

Unidad 8 – Utilización avanzada de clases

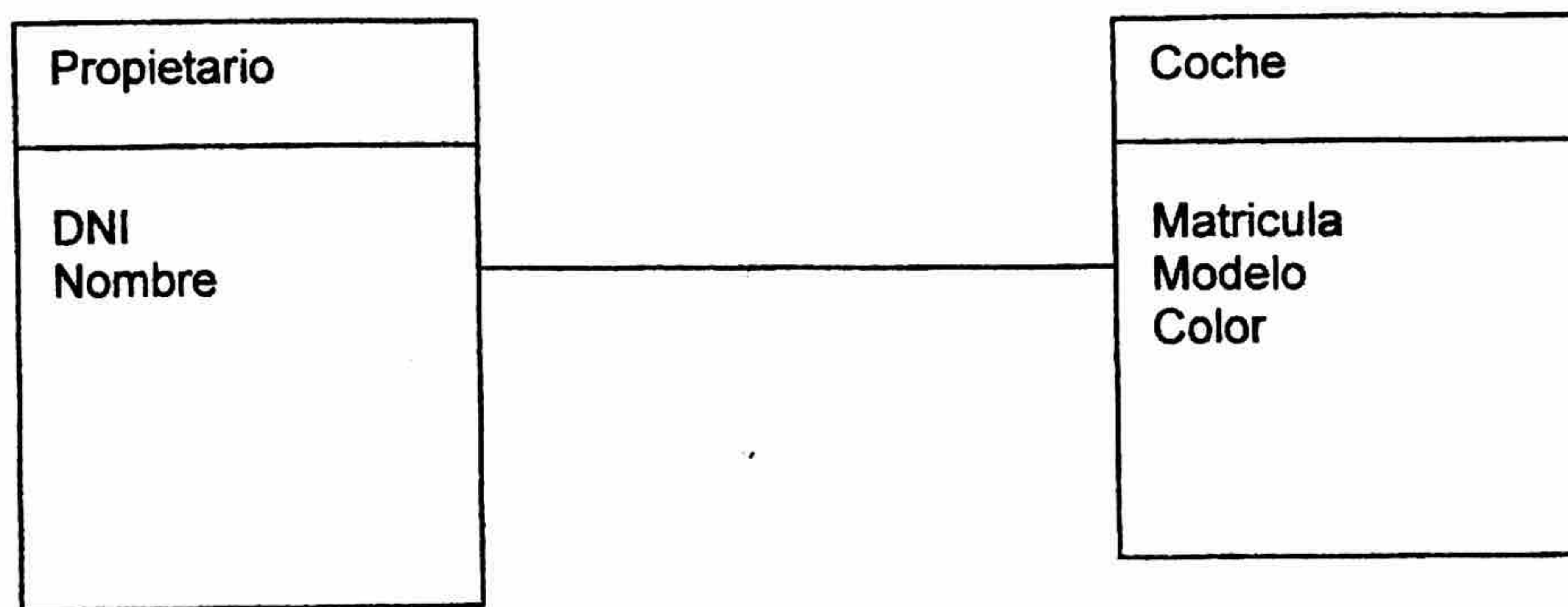
Nombre: <u>Pablo lobo</u> <u>Jorge de la Fuente</u>	Fecha: 13 / 4 / 2015	Ok: ☒
--	----------------------	---

Actividad 8.5

Objetivos

- ↓ Composición / Asociaciones
- ↓ Diferencias entre ambas
- ↓ Comprobar la relación entre las clases asociadas
- ↓ Diseño de clases

Tenemos una relación 1: 1 entre las clases **Propietario** y **Coche**



La clase **Coche** tiene un constructor con todos los parámetros y los métodos dame y pon de los atributos.

La clase **Propietario** tiene un constructor con los datos del propietario, otro con los del propietario más los del coche y los métodos dame y pon de los atributos.

ACTIVIDAD A- Composición

En el programa principal construir un Coche de matrícula 3131- SBC, modelo Audi A3 y color Rojo
Construir un objeto Propietario pidiendo sus datos por teclado y con el coche ya creado.

Mostrar los datos del propietario con los datos de su coche.

"PROPIETARIO Nombre: xxxxxx DNI: xxxxx COCHE Matricula: xxxxx Modelo: xxxx Color: xxxxx"

Cambiarle el color al coche Rosa

Mostrar de nuevo los datos del propietario con los datos de su coche

"PROPIETARIO Nombre: xxxxxx DNI: xxxxx COCHE Matricula: xxxxx Modelo: xxxx Color: xxxxx"

ACTIVIDAD B- Asociación

En el programa principal construir un Coche de matrícula 1818- SSB, modelo Opel Corsa y color Verde
Construir un objeto Propietario pidiendo sus datos por teclado y asociarlo con el coche ya creado.

Mostrar los datos del propietario con los datos de su coche

"PROPIETARIO Nombre: xxxxxx DNI: xxxxx COCHE Matricula: xxxxx Modelo: xxxx Color: xxxxx"

Cambiarle el color al coche a Azul

Mostrar de nuevo los datos del propietario con los datos de su coche

"PROPIETARIO Nombre: xxxxxx DNI: xxxxx COCHE Matricula: xxxxx Modelo: xxxx Color: xxxxx"

Unidad 8 – Utilización avanzada de clases

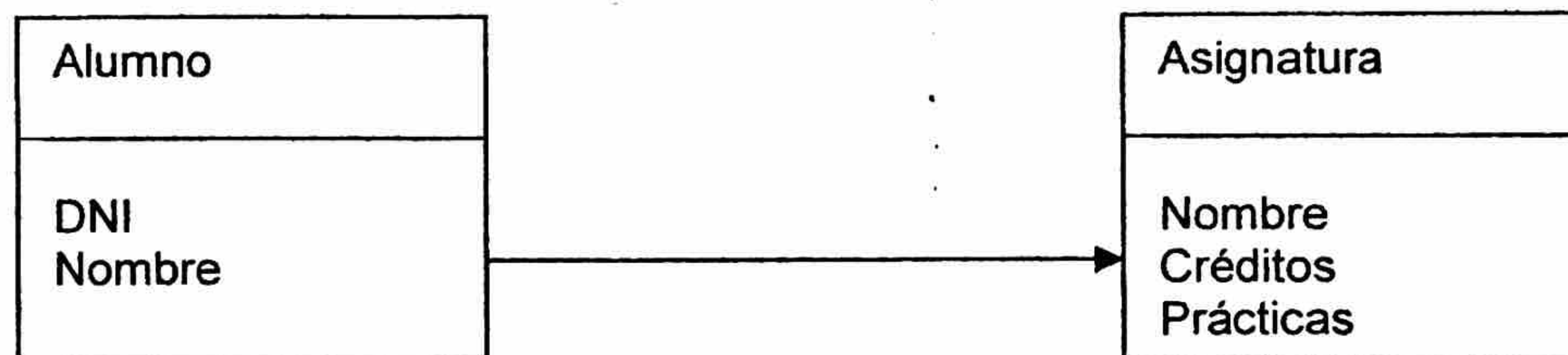
Nombre: <i>Pablo Lobo</i> <i>Jorge de La Fuente</i>	Fecha: 15 / 4 / 2015	Ok: ☒ <i>tiene un fallo.</i> <i>Muy bien en diseño</i>
--	----------------------	--

Actividad 8.6

Objetivos

- ↓ Asociaciones
- ↓ Cardinalidad de las relaciones
- ↓ Comprobar la relación entre las clases asociadas
- ↓ Diseño de clases

Tenemos una asociación entre clase Alumno que puede matricularse de hasta 6 Asignaturas. La visibilidad es de alumno a asignaturas.



Ambas clases tendrán un constructor por defecto, otro con todos los parámetros y los métodos que sean necesarios.

Programa PRINCIPAL:

- ✓ Construir un array con las asignaturas

1 Gestión Empresarial	6	no
2 Fundamentos de Electricidad y Electrónica	6	si
3 Métodos Matemáticos de la Ingeniería	12	no
4 Matemática Discreta y Lógica Matemática	12	no
5 Fundamentos de la Programación	12	si
6 Fundamentos de los Computador	12	no
7 Ampliación de Matemáticas	6	no
8 Tecnología y Organización de Computadores	6	si
9 Probabilidad y Estadística	6	no
10 Estructura de Computadores	6	si
11 Ingeniería del Software	9	no
12 Estructura de Datos y Algoritmo	9	si
13 Tecnología de la programación	12	si

El precio del crédito es 50 euros y si la asignatura tiene prácticas se pagan 50 euros más.

- ✓ Pedir por teclado los datos de un alumno y construir un objeto Alumno

Ofrecer el menú para ese alumno

1. MATRICULARSE DE UNA ASIGANTURA (habrá que comprobar que no se ha matriculado ya)
2. ANULAR LA MATRICULA DE UNA ASIGNATURA (comprobando que está matriculado de esa asignatura)
3. LISTAR LAS ASIGNATURAS (LOS NOMBRES) EN LAS QUE ESTÁ MATRICULADO
4. PREGUNTAR CREDITOS TOTALES MATRICULADO
5. PREGUNTAR PRECIO A PAGAR POR MATRICULA
6. FIN

Unidad 8 – Utilización avanzada de clases

Nombre: <i>Jorge de la Fuente</i> <i>Pablo Lobo</i>	Fecha: 20 / 4 / 2015	Ok: ☒
--	----------------------	---

Actividad 8.7

Objetivos

- ↓ Comprobación de herencia. Constructores de la subclase.
- ↓ Redefinición de funciones en la subclase
- ↓ Sobreescritura de métodos en las clases derivadas

Partimos de la estructura de clases del ejercicio 8.2

Figura

- Variables/atributos
forma: cadena [10]
color : cadena [10]
- Constructores
Por defecto. Da valor "" a la forma y al color
Recibe forma como parámetro Da valor "" al color
Recibe forma y color como cadena como parámetros
Recibe forma y color como número como parámetros
- Funciones / métodos
dameForma
dameColor
ponForma
ponColor: recibe el color como cadena
ponColor : recibe el color como número (método sobrecargado)

Define dos subclases que hereden de la clase Figura

Cuadrado

- Variables/Atributos
lado: int
- Constructores
Por defecto
Recibe el lado como parámetro
Recibe el lado y el color como cadena como parámetros
Recibe el lado y el color como número como parámetros
- Funciones/Métodos
ponLado
dameLado

Circulo

- Variables/Atributos
radio: float
- Constructores
Por defecto
Recibe el radio como parámetro:
Recibe el radio y el color como cadena como parámetros
Recibe el radio y el color como número como parámetros:
- Funciones/Métodos
ponRadio
dameRadio

Nota:

Correspondencia entre el color como cadena y como número :

Blanco.....	1	Verde.....	2
Azul.....	3	Rojo.....	4
Naranja.....	5	Violeta.....	6
Amarillo.....	7	Marrón.....	8
Rosa.....	9	Negro.....	10

En esta actividad vamos a añadir a la clase **Circulo** un método sobrescrito

void ponColor (int codColor)

de forma que ponga el siguiente color (es decir si se le indica el número 1, pondrá el color correspondiente al número 2, es decir Verde en lugar de Blanco y se le pasa el color 10 pondrá el color 1, es decir Blanco en lugar de Negro)

Actividad a

Hacer un programa main que construya de forma estática 2 objetos de la clase Circulo y 2 objetos Cuadrado

- b) un objeto **cuadrado2** y otro **circulo2** al que se le pasa como parámetro un lado/radio de valor 15
- c) un objeto **cuadrado3** y otro **circulo3** al que se le pasa como parámetro un lado/radio de valor 25 y un color "Blanco"

A continuación muestra los valores de los atributos radio, forma y color de todas las Figuras

A continuación cámbiales a todos el color a "Naranja" y volver a mostrar los valores de los atributos radio, forma y color.

A continuación cámbiales a todos el color al de código 10 y volver a mostrar los valores de los atributos radio, forma y color.

Actividad b

Hacer un programa main que construya de forma dinámica 2 objetos de la clase Circulo y 2 objetos Cuadrado

- b) un objeto **cuadrado2** y otro **circulo2** al que se le pasa como parámetro un lado/radio de valor 15
- c) un objeto **cuadrado3** y otro **circulo3** al que se le pasa como parámetro un lado/radio de valor 25 y un color "Blanco"

A continuación muestra los valores de los atributos radio, forma y color de todas las Figuras

A continuación cámbiales a todos el color a "Naranja" y volver a mostrar los valores de los atributos radio, forma y color.

A continuación cámbiales a todos el color al de código 10 y volver a mostrar los valores de los atributos radio, forma y color.

PROGRAMACION

Unidad 8 – Utilización avanzada de clases

Nombre: <i>Jorge de la Fuente</i> <i>Pablo Lobo</i>	Fecha: 20 / 4 / 2015	Ok: ☒
--	----------------------	---

Actividad 8.8

Objetivos

- ✚ Comprobación de herencia. Constructores de la subclase.
- ✚ Redefinición de funciones en la subclase
- ✚ Polimorfismo.
- ✚ Funciones virtuales
- ✚ Arrays de la clase base

Actividad a

Crea un array de 3 punteros a Figuras

- La primera Figura la crearás con el constructor **Figura** pasándole los parámetros "Triangulo" y "Verde".
- La segunda Figura la crearás un cuadrado con el constructor **Cuadrado** pasándole los parámetros 20 y "Azul".
- La tercera Figura la crearás un circulo con el constructor **Circulo** pasándole los parámetros 14 y "Rojo".

Trabaja con el array el resto del ejercicio (es decir con bucles). lo hemos creado solo con 3 elementos para simplificar

- Visualiza la forma y el color de todos los elementos del array
- Cambia el color a "Naranja" todos los elementos del array y visualiza la forma y el color de todos los elementos del array
- Cambia el color a "9" todos los elementos del array y visualiza la forma y el color de todos los elementos del array
- Modifica lo necesario para que ejecute el método ponColor de círculo para aquellas figuras que sean círculos.

Actividad b

Añade lo necesario para tener una función **área** que nos devuelva el área de cada figura

Crea un array de 9 **Figuras** (de forma dinámica, con punteros a Figura). Para darle valores pide por teclado los datos para 9 Figuras que podrán ser **Cuadrados** o **Círculos**.

Una vez formadas la Figuras recorre el array visualizando la forma, el color y el área de todos los elementos del array.

PROGRAMACION

Unidad 8 – Utilización avanzada de clases

Nombre: <i>Pablo lobo</i> <i>Jorge de la Fuente</i>	Fecha: 22 / 4 / 2015 Miércoles.	Ok: <i>C</i>
--	------------------------------------	--------------

Actividad 8.9

Objetivos

- ↓ Comprobación de herencia. Constructores de la subclase.
- ↓ Redefinición de funciones en la subclase
- ↓ Polimorfismo.
- ↓ Interfaces

Crea un array de 9 punteros a **Figuras**. Para darle valores pide por teclado los datos para 9 Figuras que podrán ser **Cuadrados** o **Círculos**.

Piensa qué métodos deberán ser **virtuales** o **virtuales puros**.

Actividad a

Crea una interfaz **Visualizar** que tenga el método **visualizarFigura()**

Modifica la clase **Figura** que implemente esa interfaz escribiendo los métodos:

- **void visualiarFigura()** mostrará el mensaje *"Figura con la forma <<forma de la figura>> y el color <<color de la figura>>"*

Recorre el array y visualiza con este método **visualizarFigura** todos los elementos del array

Actividad b

Crea una interfaz **Dibujar** que tenga los métodos **dibujar()** y **colorear()**.

Modifica la clase **Figura** que implemente esa interfaz escribiendo los métodos:

- **void dibujar()** mostrará el mensaje *"Estoy dibujando una figura con la forma <<forma de la figura>>"*
- **void colorear()** mostrará el mensaje *"Estoy coloreando una figura con el color <<color de la figura>>"*

Recorre el array y visualiza, dibuja y colorea todos los elementos del array.

Actividad c

Modifica la clase **Cuadrado** (que hereda de **Figura**) para que implemente esta interfaz escribiendo los métodos:

- **void dibujar()** mostrará el mensaje *"Estoy dibujando un cuadrado con lado <<valor del lado>>"*
- **void colorear()** mostrará el mensaje *"Estoy coloreando un cuadrado con el color <<color del cuadrado>>"*

Modifica la clase **Circulo** (que hereda de **Figura**) para que implemente esta interfaz escribiendo los métodos:

- **void dibujar()** mostrará el mensaje *"Estoy dibujando un circulo con radio <<valor del radio>>"*
- **void colorear()** mostrará el mensaje *"Estoy coloreando un circulo con el color <<color del circulo>>"*

Recorre el array y visualiza, dibuja y colorea todos los elementos del array comprobando que se ejecuta para cada objeto el método correspondiente de su clase.

PROGRAMACION

Unidad 8 – Utilización avanzada de clases

Nombre: <i>Jorge de la Fuente</i> <i>Pablo Lobo</i>	Fecha: 23 / 4 / 2015	Ok: <i>✓</i>
--	----------------------	--------------

Actividad 8.10

Objetivos

- ✚ Herencia
- ✚ Redefinición de funciones en la subclase
- ✚ Polimorfismo.
- ✚ Interfaces
- ✚ Diseño de clases

La empresa "MUEBLES ARTESANOS" paga a sus empleados por meses. Los empleados son de 4 tipos: los empleados asalariados que reciben un salario fijo mensual, sin importar el número de horas trabajado; los empleados por horas que reciben un salario de 25 euros por hora y pago extra del 10% por el número de horas que exceda de 40; los empleados por comisión que reciben un porcentaje de las ventas (el porcentaje es variable para cada empleado) y los empleados por comisión con salario base que reciben un salario base más un porcentaje de sus ventas (igualmente variable para cada empleado).

Algunos meses, según beneficios, la empresa decide recompensar a sus empleados incrementando un porcentaje el salario de cada empleado. Ese porcentaje depende del mes.

La empresa desea implementar un programa que cada mes pida los datos necesarios y muestre la nómina de la forma:

MUEBLES ARTESANOS NOMINA DEL MES DE XXXXXXXXXXXXXXXX	
Nombre: xxxxxxxxxxxxxxxxxxxxxx	Domicilio: xxxxxxxxxxxxxxxxxxxxxx
Banco: xxxxxxxxxxxxxxxxxxxxxx	
Tipo de empleado: xxxxxxxxxxxxxxxxxxxxxx	
<<<información del Salario>>>	
Bonificación: xxxx euros (1)	
Salario final: xxxxx euros (1)	

(1) estas líneas solo aparecerán si hay ese mes hay beneficios

<<<información del salario>>>

Empleados asalariados	Salario total: xxxx euros
Empleados por horas	Número de horas: xxxx Salario base: xxxx euros Extra por horas que exceden de 40: xxxx euros Salario total: xxxx euros
Empleados por comisión	Total vendido: xxxx euros Porcentaje de comisión: xx % Salario total: xxxx euros
Empleados por comisión con salario base	Salario base: xxxx euros Total vendido: xxxx euros Porcentaje de comisión: xx % Salario total: xxxx euros

Debe hacerse un diseño con una interfaz **InterfazNomina** que tenga los métodos necesarios para poder escribir la salida de la nómina de cada uno de los empleados

PROGRAMACION

Unidad 8 – Utilización avanzada de clases

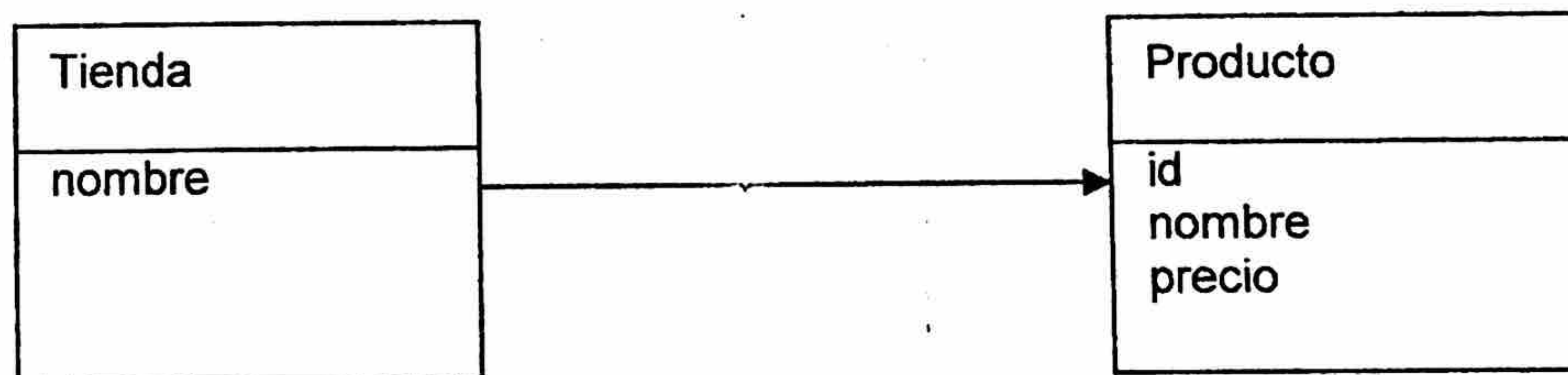
Nombre:	Fecha: 29 / 4 / 2015	Ok:
---------	----------------------	-----

Actividad 8.11

Objetivos

- ↓ Asociaciones
- ↓ Cardinalidad de las relaciones
- ↓ Comprobar la relación entre las clases asociadas
- ↓ Diseño de clases

Tenemos una asociación entre Tienda que puede tener hasta 10 Productos. La visibilidad es de Tienda a Producto



Ambas clases tendrán un constructor por defecto, otro con todos los parámetros y los métodos que sean necesarios.

Pedir por teclado los datos de dos tiendas.
Construir un array con 10 productos.

Ofrecer el menú para ese alumno

1. RECIBIR UN PRODUCTO
2. VENDER UN PRODUCTO
3. CAMBIAR EL PRECIO A UN PRODUCTO
4. CAMBIAR EL NOMBRE A UN PRODUCTO
5. FIN

Opción 1- La tienda recibe un producto (lo tendrá en su "almacén")

Introduce la tienda:

Introduce el id del producto que acabas de recibir:

Opción 2: la tienda vende un producto (lo sacará de su "almacén")

Introduce la tienda:

Introduce el id del producto que quieres comprar:

mostrar el nombre del producto y el precio a pagar o un mensaje si no está disponible en la tienda

Opción 3: se cambia el precio a un producto para todas las tiendas

Introduce el id del producto que quieres cambiar el precio:

Introduce el nuevo precio:

Opción 4: se cambia el nombre a un producto para todas las tiendas

Introduce el id del producto que quieres cambiar el nombre:

Introduce el nuevo nombre:

EJECUTA LAS OPCIONES NECESARIAS PARA COMPROBAR QUE LO HAS HECHO CON UNA ASOCIACIÓN.

PROGRAMACION

C

Unidad 10 – Colecciones

Actividad 10.1

Objetivos

- Estudio de un enunciado para proponer una solución utilizando POO
- Definir una clase con sus miembros: variables y funciones (atributos y métodos)
- Estudio de las colecciones Pila y Cola
- Implementación con un array estático.

Se desea implementar dos clases, **Pila** y **Cola**, y los métodos que las manejen

Pila es una estructura LIFO (Last In First Out), el último elemento en llegar será el primero en salir)
Cola es una estructura FIFO (First In First Out), el primer elemento en llegar será el primero en salir)

Ambas implementarán la interfaz **InterfazPC** con los métodos para trabajar con ellas.

Interfaz PC
bool vacia() void push(int) int pop()

Diseñar las clases **Pila** y **Cola** que implementen la interfaz sabiendo que para ambas estructuras se quieren implementar además las siguientes funciones:

➤ Constructor sin parámetros: inicializará lo necesario	Pila / Cola
➤ Push: introducirá un elemento de la estructura	push
➤ Pop: sacará un elemento en la estructura y lo devolverá con return	pop
➤ Delete: borrará todos los elementos de la estructura	delete
➤ Lleno: comprobará si la estructura está llena	llena
➤ Vacio: comprobará si la estructura está vacía	vacía

Estarán implementadas sobre una estructura estática (array estático, de tamaño predefinido N) y supondremos que en ellas va a haber números enteros positivos.

Una vez tengas hecha las clases y compiladas harás un programa principal para comprobar el funcionamiento que preguntará con qué tipo de elemento se quiere trabajar u ofrecerá el menú:

Ofrecer el menú.

1. **Push:** insertar un elemento
2. **Pop:** sacar un elemento con return
3. **Delete:** borrará todos los elementos
4. **Fin**

PROGRAMACION

Unidad 10 – Colecciones

Nombre: Jorge de la Fuente Jiménez Pablo Lobo	Fecha: 4 / 5 / 2015	Ok: 
--	---------------------	---

: Buscar las 7 diferencias -

Actividad 10.2

Objetivos

- Estudio de un enunciado para proponer una solución utilizando POO
- Definir una clase con sus miembros: variables y funciones (atributos y métodos)
- Estudio de las colecciones Pila y Cola
- Implementación con un array dinámico.

Se desea implementar dos clases, **Pila** y **Cola**, y los métodos que las manejen

Pila es una estructura LIFO (Last In First ON, el último elemento en llegar será el primero en salir)

Cola es una estructura FIFO (First In First ON, el primer elemento en llegar será el primero en salir)

Estarán implementadas sobre una estructura dinámica y supondremos que en ellas va a haber números enteros positivos. Esta estructura tendrá un tamaño inicial que se le podrá pasar como parámetro y cuando esté llena crecerá un tamaño que se le podrá pasar como parámetro. Ambos tamaños (el inicial y que crecerá cuando se llene) tendrán un valor por defecto que será una constante T (por ejemplo para este ejercicio 10)

Ambas implementarán la interfaz **InterfazPC** con los métodos para trabajar con ellas.

Interfaz PC
bool vacia() void push(int) int pop()

Diseñar las clases Pila y Cola que implementen la interfaz sabiendo que para ambas estructuras se quieren implementar además las siguientes funciones:

- | | |
|---|-------------|
| ➤ Constructor sin parámetros: inicializará lo necesario | Pila / Cola |
| ➤ Constructores con parámetros | |
| ➤ Delete: borrará todos los elementos de la estructura | delete |

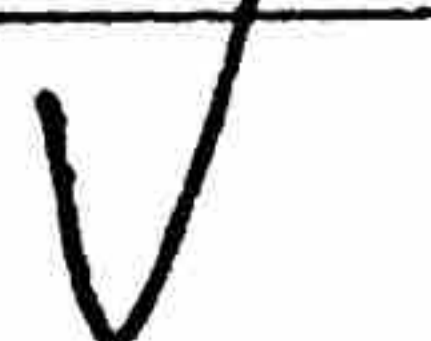
Una vez tengas hecha las clases y compiladas harás un programa principal para comprobar el funcionamiento que preguntará con qué tipo de elemento se quiere crecimiento trabajar, preguntará si se quiere definir el tamaño inicial y el tamaño de crecimiento (ambos pueden ser nulos y en ese caso se tomará la constante T ya definida) y ofrecerá el menú:

Ofrecer el menú.

1. **Push:** insertar un elemento
2. **Pop:** sacar un elemento
3. **Delete:** borrará todos los elementos
4. **Fin**

PROGRAMACION

Unidad 10 – Colecciones

Nombre: Jorge de la Fuente Pablo lobo	Fecha: 5 / 5 / 2015	Ok: 
--	---------------------	---

Actividad 10.3

Objetivos

- Estudio de un enunciado para proponer una solución utilizando POO
- Definir una clase con sus miembros: variables y funciones (atributos y métodos)
- Estudio de las colecciones Pila y Cola
- Implementación con un array ~~dinámico~~ **estático**
- Uso de las Pilas y Colas como contenedor de elementos de diferente clase

Se desea implementar dos clases, **Pila** y **Coia**, y los métodos que las manejen para trabajar sobre las clases que heredan de **Figura**.

Partiremos de las clases de **Pila** y **Cola** diseñadas en la Actividad **10.1** Ambas implementarán la interfaz **InterfazPC** con los métodos para trabajar con ellas.

Interfaz PC
bool vacia() void push(int) . int pop()

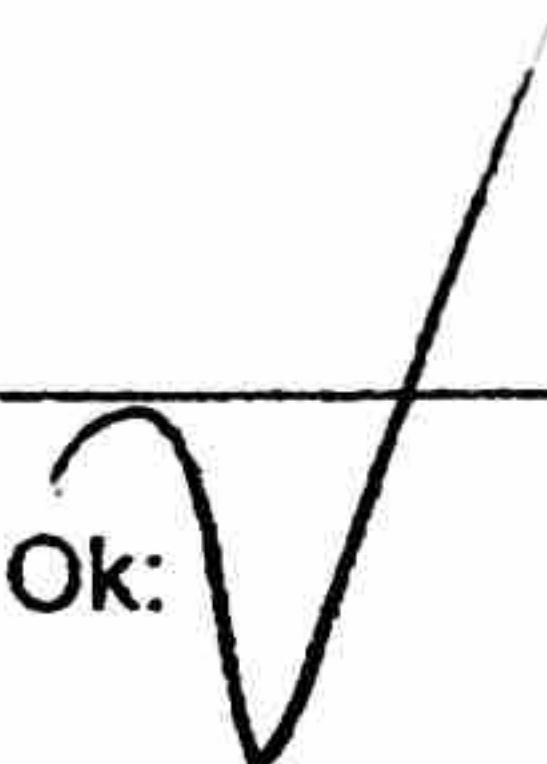
Estas estructuras pueden contener clases derivadas de la clase **Figura**, en este caso **Cuadrado** o **Círculo**. Partiendo de las implementadas en la Actividad 8.9 que implementan las interfaces **IVisualizar**, **IDibujar**.

Ofrecer el menú.

1. **Push**: insertar un elemento preguntando qué tipo de elemento se quiere insertar
2. **Pop**: sacar un elemento y visualizarlo, dibujarlo y colorearlo.
3. **Delete**: borrará todos los elementos
4. **Fin**

PROGRAMACION

Unidad 10 – Colecciones

Nombre: <i>Jorge de la Fuente</i> <i>Pablo Lobo</i>	Fecha: 6 / 5 / 2015	Ok: 
--	---------------------	---

Actividad 10.4

Objetivos

- ✚ Concepto de Nodo
- ✚ Definición de Listas
- ✚ Operaciones básicas con listas simplemente enlazadas
- ✚ Algoritmia

Partiendo del ejemplo hecho en clase con
class Nodo

```
{
    private: int num ;
            Nodo *sig;
    public:  Nodo();
            Nodo(int n);
            int dameNum();
            Nodo *dameSig();
            void ponNum(int n);
            void ponSig(Nodo *p_nodo);
};
```

Desarrollar la clase Lista Simplemente Enlazada
class ListaSE

```
{
    private: Nodo *primero;
    public : ListaSE ();
            ~ListaSE()
            void insertarNodoFinal(int n);
            void insertarNodoPrincipio(int n);
            bool borrarNodo(int n);
            Nodo *buscarNodo(int n);
            // capa de entrada/salida (debería estar en la interfaz de E/S)
            void mostrarLista();
};
```

Haz un programa principal que la pruebe.

Implementar el menú:

1. INSERTAR NODO PRINCIPIO
2. INSERTAR NODO FINAL
3. BORRAR UN NODO
4. BUSCAR UN NODO
5. MOSTRAR LA LISTA
- 6.FIN

Prestar atención al probar e incluir los casos de prueba en que la lista esté vacía.

PROGRAMACION

Unidad 10 – Colecciones

Jorge de la Fuente
Pablo Lobo

Actividad 10.5

Objetivos

- ↓ Concepto de Nodo
- ↓ Definición de Listas
- ↓ Operaciones básicas con listas simplemente enlazadas ordenadas
- ↓ Diseño de clases

Partiendo de la definición de un nodo compuesto por un objeto Alumno:

```
class Nodo
{
    private: Alumno alumno;
            Nodo *sig;
    public:  Nodo();
            Nodo(int n);
            Alumno dameAlumno();
            Nodo *dameSig();
            void ponAlumno(Alumno a);
            void ponSig(Nodo *p_nodo);
};
```

```
class Alumno
{
    private: int numExp;
            char nombre[20];
    public : Alumno();
            Alumno (int numExp, char nombre[]);
            int dameNumExp();
            char *dameNombre();
            void ponNumExp(int n);
            void ponNombre(char nombre[]);
};
```

Desarrollar la clase Lista Simplemente Enlazada Ordenada (siempre estará ordenada Alumno, lo que equivale a estar ordenada por NumExpediente). Estas clases se desarrollarán para poder realizar lo que indica el menú.

Implementar el menú:

1. INSERTAR ALUMNO
insertará ordenadamente
2. BUSCAR UN ALUMNO
dado el numExpediente mostrará el nombre o un mensaje si no existe.
nota: dado que la lista está ordenada, la búsqueda será ordenada
3. BORRAR UN ALUMNO
borrará el alumno cuyo numExpediente se introduce, mostrando un mensaje si se ha realizado o no
4. MOSTRAR UN ALUMNO DADO SU NÚMERO EN LA LISTA
dado un número mostrará el alumno que ocupa ese lugar EN LA LSITA o un mensaje si no existe
5. MOSTRAR LA LISTA
- 6.FIN

PROGRAMACION

Unidad 10 – Colecciones

Nombre:	Fecha: 11 / 5 / 2015	Ok:
---------	----------------------	-----

Actividad 10.6

Objetivos

- ✚ Concepto de Nodo
- ✚ Definición de Listas
- ✚ Operaciones básicas con listas doblemente enlazadas
- ✚ Lista con dos "entradas": primero y último
- ✚ Algoritmia

class *Nodo*

```
{
    private: Alumno alumno;
            Nodo *sig;
            Nodo *ant;
    public: Nodo();
            Nodo(Alumno a);
            Alumno dameAlumno();
            Nodo *dameSig();
            Nodo *dameAnt();
            void ponAlumno(Alumno a);
            void ponSig(Nodo *p_nodo);
            void ponAnt(Nodo *p_nodo);
};
```

class *ListaDE*

```
{
    private: Nodo *primero;
            Nodo *ultimo;
    public : ListaDE();
};
```

class *Alumno*

```
{
    private: int numExp;
            char nombre[20];
    public : Alumno();
            Alumno (int numExp, char nombre);
            int dameNumExp();
            char *dameNombre();
            void ponNumExp(int n);
            void ponNombre(char nombre);
};
```

Definir los métodos para poder insertar un alumno al inicio, insertar un alumno al final, buscar un alumno y borrar un alumno
También los métodos de la capa de entrada /salida para mostrar de inicio a fin y de fin a inicio.

Haz un programa principal que la pruebe. Creará la lista vacía y mostrará el menú:

1. AÑADIR UN ALUMNO A LA LISTA POR EL INCIO
2. AÑADIR UN ALUMNO A LA LISTA POR EL FINAL
3. BUSCAR UN ALUMNO POR NUMEXPEDIENTE
4. BORRAR UN ALUMNO DE LA LISTA
5. MOSTRAR TODOS LOS ALUMNOS DE INICIO A FIN
6. MOSTRAR TODOS LOS ALUMNOS DE FIN A INICIO
7. FIN

PROGRAMACIÓN

UNIDAD 11. Ficheros

Nombre: <u>Jorge de la Fuente</u> <u>Pablo Lobo</u>	Fecha: 22 / 4 / 2015	Ok: ☒
--	----------------------	---

Actividad 11.1

Objetivos

- Trabajar con ficheros secuenciales
- Practicar las diferentes formas de apertura del fichero
- Practicar los procesos que se pueden realizar con ficheros secuenciales
- Concepto de registro

Se tiene un almacén de productos donde se guarda en un fichero secuencial los datos de los productos que dispone.

El fichero almacena productos

Productos
- Num_producto: entero - Descripción : cadena - Existencias: entero
+ Producto() + Producto(con parámetros) + dame y pon

Operaciones básicas

1- Crear un fichero secuencial

El nombre externo "*Productos.dat*" y se irán introduciendo los datos de los productos, formado un objeto Producto y grabándose. Los datos de los registros a grabar se introducen por teclado hasta que se introduce num_producto=0.

2- Listado de todo el fichero

Sacar por pantalla los datos de todos los productos del almacén.

3- Listado de producto con existencias cero.

Sacar por pantalla el número de producto y la descripción de los productos de los que haya existencias cero.

4- Búsqueda de los datos de un producto.

Se pedirá el número de producto y se sacará por pantalla los datos de ese producto o un mensaje si no existe.

5- Altas al final del fichero

Los productos que llegan nuevos se añaden al final del fichero. Pedir los datos de dos nuevos productos y añadirlos al final del fichero.

Una vez añadidos hacer un listado de todo el fichero para ver que se han añadido bien.

6- Listar los dos últimos productos

Listar de nuevo el fichero para ver que se han añadido bien

PROGRAMACIÓN

UNIDAD 11. Ficheros

Nombre: <i>Jorge de la Fuente</i> <i>Pablo lobo</i>	Fecha: 18 / 5 / 2015	Ok: ☒
--	----------------------	---

Actividad 11.2

Objetivos • Trabajar con ficheros secuenciales • Actualización de ficheros secuenciales • Comprender el funcionamiento de la instrucción para posicionarse en el fichero (mover el puntero en el flujo) • Aprender la forma habitual de trabajar con ficheros secuenciales.
--

Se tiene un almacén de productos donde se guarda en un fichero secuencial los datos de los productos que dispone.
El fichero almacena productos y tiene un registro con la forma

Productos
- Num_producto: entero - Descripción : cadena - Existencias: entero
+ Producto() + Producto(con parámetros) + dame y pon

Partiendo de el fichero "productos.dat" creado en la actividad 11.1

Ofrecer el menú:

- 1 - ALTA DE UN PRODUCTO
- 2 - BAJA DE UN PRODUCTOS
- 3 - BUSQUEDA DE UN PRODUCTO
- 4 - MODIFICACIÓN DE LAS EXISTENCIAS DE UN PRODUCTO
- 5 - MOSTRAR TODO EL FICHERO
- 6 - FIN

- 1 - ALTA DE UN PRODUCTO: pedirá los datos y añadirá el producto al final del fichero
- 2 - BAJA DE UN PRODUCTOS: pedirá el número de producto y hará la baja física
- 3 - BUSQUEDA DE UN PRODUCTO: pedirá el número de producto y mostrará las descripción y las existencias
- 4 - MODIFICACIÓN DE LAS EXISTENCIAS DE UN PRODUCTO: pedirá el número de producto y las nuevas existencias y las sumará con las que haya
- 5 - MOSTRAR TODO EL FICHERO: mostrará todos los productos que no estén dados de baja
- 6 - FIN: Creará un nuevo fichero "prodcutos2.dat" sin los productos que estén dados de baja.

PROGRAMACIÓN

UNIDAD 11. Ficheros

Nombre: <u>Jorge de la Fuente</u> <u>Pablo lobo</u>	Fecha: 18 / 5 / 2015	Ok: 
--	----------------------	---

Actividad 11.3

Objetivos

- Trabajar con ficheros secuenciales
- Cargar y salvar una estructura en un fichero secuencial

Tenemos definido un array Array Elementos [N]

```
class Elemento
{
    private: int num_producto;
            int posicion;
    public: Elemento();
            Elemento (int n, int p);
            ind dameNumProd();
            int damePosicion();
            void ponNumProd(int n);
            void ponPosicion();
};
```

El constructor por defecto pondrá los valores a "nulo" que será una constante NULO predefinida con el valor -9.

Programa principal:

- Al inicio cargar el array: se leerá de un fichero secuencial "Alumnos.dat". Todos los registros del fichero se cargarán en el array. Si hay menos registros en el fichero que el tamaño del array el resto de los elementos del array deberán tener el valor "nulo".
La primera vez que se ejecute el programa (que el fichero no existirá) el array se completará con valores "nulos".
- Ofreceremos un menú que nos permita modificar el array, añadir, quitar o cambiar elementos del array.
- Salvaremos el array en el fichero secuencial, sobrescribiendo lo que hubiese anteriormente. SOLO debe grabarse en el fichero los elementos que no sean "nulos".

* realizar con Lola y Pika