euros. (1 € = 166.386 pts).

**Ejercicio 35:**

Cuál es el resultado del siguiente programa?

```
class Ejercicio {
    public static void main(String [] args) {
        int s,x=0;
        switch (x) {
            case 0:
                s=0;
            default:
                if (x<0)
                    s=-1;
                else
                    s=1;
        }
        System.out.println(s);
    }
} //fin main
} //fin clase
```

**Ejercicio 36:**

Desarrollar un programa en el que se pida al usuario un valor real, x, y se escriba en pantalla su valor absoluto.

**Ejercicio 37:**

Desarrollar un programa en el que se pida al usuario un valor real, x, y se escriba en pantalla el valor de la función  $f(x) = 1/(x^2 - 1)$ . Para los valores de x en los que la función f(x) no esté correctamente definida, se debe escribir un aviso en pantalla.

**Ejercicio 38:**

Desarrollar un programa en el que se pidan al usuario dos valores reales, x e y, y se indique en pantalla cuál es el máximo y cuál es el mínimo. Si los dos valores son iguales, se ha de escribir en pantalla un mensaje indicándolo.

**Ejercicio 39:**

Desarrollar un programa en el que se pidan al usuario las coordenadas de un punto del plano, (x, y), e indique en pantalla si el punto se encuentra en el interior del círculo unidad (centro (0, 0) y radio 1), en el borde de dicho círculo o en el exterior del mismo.

**Ejercicio 40:**

Desarrollar un programa en el que se pidan al usuario tres números reales, y se escriban en pantalla ordenados de menor a mayor.

**Ejercicio 41:**

Una línea de autobuses cobra un mínimo de 20 euros por persona y trayecto. Si el trayecto es mayor de 200 km el billete tiene un recargo de 3 céntimos por km adicional. Sin embargo, para trayectos de más de 400 km el billete tiene un descuento del 15 %. Por otro lado, para grupos de 3 o más personas el billete tiene un descuento del 10 %. Con las consideraciones anteriores, escriba en Java un programa estructurado que lea por teclado la distancia del viaje a realizar, así como el número de personas que viajan juntas y que con ello calcule el precio del billete individual.

## 2.- ESTRUCTURAS CONDICIONALES E ITERATIVAS

### Ejercicio 1:

¿Cuántas veces se ejecutaría el cuerpo de los siguientes bucles for?

```
for (i=1; i<10; i++) ...
for (i=30; i>1; i-=2) ...
for (i=30; i<1; i+=2) ...
for (i=0; i<30; i+=4) ...
```

### Ejercicio 2:

Ejecute paso a paso el siguiente bucle:

```
c = 5;
for (a=1; a<5; a++)
    c = c - 1;
```

### Ejercicio 3:

El siguiente fragmento de programa pretende sumar los enteros de 1 a n (ambos inclusive) almacenando el resultado en la variable sum. ¿Es correcto el programa? Si no lo es, indique por qué y qué habría que hacer para solucionarlo.

```
i=0;
sum=0;
while (i<=n) {
    i=i+1;
    sum=sum+i;
}
```

### Ejercicio 4:

Reestructure el siguiente fragmento de código para evitar el uso de saltos incondicionales.

```
while (i < n) {
    j=Integer.parseInt(leer.readLine());
    if (j== -1) break;
    i++;
}
```

### Ejercicio 5:

Muestra los 10 primeros números naturales por pantalla mediante una estructura for

### Ejercicio 6:

Muestra los números impares que hay entre 1 y 10 por pantalla mediante una estructura for

### Ejercicio 7:

Muestra los 10 primeros números naturales impares por pantalla mediante una estructura for

**Ejercicio 8:**

Desarrollar un programa en el que, usando un bucle while, se escriba en pantalla una tabla de conversión de euros a pesetas para valores desde 1 hasta 10 (euros). (1 e = 166.386 pts).

**Ejercicio 9:**

Desarrollar un programa en el que, usando un bucle while, se escriban en pantalla los 51 primeros valores de la sucesión de Fibonacci, definida por recurrencia como sigue:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_{n+2} = f_{n+1} + f_n$$

**Ejercicio 10:**

Desarrollar un programa en el que se pida al usuario un valor entero positivo, n y, usando un bucle while, se escriba en pantalla el valor del factorial de n.

$$n! = n \times (n - 1) \times \dots \times 1.$$

**Ejercicio 11:**

Desarrollar un programa en el que se pida al usuario un valor entero positivo, n y, usando un bucle while, se escriba en pantalla el valor de la raíz cuadrada entera de n.

**Ejercicio 12:**

Desarrollar un programa en el que se pida al usuario un valor entero positivo, n y, usando un bucle while, se escriban en pantalla todos los múltiplos de 7 menores o iguales que el número n.

**Ejercicio 13:**

Desarrollar un programa en el que, usando un bucle for, se escriba en pantalla una tabla de conversión de grados Fahrenheit a Celsius para valores desde 0 hasta 300 a intervalos de 20. (0, 20, 40, ...). La regla de conversión es la siguiente:  $C = (5/9)(F - 32)$ .

**Ejercicio 14:**

Desarrollar un programa en el que se pida al usuario dos números enteros positivos, n y m, y, usando un bucle for, se escriba en pantalla el valor de n elevado a m.

**Ejercicio 15:**

Desarrollar un programa en el que, usando bucles for, se escriban en pantalla todos los pares [i, j] que representan una ficha de domino en la que  $0 \leq i \leq j \leq 6$ .

**Ejercicio 16:**

Escriba una función que, a partir de los dígitos de un ISBN, calcule el carácter de control con el que termina todo ISBN. Para calcular el carácter de control, debe multiplicar cada dígito por su posición (siendo el dígito de la izquierda el que ocupa la posición 1), sumar los resultados obtenidos y hallar el resto de dividir por 11. El resultado será el carácter de control, teniendo en cuenta que el carácter de control es 'X' cuando el resto vale 10.

### 3.- ARRAYS UNIDIMENSIONALES Y MULTIDIMENSIONALES

#### Ejercicio 1:

Desarrollar un programa que escriba en pantalla línea a línea todos los datos proporcionados en la entrada.

#### Ejercicio 2:

Desarrollar un programa que escriba en pantalla todos los datos proporcionados en la entrada en forma de vector, utilizando '[' y ']' como delimitadores y ',' como separador.

De esta forma:

```
> java Ejercicio??? 1 2 3 4 5  
[1,2,3,4,5]
```

#### Ejercicio 3:

Desarrollar un programa en el que se pida al usuario un vector de números enteros y obtenga el máximo de los valores incluidos en el vector.

#### Ejercicio 4:

Desarrollar un programa en el que se pida al usuario un vector de números enteros y obtenga el mínimo de los valores incluidos en el vector.

#### Ejercicio 5:

Desarrollar un programa en el que se pida al usuario un vector de números enteros e indique en pantalla si los elementos de dicho vector están ordenados de menor a mayor o no.

#### Ejercicio 6:

Desarrollar un programa en el que se pida al usuario un vector de números enteros e indique en pantalla la media de todos sus elementos.

#### Ejercicio 7:

Desarrollar un programa en el que se pidan al usuario un vector de números enteros e indique en pantalla si dicho vector es capicúa, es decir, la secuencia de sus elementos es igual vista de delante hacia atrás y de detrás hacia delante.

#### Ejercicio 8:

Desarrollar un programa en el que se pida al usuario un vector de números enteros e indique en pantalla cuantos de dichos elementos son números impares.

#### Ejercicio 9:

Desarrollar un programa en el que se pida al usuario un vector de números enteros e indique en pantalla el número de ocurrencias de elementos repetidos.

Por ejemplo el vector [1, 2, 3, 1, 2, 1] tiene tres ocurrencias de elementos repetidos (dos 1 y un 2).

**Ejercicio 10:**

Desarrollar un programa en el que se pidan al usuario dos vectores de números enteros  $v_1$  y  $v_2$ , se construya el vector resultado de “concatenar” los vectores  $v_1$  y  $v_2$ , es decir, poner los elementos de  $v_2$  a continuación de los de  $v_1$  y, finalmente, se escriban en pantalla todos los elementos de la concatenación.

**Ejercicio 11:**

Desarrollar un programa en el que se pida al usuario dos vectores de números enteros, se construya un nuevo vector  $v$  que almacene la suma de ambos vectores y, finalmente, se escriban en pantalla todos los elementos de  $v$ . El vector suma se ha de ajustar al vector más largo proporcionado por el usuario, completando el más corto con ceros.

Por ejemplo, la suma de los vectores  $[1, 2, 3]$  y  $[1, 2, 3, 4, 5]$  es  $[2, 4, 6, 4, 5]$ .

**Ejercicio 12:**

Desarrollar un programa en el que se construya una matriz de tamaño  $3 \times 3$  de números enteros a partir de los datos proporcionados por el usuario. Los datos de la matriz se pedirán con el procedimiento de lectura de vectores, uno por fila, desechando los elementos que se escriban de más y rellenando con ceros los que se escriban de menos. Una vez construida la matriz, el programa ha de escribir sus elementos en pantalla, una fila por línea.

**Ejercicio 13:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es nula. (Todos sus elementos iguales a cero)

**Ejercicio 14:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es positiva. (Todos sus elementos mayores o iguales a cero)

**Ejercicio 15:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es negativa. (Todos sus elementos menores o iguales a cero)

**Ejercicio 16:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es diagonal. (Todos los elementos que no están en la diagonal principal son nulos).

**Ejercicio 17:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es triangular superior. (Todos los elementos que están por debajo de la diagonal principal son nulos).

**Ejercicio 18:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es triangular inferior. (Todos los elementos que están por encima de la diagonal principal son nulos).

**Ejercicio 19:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es dispersa. (Todas las filas y todas las columnas contienen al menos un elemento nulo).

**Ejercicio 20:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y compruebe si la matriz es simétrica. (Los elementos de la matriz  $(i, j)$  y  $(j, i)$ , si existen, son iguales).

**Ejercicio 21:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y construya un vector con la suma de todas las filas de la matriz inicial.

**Ejercicio 22:**

Desarrollar un programa en el que se pida al usuario una matriz de dimensiones  $N \times M$ , y construya un vector con la suma de todas las columnas de la matriz inicial.

**Ejercicio 23:**

Desarrollar un programa en el que se pidan al usuario dos matrices de dimensiones  $N \times M$ , y construya una nueva matriz representando la suma de las matrices iniciales.

**Ejercicio 24:**

Desarrollar un programa en el que se pidan al usuario dos matrices de dimensiones  $N \times M$ , y construya una nueva matriz representando la resta de las matrices iniciales.

## 4.- METODOS Y CLASES

### Ejercicio 1:

¿Qué imprime el siguiente programa?

```
class Ejercicio {  
  
    public static void metodo1(int [] m1, int [] m2) {  
        m1[1]=4;  
        m2[1]=m1[3]+m2[2];  
    }//fin metodo1  
  
    public static void main(String [] args) {  
        int[] matriz1={9,1,3,5};  
        int [] matriz2=matriz1;  
        matriz2[3]=4;  
        matriz1[1]=3;  
        metodo1(matriz1, matriz2);  
        System.out.print(matriz1[1]+" "+matriz2[2]+" ");  
    }//fin main  
}//fin clase
```

### Ejercicio 2:

¿Qué imprimiría el siguiente programa?

```
class Ejemplo {  
    static void metodo1(int [] b, int c) {  
        c++;  
        for (int i=0; i<b.length; i++) {  
            b[i] = b[i] + c;  
        }  
  
    public static void main (String [] args) {  
        int [] a= {3,4,5};  
        metodo1(a, a[1]);  
        for (int i=0; i<a.length; i++)  
            System.out.print (a[i]+ " ");  
    }  
}
```

**Ejercicio 3:**

Desarrollar una clase Punto cuyos datos miembro sean las coordenadas de un punto del plano. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Punto que recibe como argumentos dos números reales, a y b y construye un nuevo objeto de la clase Punto cuyas coordenadas son a y b.
- Los métodos de acceso coordenadaX y coordenadaY, sin argumentos, que devuelven las coordenadas de un objeto Punto.
- Los métodos modificadores coordenadaX y coordenadaY, que reciben un argumento y modifican el valor de la correspondiente coordenada de un objeto Punto.
- El método igual, que comprueba si un objeto de la clase Punto es igual a otro dado que se pasa como argumento.
- El método distancia, sin argumentos, que calcula la distancia de un objeto de la clase Punto al origen de coordenadas.
- El método distancia, que calcula la distancia de un objeto de la clase Punto a otro que se proporciona como argumento.

**Ejercicio 4:**

Desarrollar una clase Vector para representar vectores. Los datos miembro de esta clase son las coordenadas del vector. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Vector que recibe como argumentos dos números reales, a y b y construye un nuevo objeto de la clase Vector cuyas coordenadas a y b.
- El constructor Vector que recibe como argumento un objeto de la clase Punto y construye un nuevo objeto de la clase Vector cuyas coordenadas coinciden con las del objeto de la clase Punto.
- El constructor Vector que recibe como argumentos dos objetos de la clase Punto, P<sub>1</sub> y P<sub>2</sub>, y construye un nuevo objeto de la clase Vector que representa el vector de origen P<sub>1</sub> y extremo P<sub>2</sub>.
- Los métodos de acceso extremoX y extremoY, sin argumentos, que devuelven las coordenadas de un objeto Vector.
- Los métodos modificadores extremoX y extremoY, que reciben un argumento y modifican el valor de la correspondiente coordenada de un objeto Vector.
- El método igual, que comprueba si un objeto de la clase Vector es igual a otro dado que se pasa como argumento.
- El método longitud, sin argumentos, que calcula la longitud de un objeto de la clase Vector.
- El método proporcional, que comprueba si un objeto de la clase Vector es proporcional a otro dado que se pasa como argumento.
- El método perpendicular, que comprueba si un objeto de la clase Vector es perpendicular a otro dado que se pasa como argumento.
- El método traslada, que recibe como argumento un objeto de la clase Punto, P, y devuelve el objeto Punto resultado de trasladar el punto P usando un objeto de la clase Vector.

**Ejercicio 5:**

Desarrollar una clase Recta para representar líneas rectas. Los datos miembro de esta clase son un objeto de la clase Punto perteneciente a la recta y un objeto de la clase Vector que representa la dirección de la recta. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Recta que recibe como argumentos un objeto de la clase Punto, P, y un objeto de la clase Vector, v, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por P con dirección v.
- Los métodos de acceso punto y vector, sin argumentos, que devuelven respectivamente los objetos Punto y Vector, datos miembro de un objeto Recta.
- Los métodos modificadores punto y vector, que respectivamente reciben como argumento un objeto de la clase Punto y un objeto de la clase Vector, y modifican el correspondiente dato miembro de un objeto Recta.
- El constructor Recta que recibe como argumento un objeto de la clase Vector, v, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por el origen de coordenadas con dirección v.
- El constructor Recta que recibe como argumentos dos objetos de la clase Punto, P<sub>1</sub> y P<sub>2</sub>, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por P<sub>1</sub> y P<sub>2</sub>.
- El método perpendicular, que comprueba si un objeto de la clase Recta es perpendicular a otro dado que se pasa como argumento.
- El método paralela, que comprueba si un objeto Recta es paralelo a otro que se pasa como argumento.
- El método pertenece, que recibe como argumento un objeto Punto P, y comprueba si P se encuentra en la recta representado por un objeto Recta.
- El método igual, que comprueba si un objeto de la clase Recta es igual a otro dado que se pasa como argumento.
- El método paralelaPunto, que recibe como argumento un objeto Punto P, y construye la representación de la recta paralela a un objeto Recta que pasa por el punto P.
- El método perpendicularPunto, que recibe como argumento un objeto Punto P, y construye la representación de la recta perpendicular a un objeto Recta que pasa por el punto P.

**Ejercicio 6:**

Desarrollar una clase Segmento para representar segmentos. Los datos miembro de esta clase son dos objetos de la clase Punto extremos de un segmento. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Segmento que recibe como argumentos dos objetos de la clase Punto, P<sub>1</sub> y P<sub>2</sub>, y construye un nuevo objeto de la clase Segmento cuyos extremos son P<sub>1</sub> y P<sub>2</sub>.
- Los métodos de acceso extremoA y extremoB, sin argumentos, que devuelven los extremos de un objeto Segmento.
- Los métodos modificadores extremoA y extremoB, que reciben como argumento un objeto de la clase Punto y modifican el correspondiente extremo de un objeto Segmento.
- El método igual, que comprueba si un objeto de la clase Segmento es igual a otro dado que se pasa como argumento.
- El método longitud, sin argumentos, que calcula la longitud de un objeto de la clase Segmento.
- El método proporcional, que comprueba si un objeto de la clase Segmento es proporcional a otro dado que se pasa como argumento.
- El método perpendicular, que comprueba si un objeto de la clase Segmento es perpendicular a otro dado que se pasa como argumento.
- El método puntoMedio, sin argumentos, que devuelve el objeto Punto que representa el punto medio de un objeto Segmento.
- El método pertenece, que recibe como argumento un objeto Punto P, y comprueba si P se encuentra en el segmento representado por un objeto Segmento.
- El método recta, sin argumentos, que construye un objeto Recta que contiene a un objeto Segmento.
- El método mediatriz, sin argumentos, que devuelve un objeto Recta que representa la mediatriz de un objeto Segmento.

**Ejercicio 7:**

Diseñe una clase Cuenta que represente una cuenta bancaria y permita realizar operaciones como ingresar y retirar una cantidad de dinero, así como realizar una transferencia de una cuenta a otra

Se pide:

- a. Represente gráficamente la clase utilizando la notación UML
- b. Defina la clase utilizando la sintaxis de Java, definiendo las variables de instancia y métodos que crea necesarios.
- c. Implemente cada uno de los métodos de la clase. Los métodos deben actualizar el estado de las variables de instancia y mostrar un mensaje en el que se indique que la operación se ha realizado con éxito.
- d. Cree un programa en Java (en una clase llamada CuentaTest) que cree un par de objetos de tipo Cuenta y realice operaciones con ellos. El programa debe comprobar que todos los métodos de la clase Cuenta funcionan correctamente.

**Ejercicio 8:**

Construir un módulo para tratamiento de una máquina de tickets en un supermercado. Un cliente del supermercado, para realizar una compra, coge un ticket que le indica su turno.

Para ello hace uso de un máquina de tickets que permite:

Operaciones:

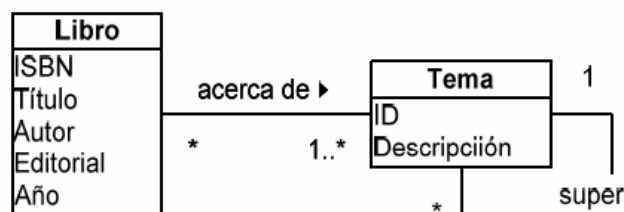
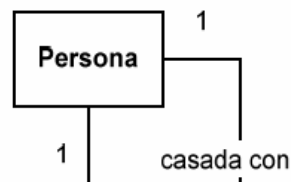
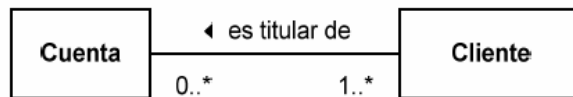
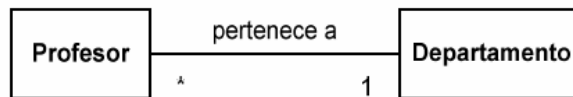
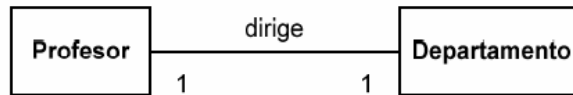
- Inicializar la cuenta de los tickets desde cero.
- Obtener un nuevo ticket (y actualizar para que la siguiente persona reciba el número siguiente al ultimo expedido).

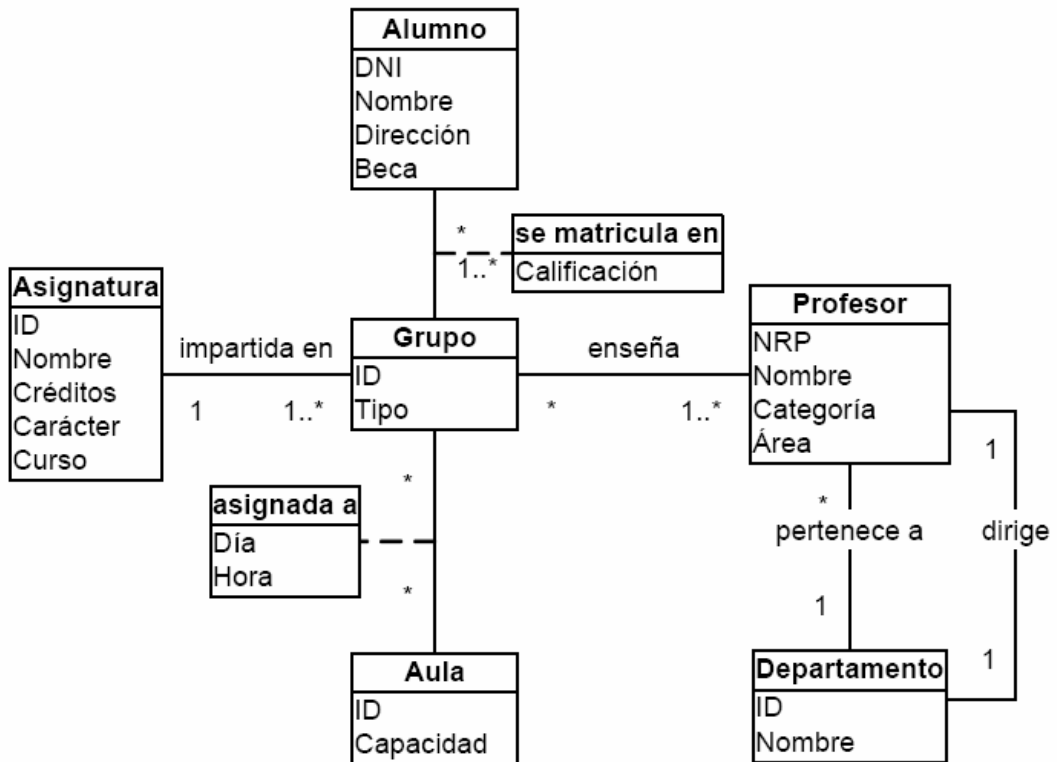
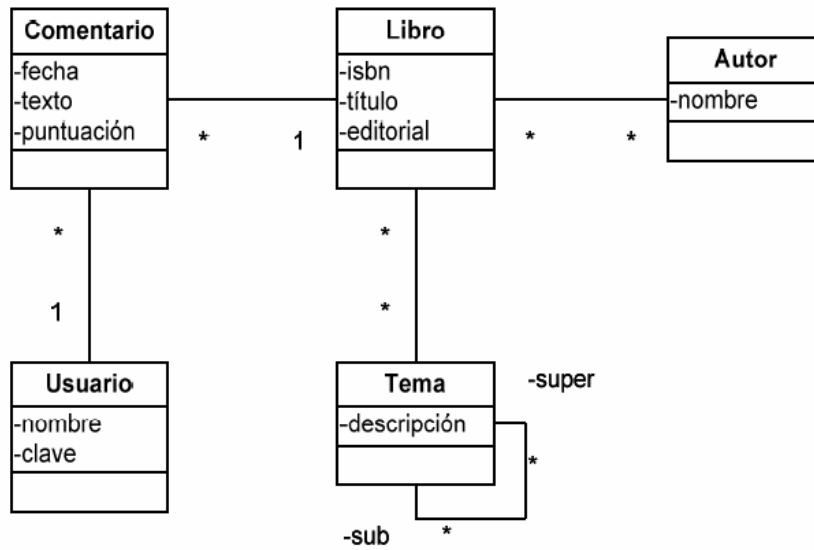
- a) Implementa dicho(s) módulo(s) como una clase(s) Java.
- b) Implementa un programa de prueba.

## 5.- DISEÑO UML

### Ejercicio 1:

Definir adecuadamente las clases en Java que se derivan de los siguientes diagramas de clases UML:





## 6.- HERENCIA

### Ejercicio 1:

¿Cuál es el resultado del siguiente programa?

```
class Uno {
    protected int i=2;
    public void frase() {
        System.out.println("Estoy en un objeto de clase Uno");
    }
}

class Dos extends Uno {
    public void frase() {
        int i=3;
        System.out.println("Estoy en un objeto de clase Dos con i:"+i);
    }
}

class Tres extends Dos {
    public void frase() {
        System.out.println("Estoy en un objeto de clase Tres con i:"+i);
    }
}

class Driver {
    public static void main(String[] args) {
        Uno [] lista =new Uno [2];
        lista [0]= new Dos();
        lista [1]= new Tres();
        for (int i=0; i<2; i++){
            lista[i].frase();
        }
    }
}
//fin main
//fin clase Driver
```

**Ejercicio 2:**

¿Cuál es el resultado del siguiente programa?

```
class Padre {
    protected int aa=0;
    public int aa() {
        return aa;
    }
}

class Hija extends Padre {
    public Hija (int bb) {
        this.aa=bb+1;
    }
}

class Nieta extends Hija {
    public Nieta (int cc) {
        super(cc+2);
    }
}

class Familia {
    private static Nieta f (Padre h) {
        Nieta n=new Nieta (h.aa());
        return n;
    }

    public static void main(String [] args){
        Hija h= new Hija(4);
        h=f(h);
        System.out.println (h.aa());
    }
}
```

**Ejercicio 3:**

En una tienda se venden 2 tipos de ordenadores: portátiles y de sobremesa. Ambos tipos de ordenadores se caracterizan por su código y por su precio. Además cada uno tiene un eslogan que es: "Ideal para sus viajes" en el caso de los portátiles y "Es el que más pesa, pero el que menos cuesta" para el caso de los ordenadores de sobremesa. Además los ordenadores portátiles tienen un atributo peso, y los de sobremesa la descripción del tipo de torre.

- Realice el diseño en UML indicando las clases que hay que tener en cuenta y su relación.
- Implemente en Java dichas clases.

**Ejercicio 4:**

Desarrollar una clase Empresa cuyos datos miembro sean un nombre, un tamaño y un array de empleados personal (la clase Empleado se pide en el siguiente ejercicio). El tamaño de la empresa es inmutable. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Empresa que recibe como argumentos una cadena de texto nombre y un valor entero tamaño, y construye un nuevo objeto de la clase Empresa cuyo nombre y tamaño vienen dados respectivamente por los argumentos nombre y tamaño. El tamaño del array de empleados personal viene dado por el valor de la variable tamaño.
- Los métodos de acceso getNombre y getTamaño, sin argumentos, que respectivamente devuelven el nombre y el tamaño de un objeto Empresa.
- El método de acceso getEmpleado, que recibe como argumento un número entero menor que el tamaño de la empresa, y devuelve el correspondiente campo del array de empleados.
- El método despideEmpleado, que recibe como argumento un número entero menor que el tamaño de la empresa, y asigna la referencia null al correspondiente campo del array de empleados.

**Ejercicio 5:**

Desarrollar una clase Empleado cuyos datos miembro sean una empresa, un nombre, un sueldo y un número de empleado (numEmpleado). Estos datos han de ser protegidos (protected). Además, el número de empleado y la empresa son inmutables. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Empleado que recibe como argumentos emp, una referencia a un objeto Empresa, una cadena de texto nombre y un valor entero sueldo, y construye un nuevo objeto de la clase Empleado en el que la empresa, el nombre y el sueldo vienen dados respectivamente por los argumentos emp, nombre y sueldo. El número de empleado se crea de manera única por cada empleado, usando para ello una variable contador perteneciente a la clase (static y private).
- El constructor protegido Empleado que recibe como argumentos emp, una referencia a un objeto Empresa, una cadena de texto nombre, un valor entero sueldo y un número de empleado numero, y construye un nuevo objeto de la clase Empleado en el que la empresa, el nombre, el sueldo y el número de empleado vienen dados respectivamente por los argumentos emp, nombre, sueldo y numero.
- Los métodos de acceso getNombre, getSueldo y getNumeroEmpleado, sin argumentos, que respectivamente devuelven los campos nombre, sueldo y numEmpleado de un objeto Empleado.
- El método modificador setNombre, que recibe como argumento una cadena y modifica el valor del campo nombre de un objeto Empleado.
- El método modificador setSueldo, que recibe como argumento un valor entero y modifica el valor del campo sueldo de un objeto Empleado.
- El método de impresión en pantalla toString, sin argumentos, que devuelve una cadena (String) con el nombre, número y sueldo de un objeto Empleado.
- El método aumentarSueldo, que recibe como argumento un número entero N y modifica el sueldo del objeto Empleado sobre el que se evalúa, aumentándolo un N%. Este método no puede ser modificado por clases derivadas.
- El método despedir, sin argumentos, que despide al objeto Empleado sobre el que se evalúa.

**Ejercicio 6:**

Añadir a la clase Empresa el método nuevoEmpleado, que recibe como argumentos una cadena de texto nombre y un valor entero sueldo, crea un nuevo empleado asignado a la empresa sobre la que se evalúa el método, cuyo nombre y sueldo vienen dados por los argumentos nombre y sueldo y, finalmente, utiliza el número de empleado como índice para almacenar una referencia al objeto Empleado recién creado en el array de empleados de la empresa.

**Ejercicio 7:**

Desarrollar una clase Ejecutivo derivada de la clase Empleado anterior, con un campo entero adicional presupuesto. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Ejecutivo que recibe como argumentos emp, una referencia a un objeto Empresa, una cadena de texto nombre y un valor entero sueldo, y construye un nuevo objeto de la clase Ejecutivo utilizando el constructor de la clase base (super(...)).
- El constructor Ejecutivo que recibe como argumentos emp, una referencia a un objeto Empresa, una cadena de texto nombre, un valor entero sueldo y un número de empleado numero, y construye un nuevo objeto de la clase Ejecutivo utilizando el constructor de la clase base (super(...)).
- El método de acceso getPresupuesto, sin argumentos, que devuelve el valor del campo presupuesto de un objeto Ejecutivo.
- El método modificador asignaPresupuesto, que recibe como argumento un valor entero y modifica el valor del campo presupuesto de un objeto Ejecutivo.
- Redefinir el método de impresión en pantalla toString para que indique que el objeto sobre el que se evalúa es un ejecutivo.

**Ejercicio 8:**

Añadir a la clase Empleado el método ascender sin argumentos, que crea un nuevo objeto Ejecutivo con los datos del objeto Empleado sobre el que se evalúa el método y cambia la referencia en el array de personal de la empresa a la que pertenece dicho objeto.

**Ejercicio 9:**

Desarrollar una clase Producto cuyos datos miembro sean una cadena de texto identificación y un valor real (double) precioBase. Ambos datos de tipo protegido. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Producto que recibe como argumentos una cadena de texto identificación y un valor real (double) precioBase, y construye un nuevo objeto de la clase Producto cuya identificación y precio base vienen dados respectivamente por los argumentos identificación y precioBase.
- Los métodos de acceso getIdentificacion y getPrecioBase, sin argumentos, que respectivamente devuelven el identificador y el precioBase de un objeto Producto.
- El método modificador setIdentificacion, que recibe como argumento una cadena de texto identificación y cambia el campo identificación del objeto sobre el que se aplica, asignándole como nuevo valor el del argumento identificación.
- El método modificador setPrecioBase, que recibe como argumento un valor real (double) precioBase y cambia el campo precioBase del objeto sobre el que se aplica, asignándole como nuevo valor el del argumento precioBase.
- El método de impresión en pantalla toString, sin argumentos, tal que al ser evaluado sobre un objeto de tipo Producto cuya identificación es "RJ45" y cuyo precio base es 10.50, genere la siguiente cadena:

RJ45 (10.5)

**Ejercicio 10:**

Desarrollar una clase `ProductoInventariado` derivada de la clase `Producto` anterior, con dos campos enteros adicionales `cantidad` y `beneficio`. Ambos de tipo protegido. Para esta clase se piden los siguientes constructores y métodos:

- El constructor `ProductoInventariado`, que recibe como argumentos una cadena de texto identificación, un valor real (`double`) `precioBase` y dos datos enteros `cantidad` y `beneficio`, y construye un nuevo objeto de la clase `Producto` utilizando el constructor de la clase base (`super(...)`), cuya identificación, precio base, cantidad y beneficio vienen dados respectivamente por los argumentos identificación, `precioBase`, `cantidad` y `beneficio`.
- Los métodos de acceso `getCantidad` y `getBeneficio`, sin argumentos, que respectivamente devuelven la cantidad y el beneficio de un objeto `ProductoInventariado`.
- Los métodos modificadores `setCantidad` y `setBeneficio`, que reciben como argumento un valor entero y modifican, respectivamente, el valor del campo `cantidad` y `beneficio` del objeto sobre el que se aplican, asignándoles como nuevo valor el del argumento recibido.
- El método `precioFinal`, sin argumentos, que devuelve el precio final de un objeto de la clase `ProductoInventariado` determinado como el precio base al que se suma el porcentaje de beneficio.
- El método de impresión en pantalla `toString` sin argumentos, tal que al ser evaluado sobre un objeto de tipo `ProductoInventariado` cuya identificación es "RJ45", cuyo precio base es 10.50, cuya cantidad es 13 y cuyo beneficio es 10, genera la siguiente cadena: 10 RJ45 (10.5) (+13%)

**Ejercicio 11:**

Desarrollar una clase `Tienda` cuyos datos miembro son una cadena de texto `nombre`, un valor entero inmutable `maxProducto` que indica el número máximo de productos distintos que puede tener la tienda, `inventario` un array de objetos del tipo `ProductoInventariado` donde se almacena información de los productos de la tienda, un valor entero `ultimaEntrada` que indica la primera posición libre en el array `inventario` y un valor real (`double`) `caja` que almacena el dinero del que dispone la tienda. Para esta clase se piden los siguientes constructores y métodos:

- El constructor `Tienda`, que recibe como argumentos una cadena de texto `nombre`, un valor entero `maxProducto` y un valor real `caja`, y construye un nuevo objeto de la clase `Tienda` cuyo `nombre`, `maxProducto` y `caja` vienen dados respectivamente por los argumentos `nombre`, `maxProducto` y `caja`. El tamaño del array `inventario` viene dado por el valor del argumento `maxProducto` y el valor de `ultimaEntrada` es 0.3
- El método `buscaProducto`, que recibe como argumento una cadena de texto `id`, y devuelve el índice del array `inventario` en el que se encuentra un producto cuyo identificador coincide con `id`, si es que existe, o el valor de `ultimaEntrada` en caso contrario.
- El método `añadirProducto`, que recibe como argumento un identificador de producto `id`, un precio base `p` (tipo `double`), una cantidad `c` (tipo `int`) y un beneficio `b` (tipo `int`) y lo añade al `inventario`. Si el producto ya estaba en el `inventario` entonces solo hay que modificar los datos relativos al precio base, cantidad y beneficio. Si el producto no está en el `inventario` entonces hay que añadirlo. En cualquier caso, solo se podrá añadir un producto si el coste total ( $\text{cantidad} \times \text{precio base}$ ) es menor o igual que el dinero del que dispone la tienda, el cual ha de ser disminuido de manera adecuada. Si en el `inventario` no hay sitio para el producto o éste no puede ser adquirido por no disponer de suficiente dinero entonces se ha de indicar en pantalla un mensaje informativo.
- El método `venderProducto`, que recibe como argumento un identificador de producto `id` y una cantidad `c` y, si el producto existe en el `inventario` en una cantidad mayor o igual que `c`, entonces disminuye en `c` unidades la cantidad del producto `id` que hay en el `inventario` y modifica adecuadamente la `caja`. Si la tienda se queda sin unidades del producto `id` entonces hay que modificar adecuadamente el array `inventario` y el valor de `ultimaEntrada` para evitar "huecos vacíos". Si no hay unidades suficientes del producto `id` para vender, entonces se ha de indicar en pantalla un mensaje informativo.

## 7.- INTERFACES

### Ejercicio 1:

Desarrollar una interfaz Vehiculo que declare los métodos factura, hayPlaza, aparca y setTiempo, tales que: factura (sin argumentos) proporciona el importe a pagar por estacionar un Vehiculo durante determinado tiempo en un parking. hayPlaza recibe como argumento una referencia a un objeto Parking y determina si hay sitio en dicho Parking para estacionar un Vehiculo. aparca recibe como argumento una referencia a un objeto Parking y estaciona un Vehiculo en dicho Parking. setTiempo recibe como argumento un entero y establece ese entero como tiempo de estancia del vehiculo en el parking.

### Ejercicio 2:

Implemente el código de una interfaz llamada Primera que contenga dos métodos A y B. Defina otra interfaz llamada Segunda que herede de la anterior y además contenga un método llamado C. Escriba el código de otra clase llamada Objetos que use la segunda interfaz. ¿Cuántos métodos debe implementar esta clase? Implemente dichos métodos de forma que cada método imprima una línea indicando el nombre del método. Cree un programa que utilice los métodos definidos.

### Ejercicio 3:

¿Cual es el error del siguiente programa?

```
interface A {
    double a=2.3;
    void imprimirresultadoA () {
        System.out.println ("valor de a"+ a);
    }
}

interface B {
    int b=435;
    void imprimirresultadoB ();
}

class AA implements A, B {
    double aa=324.32;
    public void imprimirresultadoA () {
        System.out.println ("valor de a"+ a+ "valor de aa"+ aa);
    }

    public void imprimirresultadoB () {
        System.out.println ("valor de b"+ b+ "valor de aa"+ aa);
    }
}

class Principal {
    public static void main (String [] args) {
        AA ob1=new AA();
        ob1.imprimirresultadoA();
        ob1.imprimirresultadoB();
    }
}
```

**Ejercicio 4:**

Desarrollar una clase Parking, cuyos datos miembro sean un nombre, un entero plazasTotal, indicando el numero total de plazas, un entero plazasOcupadas, indicando el numero total de plazas ocupadas, un entero plazasAbonados, indicando el número total de plazas reservadas para clientes abonados, un entero plazasHotel, indicando el número total de plazas reservadas para clientes del hotel y un array plazas de Vehiculos. El campo plazasTotal es inmutable. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Parking que recibe como argumentos una cadena de texto nombre y un valor entero plazasTotal, y construye un nuevo objeto de la clase Parking cuyo nombre y número total de plazas vienen dados respectivamente por los argumentos nombre y plazasTotal. El tamaño del array de vehiculos plazas viene dado por el valor de la variable plazasTotal.
- Los métodos de acceso getNombre, getPlazasTotal, getPlazasOcupadas, getPlazasAbonados, getPlazasHotel y getPlazasClientes, sin argumentos, que devuelven, respectivamente, el nombre, el número total de plazas, el número de plazas ocupadas, el número de plazas reservadas a clientes abonados, el número de plazas reservadas a clientes del hotel y el número de plazas restantes dedicadas a clientes externos.
- El método de acceso getVehiculo, que recibe como argumento un número entero, y, si dicho número es menor que el número total de plazas del parking entonces devuelve el correspondiente campo del array de plazas. En caso contrario devuelve una referencia vacía (null).
- El método de modificación setPlazasAbonados, que reciben como argumento un número entero y, si se dispone de dicho número de plazas en el parking, se almacena dicho valor en el campo plazasAbonados. Hay que tener en cuenta las plazas que ya están reservadas para los vehículos del hotel.
- El método de modificación setPlazasHotel, que reciben como argumento un número entero y, si se dispone de dicho número de plazas en el parking, se almacena dicho valor en el campo plazasHotel. Hay que tener en cuenta las plazas que ya están reservadas para los vehículos de clientes abonados.
- El método numAbonados, sin argumentos, que devuelve el número de plazas del parking ocupadas por vehículos de abonados. La clase que almacena información sobre los vehículos de los abonados es VehiculoAbonado.
- El método numHotel, sin argumentos, que devuelve el número de plazas del parking ocupadas por vehículos del hotel. La clase que almacena información sobre los vehículos del hotel es VehiculoHotel.
- El método numClientes, sin argumentos, que devuelve el numero de plazas del parking ocupadas por vehículos del hotel. La clase que almacena información sobre los vehículos del hotel es VehiculoCliente.
- El método de modificación setVehiculo, que recibe como argumento una referencia a un objeto de tipo Vehiculo y, si hay sitio en el parking, sitúa dicha referencia en una plaza libre. En caso de no existir plazas libres se tiene que presentar un mensaje informativo en pantalla.

**Ejercicio 5:**

Desarrollar una clase VehiculoCliente implementando la interfaz Vehiculo, cuyos datos miembro son una cadena de texto id y un valor entero tiempo. La cadena de texto id funciona como identificación del vehículo y no puede ser modificada. El valor entero tiempo almacena el número de horas que el vehículo está en el parking. Para esta clase se piden los siguientes constructores y métodos:

- El constructor VehiculoCliente, que recibe como argumento un identificador de vehículo, y construye un nuevo objeto del tipo VehiculoCliente. El valor inicial del campo tiempo es 0.
- El método setTiempo, que recibe como argumento un número entero y lo almacena como valor del campo tiempo del objeto VehiculoCliente.
- El método factura, sin argumentos, que determina lo que tiene que pagar un objeto del tipo VehiculoCliente por su estancia en el parking. El coste de la estancia es de 12€ el día completo y 0.6€ la hora o fracción.
- El método hayPlaza, que recibe como argumento una referencia a un objeto Parking y determina si hay una plaza en dicho Parking para estacionar un objeto del tipo VehiculoCliente.
- El método aparca, que recibe como argumento una referencia a un objeto Parking y estaciona un VehiculoCliente en dicho Parking.

**Ejercicio 6:**

Desarrollar una clase VehiculoAbonado implementando la interfaz Vehiculo, cuyos datos miembro son una cadena de texto id y un valor entero tiempo. La cadena de texto id funciona como identificación del vehículo y no podrá ser modificada. El valor entero tiempo almacena el número de meses para los que el vehículo tiene contratado el parking. Para esta clase se piden los siguientes constructores y métodos:

- El constructor VehiculoAbonado, que recibe como argumento un identificador de vehículo y un número entero tiempo indicando el número de meses para los que se ha contratado el parking, y construye un nuevo objeto del tipo VehiculoAbonado.
- El método factura, sin argumentos, que determina lo que tiene que pagar un objeto del tipo VehiculoAbonado por su estancia en el parking. El coste de la estancia es de 200€ el mes.
- El método hayPlaza, que recibe como argumento una referencia a un objeto Parking y determina si hay una plaza en dicho Parking para estacionar un objeto del tipo VehiculoAbonado.
- El método aparca, que recibe como argumento una referencia a un objeto Parking y estaciona un VehiculoAbonado en dicho Parking.

**Ejercicio 7:**

Desarrollar una clase VehiculoHotel implementando la interfaz Vehiculo, cuyos datos miembro son una cadena de texto id y un valor entero tiempo. La cadena de texto id funciona como identificación del vehículo y es inmutable. El valor entero tiempo almacena el número de días para los que el vehículo tiene concertado el parking. Para esta clase se piden los siguientes constructores y métodos:

- El constructor VehiculoHotel, que recibe como argumento un identificador de vehículo y un número entero tiempo indicando el número de días para los que se ha concertado el parking, y construye un nuevo objeto del tipo VehiculoHotel.
- El método factura, sin argumentos, que determina lo que tiene que pagar un objeto del tipo VehiculoHotel por su estancia en el parking. El coste de la estancia es de 10€ el día.
- El método hayPlaza, que recibe como argumento una referencia a un objeto Parking y determina si hay una plaza en dicho Parking para estacionar un objeto del tipo VehiculoHotel.
- El método aparca, que recibe como argumento una referencia a un objeto Parking y estaciona un VehiculoHotel en dicho Parking.

**Ejercicio 8:**

Una centralita gestiona las llamadas telefónicas de un conjunto de clientes. Cada cliente está identificado de manera única en la centralita y dispone de un saldo. Para cada llamada se guarda el número de teléfono al que se llama y la duración de la misma. La centralita guarda un registro de las últimas llamadas realizadas por los clientes.

Las llamadas pueden ser locales o provinciales. El coste de una llamada local es de 0.15e el segundo. El coste de una llamada provincial depende de la franja horaria en la que se realiza dicha llamada: 0.20e en franja 1, 0.25e en franja 2 y 0.30e en franja 3.

- Implementar la interfaz Llamada y las clases LlamadaLocal y LlamadaProvincial, para almacenar la información relativa a las llamadas. En estas clases se ha de poder calcular el coste de la llamada.
- Implementar la clase Cliente, para almacenar la información relativa a los clientes de la centralita.
- Implementar la clase Registro, para almacenar la información de cada uno de los registros que almacena la centralita.
- Implementar la clase Centralita de forma que almacene información sobre los clientes y el registro de llamadas de estos. Las operaciones que se han de poder hacer en la centralita son las siguientes: Dar de alta a un cliente. Dar de baja a un cliente. Incrementar el saldo de un cliente (o darlo de alta si es que no existe). Añadir un nuevo registro al registro de llamadas.

Presentar en pantalla la lista de los últimos N registros ( $0 \leq N \leq 100$ ).

## 8.- EXCEPCIONES

### Ejercicio 1:

Construir un módulo para tratamiento de Fracciones que realice las siguientes operaciones:

- 1.1 Constructor
- 1.2 Acceso al numerador y al denominador
- 1.3 Operaciones aritméticas entre fracciones, tales como: suma, incremento de una fracción en otra, resta, decremento de una fracción en otra, producto, cociente y simplificar una fracción.
- 1.4 Operaciones de comparación de fracciones, tales como: igualdad, mayor, menor, mayor o igual y menor o igual.

Identifica los casos excepcionales en las operaciones y define las excepciones correspondientes

### Ejercicio 2:

Construir un módulo para tratar la abstracción matemática Conjunto, que maneje conjuntos de números naturales del 1 al 100.

Características:

- Descripción por enumeración, por ejemplo {1,4,6}
- No hay orden asociado.
- No hay elementos repetidos.
- No tiene sentido manipular los componentes individuales.

Operaciones:

- 1.1 Crear el conjunto vacío.
- 1.2 Insertar un elemento.
- 1.3 Borrar un elemento.
- 1.4 Igualdad entre conjuntos
- 1.5 Comprobar si el conjunto está vacío, si un elemento pertenece a un conjunto y si dos conjuntos con iguales.
- 1.6 Unión de conjuntos
- 1.7 Intersección de conjuntos
- 1.8 Resta de conjuntos
- 1.9 Complementario de un conjunto
- 1.10 Lectura y escritura de conjuntos en la entrada/salida estándar.
- 1.11 Identifica los casos excepcionales en las operaciones y define las excepciones correspondientes
  - a) Dibuja en UML la clase que implementa la anterior interfaz.
  - b) Implementa la interfaz y la clase anteriores en Java.
  - c) Implementa un programa de prueba.

## 9.- FICHEROS

### Ejercicio 1:

A fin de ocultar los detalles de implementación, diseñe y programe una clase para lectura y escritura de enteros, dobles y cadenas en un fichero secuencial de datos pasados a byte.

Construya un programa principal que use dicha clase. Para ello el programa escribirá en un fichero algunos datos de un cierto número de clientes. Los datos serán número, nombre y saldo del cliente. El nombre del fichero, el número de clientes y los datos de cada cliente se leerán por teclado. Los datos se leerán posteriormente del fichero y se imprimirán en pantalla.

### Ejercicio 2:

A fin de ocultar los detalles de implementación, diseñe y programe una clase para lectura y escritura de cadenas en ficheros secuenciales de texto.

Construya un programa principal que use dicha clase.

### Ejercicio 3:

A fin de ocultar los detalles de implementación, diseñe y programe una clase para lectura y escritura de objetos en ficheros.

Construya un programa principal que use dicha clase.

## 10.- INTERFACES GRAFICAS

### Ejercicio 1:

Realizar una interfaz grafica formada por un único botón

### Ejercicio 2:

Ampliar la interfaz grafica del ejercicio anterior, de forma que al pulsar el botón se muestre por la salida estándar, nuestro nombre.

### Ejercicio 3:

Realizar una interfaz formada por un cuadro de texto y un botón

### Ejercicio 4:

Ampliar la interfaz grafica del ejercicio anterior, de forma que al pulsar el botón se muestre por la salida estándar, el texto indicado en el cuadro de texto.

### Ejercicio 5:

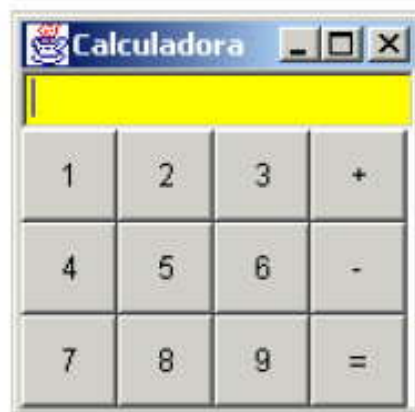
Realizar una interfaz formada por un cuadro de texto y dos botones.

### Ejercicio 6:

Ampliar la interfaz grafica del ejercicio anterior, de forma que al pulsar uno de los botones se muestre por la salida estándar, el texto indicado en el cuadro de texto. Al pulsar el segundo de los botones se deberá borrar el contenido del cuadro de texto.

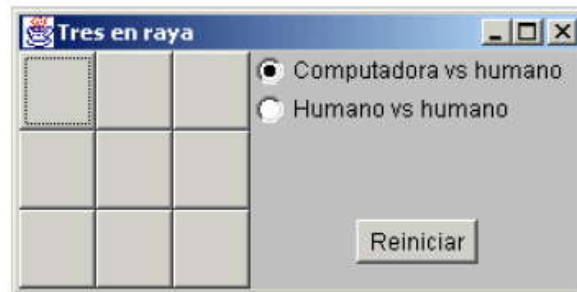
### Ejercicio 7:

Desarrollar el interfaz visual de una calculadora. El resultado debe ser algo parecido a:



**Ejercicio 8:**

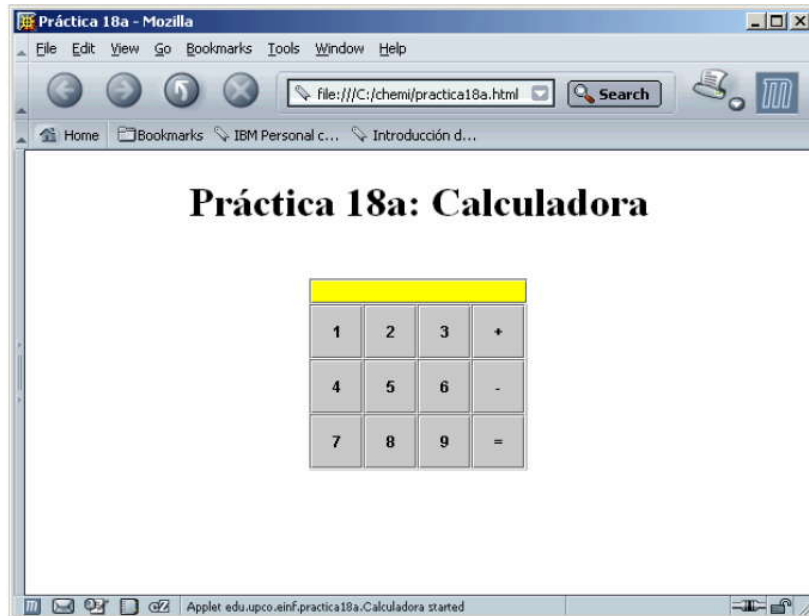
Desarrollar el interfaz visual de un juego de la tres en raya. El resultado debe ser algo parecido a:



## 11.- APPLET'S

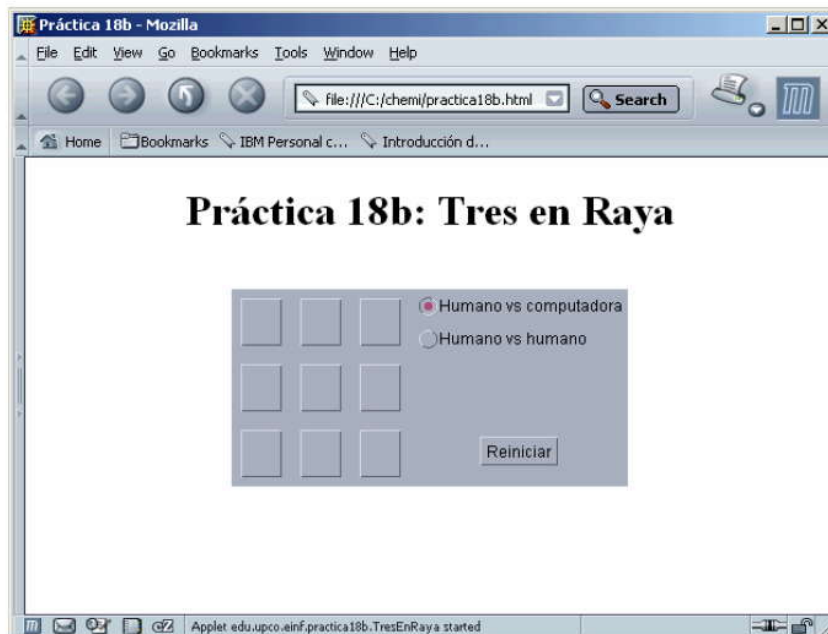
### Ejercicio 1:

Convertir la calculadora de la practica de interfaces graficas en un applet. El resultado debe ser algo parecido a:



### Ejercicio 2:

Convertir el juego de las tres en raya de la practica de interfaces graficas en un applet. El resultado debe ser algo parecido a:



**Ejercicio 3:**

Desarrollar un applet que convierta de euros a pesetas. El resultado debe ser algo parecido a:



## 12.- SOCKETS

### Ejercicio 1:

Realizar un programa que, tomando a partir del nombre de una maquina, escriba por pantalla su dirección IP en notación de punto.

### Ejercicio 2:

Realizar un programa que solicite al usuario una cadena de caracteres, establezca una conexión con otro ordenador y envíe a este la cadena leída.

### Ejercicio 3:

Realizar un programa que envíe un entero a otra maquina, para que cuando la maquina destino lo reciba, sume uno a dicho numero y lo imprima por pantalla.

### Ejercicio 4:

Realizar un programa que solicite al usuario un numero, a continuación envíe el numero leído a otra maquina, para que cuando la maquina destino lo reciba, sume un cierto valor a dicho número. Una vez realizada la suma, la maquina destino devolverá a la maquina origen el resultado de la operación, para que está intente adivinar que numero fue sumado.

### Ejercicio 5:

Realizar un programa que simule un Chat en modo texto.

### Ejercicio 6:

Realizar un programa que simule un Chat en modo grafico.

### Ejercicio 7:

Realizar un programa que envíe un fichero a un destino.

**CASO PRACTICO**

Un centro de formación nos ha contratado para realizar e implementar un software de Gestión Académica.

Dicho software debe realizar una administración de:

- Alumnos
- Profesores
- Empleados
- Aulas
- Equipos Informáticos
- Cursos
- Matriculas

Una vez realizada la primera entrevista con el personal del centro para comentar cada uno de los puntos a tratar nos han indicado lo siguiente:

**Alumnos:**

Cada alumno vendrá al centro para matricularse posteriormente a uno de nuestros cursos. De los alumnos nos interesa tener todos los datos posibles, como pueden ser Nombre, Dirección, DNI, Fecha de Nacimiento, Estudios, Si desea formar parte de nuestra bolsa de empleo...

**Profesores:**

Se dispondrá de una bolsa de profesores, a partir de la cual se irán seleccionando uno o varios de ellos para impartir cursos.

De los profesores nos interesa conocer además de sus datos personales, los estudios realizados, cursos impartidos, horario preferente en el cual impartir cursos y cursos de interés personal.

**Empleados:**

Almacenaremos sus datos personales, titulación académica y cargo que ocupan en el centro. Además debemos tener su sueldo y horario de trabajo.

En este punto se deberá tener en cuenta que dependiendo del puesto de trabajo desempeñado podrá tener un sueldo u otro, y el mismo se podrá calcular de distintas formas.

**Aulas:**

Debemos gestionar cada una de las aulas del centro, de forma que de cada una debemos tener indicado el nombre del aula, superficie en metros cuadrados, numero de plazas disponibles, si dispone de ordenadores y cuantos, y si dispone de Internet vía inalámbrica, cable o no tiene, además de si hay toma de corriente y cuantas. También sería interesante tener en cuenta si dispone de proyector o no.

Por ultimo nos indican que el aula se puede alquilar a personal externo, por lo que sería recomendable almacenar también el precio de alquiler por hora.

**Equipos Informáticos:**

En el centro se dispone de diverso material informático como pueden ser Ordenadores de Sobremesa, Ordenadores Portátiles, Impresoras...

Para cada uno de ellos nos interesa saber su numero de serie o referencia, descripción, características, sistemas operativos instalados, si esta operativo o no...

**Cursos:**

El centro de enseñanza se dedica a impartir cursos o master, por lo que se desea registrar el nombre del curso, numero de horas, temario, horario, Fecha de Inicio, Fecha de Finalización...

**Matriculas:**

El alumno visitara el centro para matricularse de uno de los cursos ofertados, por lo que se deberá registrar dicha reserva.

Se debe tener en cuenta que en cualquier momento se podrá consultar el numero de alumnos matriculados en un curso, o que dado un alumno nos muestre en los cursos en los que esta matriculado.