Bloque práctico 1.2: *C++*

Polimorfismo y *dinamic_cast< >*.

En esta segunda parte de la práctica 1 utilizaremos otras capacidades adicionales de programación orientada a objetos, tanto básicas como avanzadas, de *C++*. El objetivo de la práctica es programar una clase Empresa polimórfica capaz de agrupar los diferentes tipos de contratos de los clientes en dicho objeto.

0) Antes de empezar.

Cree un nuevo directorio denominado `Empresa` y dentro un proyecto para compilar todos los ficheros necesarios para realizar esta segunda parte de la práctica siguiendo la misma filosofía modular de la práctica anterior (por cada clase creamos un .h y un .cpp).

1) Hacer que `Contrato` muestre un comportamiento polimórfico dinámico: polimorfismo.

Modifica el programa para que éste tenga un comportamiento polimórfico y permita que el siguiente código añadido al final del fichero Prueba2.cpp produzca la salida indicada.

añadir al final de Prueba2.cpp, antes del system ("PAUSE")

```
  Contrato *t[4];
  t[0]=p;  t[1]=&c;  t[2]=&ct2;  t[3]=&cm1;
  cout << "\n-- Datos de los contratos: -- \n";
  t[3]->setDniContrato(75547111);
  for(int i=0; i<4; i++) {
    t[i]->setFechaContrato(t[i]->getFechaContrato()+2);
    t[i]->ver(); cout << endl;
  }
```

Salida:

```
-- Datos de los contratos: --
17333256 (1 - 02/03/2001)
23000111 (2 - 04/02/2002)
12555100 (4 - 02/03/2004) 320m, 350(12.00)
75547111 (6 - 02/03/2001) 160m, ESPAÑOL 0.12
```

2) Hacer que `Contrato` sea una clase abstracta

Realice las modificaciones necesarias para que el compilador restrinja la instanciación de la clase *`Contrato`*, de forma que no se puedan crear objetos de dicha clase.

Pruebe el programa anterior y compruebe que el compilador da error cuando intentamos crear instancias de la clase *`Contrato`*. Comenta las líneas del fichero Prueba2.cpp que ahora dan problemas y compruebe que todo funciona correctamente.

Si eliminamos la posibilidad de crear objetos de la clase *`Contrato`*, ¿es esto equivalente a quitar la clase de nuestro diseño?

3) Programar la clase *Empresa*

Programe la clase `Empresa`, la cual debe permitir crear y eliminar `Clientes` y `Contratos` (ya sean de tipo Tarifa Plana o Movil) en nuestra aplicación.

Para crear un contrato es necesario primero que el cliente sea dado de alta en la aplicación, introduciendo sus datos (los indicados en la clase `Cliente`). Una vez dado de alta un cliente, éste puede contratar las tarifas (Tarifa Plana o Movil) que necesite.

Programe en la clase *Empresa* los siguientes métodos:

- `crearContrato()`: Este método debe tener en cuenta si el contrato es para un nuevo cliente o para uno ya existente, para evitar añadirlo más de una vez. Para ello el método `crearContrato()` pedirá el DNI del titular del contrato y buscará si ya es o no cliente de la empresa. Si es un nuevo cliente (no existe un cliente con dicho DNI en la empresa) lo dará de alta, pidiendo los datos necesarios. Una vez el cliente es identificado o dado de alta, pedirá los datos del tipo de contrato que desea abrir (`ContratoTP` o `ContratoMovil`).

- `cancelarContrato()`: que se encargará de darlo de baja de la Empresa (y eliminarlo de la memoria). Para ello bastará con introducir el número de contrato (`idContrato`) a eliminar y pasarlo como parámetro al método. Si el contrato no existe, el método debe mostrar un mensaje por pantalla indicando tal circunstancia.

- `bajaCliente()`: Si un cliente desea darse de baja, bastará con pasar su `dni` como parámetro al método el cual se encargará de eliminar todos sus contratos y sus datos personales de la Empresa.

Implementa la clase *Empresa* de manera que mantenga un array con la lista de contratos y otro con la lista de clientes que va creando. Hacer que la empresa admita como máximo 100 clientes (uso de array estático) de forma que no se permita superar dicha cifra. En cuanto al número de contratos hacer que éste sea ilimitado (uso de array dinámico, de tamaño inicial de 10, que se duplica cada vez que se llena: 10, 20, 40, 80…)

4) *dynamic_cast<> y typeid*.

Programe los métodos `descuento(float porcentaje)` que permite rebajar la tarifa de todos los contratos Movil en el porcentaje indicado, y `nContratosTP()` que devuelve el número de contratos de Tarifa Plana existentes.

Deberá utilizar la funcionalidad `dynamic_cast<>` y `typeid` que se definen en la cabecera `<typeinfo>` para la programación de estos métodos.

Utilice el operador `dynamic_cast<>` para implementar el primer método para poder hacer una conversión dinámica del tipo base `Contrato` al tipo `ContratoMovil` ya que hay que acceder a funciones sólo existentes en la clase derivada `ContratoMovil` a través de punteros a la clase base `Contrato`. Para el segundo método utilice el operador `typeid` que permite determinar el tipo de un objeto en tiempo de ejecución.

CONSIDERACIONES A TENER EN CUENTA:

Para no alargar la práctica en exceso, a la hora de programar la clase Empresa vamos a suponer que NUNCA vamos a asignar una Empresa a otro Empresa en el `main()` y que NUNCA vamos a pasar una Empresa por copia ni a devolverlo en ningún método.

Por tanto al suponer que NUNCA vamos a utilizar en el `main()` el operador de asignación ni el constructor de copia, no programaremos dichos métodos, aunque la clase `Empresa` tenga atributos de tipo puntero en su interior (esta es una simplificación que vamos a hacerle a la práctica para acortarla un poco).

Teniendo en cuenta lo dicho anteriormente, programe la clase `Empresa` (así como los métodos de dicha clase) de forma que el siguiente código de prueba pueda ser ejecutado y produzca la salida siguiente:

Prueba3.cpp

```
#include <cstdlib>
#include <iostream>
#include <iomanip> //std::setprecision
#include "Fecha.h" //definicion clase Fecha
#include "Cliente.h" // definicion clase Cliente
#include "Contrato.h" // definicion de la clase Contrato
#include "ContratoTP.h" // definicion de la clase ContratoTP
#include "ContratoMovil.h" // definicion de la clase ContratoMovil
#include "Empresa.h" // definicion de la clase Empresa

using namespace std;

int main(int argc, char *argv[])
{
  bool ok;
  Empresa Yoigo;

  cout << setprecision(2) << fixed; //a partir de aqui float se muestra con 2 decimales
  cout << endl << "APLICACION DE GESTION TELEFONICA\n" << endl;
  Yoigo.cargarDatos(); //crea 3 clientes y 7 contratos. metodo creado para no
  Yoigo.ver();         //tener que meter datos cada vez que pruebo el programa
  cout <<"Yoigo tiene " << Yoigo.nContratosTP() << " Contratos de Tarifa Plana\n\n";

  Yoigo.crearContrato(); //ContratoMovil a 37000017 el 01/01/2017 con 100m a 0.25
  Yoigo.crearContrato(); //ContratoTP a 22330014 (pepe luis) el 2/2/2017 con 305m

  ok=Yoigo.cancelarContrato(28);   //este Contrato no existe
  if (ok) cout << "Contrato 28 cancelado\n"; else cout << "El Contrato 28 no existe\n";

  ok=Yoigo.cancelarContrato(4);    //este Contrato si existe
  if (ok)
    cout << "El Contrato 4 ha sido cancelado\n";
  else
    cout << "El Contrato 4 no existe\n";

  ok=Yoigo.bajaCliente(75547001); //debe eliminar el cliente y sus 3 Contratos
  if (ok) cout << "El cliente 75547001 y sus Contratos han sido cancelados\n";
  else cout << "El cliente 75547001 no existe\n";

  Yoigo.ver();

  Yoigo.descuento(20);
  cout << "\nTras rebajar un 20% la tarifa de los ContratosMovil...";
  Yoigo.ver();
  cout <<"Yoigo tiene " << Yoigo. nContratosTP () << " Contratos de Tarifa Plana\n";

  system("PAUSE");
  return 0;
}
```

Salida:

```
APLICACION DE GESTION TELEFONICA

La Empresa tiene 3 clientes y 7 contratos
Clientes:
Peter Lee (75547001 - 28 feb 2001)
Juan Perez (45999000 - 29 feb 2000)
Luis Bono (37000017 - 31 ene 2002)

Contratos:
75547001 (1 - 28 feb 2001) 110m, DANES 0.12 - 13.20€
75547001 (2 - 31 ene 2002) 170m, DANES 0.09 - 15.30€
37000017 (3 - 01 feb 2002) 250m, 300(10.00) - 10.00€
75547001 (4 - 28 feb 2001) 312m, 300(10.00) - 11.80€
45999000 (5 - 31 ene 2002) 202m, ESPAÑOL 0.10 - 20.20€
75547001 (6 - 31 ene 2002) 80m, DANES 0.15 - 12.00€
45999000 (7 - 01 feb 2002) 400m, 300(10.00) - 25.00€
Yoigo tiene 3 contratos de Tarifa Plana

Introduzca dni: 37000017
Tipo de Contrato a abrir (1-Tarifa Plana, 2-Movil): 2
Fecha del contrato
dia: 1 mes: 1 anio: 2017
minutos hablados: 100
Precio minuto: 0.25
Nacionalidad: alemán

Introduzca dni: 22330014
Nombre del cliente: pepe luis
dia: 2 mes: 2 anio: 2017
Tipo de Contrato a abrir (1-Tarifa Plana, 2-Movil): 1
Fecha del contrato
dia: 2 mes: 2 anio: 2017
minutos hablados: 305

El contrato 28 no existe
El contrato 4 ha sido cancelado
El cliente 75547001 y sus contratos han sido cancelados

La Empresa tiene 3 clientes y 5 contratos
Clientes:
Juan Perez (45999000 - 29 feb 2000)
Luis Bono (37000017 - 31 ene 2002)
pepe luis (22330014 - 02 feb 2017)

Contratos:
37000017 (3 - 01 feb 2002 ) 250m, 300(10.00) - 10.00€
45999000 (5 - 31 ene 2002 ) 202m, ESPAÑOL 0.10 - 20.20€
45999000 (7 - 01 feb 2002 ) 400m, 300(10.00) - 25.00€
37000017 (8 - 01 ene 2017 ) 100m, aleman 0.25 - 25.00€
22330014 (9 - 02 feb 2017 ) 305m, 300(10.00) - 10.75€

Tras rebajar un 20% la tarifa de los ContratosMovil...

La Empresa tiene 3 clientes y 5 contratos
Clientes:
Juan Perez (45999000 - 29 feb 2000)
Luis Bono (37000017 - 31 ene 2002)
pepe luis (22330014 - 02 feb 2017)

Contratos:
37000017 (3 - 01 feb 2002 ) 250m, 300(10.00) - 10.00€
45999000 (5 - 31 ene 2002 ) 202m, ESPAÑOL 0.08 - 16.16€
45999000 (7 - 01 feb 2002 ) 400m, 300(10.00) - 25.00€
37000017 (8 - 01 ene 2017 ) 100m, aleman 0.20 - 20.00€
22330014 (9 - 02 feb 2017 ) 305m, 300(10.00) - 10.75€

Yoigo tiene 3 contratos de Tarifa Plana

Presione una tecla para continuar . . .
```