PROGRAMACIÓN ORIENTADA A OBJETOS

GRADO EN INGENIERÍA INFORMÁTICA
SEGUNDO CURSO

DEPARTAMENTO DE INFORMÁTICA Y ANÁLISIS NUMÉRICO
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CÓRDOBA

CURSO ACADÉMICO: 2011 – 2012

- **Examen de prácticas: 23 de enero de 2012**

- **Objetivo**
  - Definir y utlizas siguientes clases
    - **Persona**:
    - **Socio**:
      - Hereda de forma pública de Persona
      - Representa a una persona que es socia de una ONG
    - **Ong**:
      - Clase que representa el TAD “organización no gubernamental” y está compuesta por socios

---

- **Definición de la clase Persona**
  - Posee los atributos
    - _nombre
    - _apellidos
    - _edad

  - Funciones o métodos públicos
    - Constructor:
      - Versión 1
        - Constructor parametrizado
        - Todos los argumentos deberán tener un valor por defecto
      - Versión 2
        - Constructor de copia

  - Funciones de acceso
    - getNombre
    - getEdad
    - getSexo

- Funciones de modificación
  - setNombre
  - setEdad
  - setSexo

- Funciones de lectura y escritura
  - leerPersona
    - Lee desde el teclado los valores de los atributos de la Persona actual.
  - escribirPersona
    - Escribe por pantalla los valores de los atributos de la Persona actual.

- **Definición de la clase Socio**

  - Socio hereda de forma pública de la clase Persona
  - Atributo propio
    - _donativo, que representa el dinero que dona mensualmente a la ONG

  - Funciones o métodos públicos
    - Constructor:
      - Versión 1
        - Constructor parametrizado
        - Todos los argumentos deberán tener un valor por defecto
      - Versión 2
        - Constructor de copia

    - Funciones de acceso
      - getDonativo

    - Funciones de modificación
      - setDonativo

      - Sobrecarga del operador de asignación “=”

    - Funciones de lectura y escritura
      - leerSocio
        - Lee desde el teclado los valores de los atributos del Socio actual.
      - escribirSocio
        - Escribe por pantalla los valores de los atributos del Socio actual.

    - Sobrecarga de los operadores “<<” y “>>”
      - Sobrecargar el operador “>>” para leer desde el teclado los

valores de los atributos del Socio actual.
- Sobrecargar el operador “<<” para escribir por pantalla los valores de los atributos del Socio actual.

- **Definición de la clase Ong**

  - Descripción
    - Clase que representa a las organizaciones no gubernamentales.

  - Atributos
    - _nombreOng
    - _vectorSocios
      - Cada alumno deberá decidir cómo implementar el vector de socios: mediante memoria dinámica o usando el contenedor STL vector

  - Funciones o métodos públicos
    - Constructor: Ong
      - Versión 1: constructor sin argumentos

  - Destructor de la clase Ong
    - Libera la memoria ocupada por la Ong actual

  - Métodos de acceso para los atributos propios
    - getNombreOng
      getVectorSocios
    - getNumeroSocios

  - Otra función de consulta
    - getSocio:
      - Recibe como parámetro un “indice” de tipo entero
      - Devuelve una referencia al Socio que ocupa el lugar señalado por “indice”.
    - Sobrecarga del operador []:
      - Recibe como parámetro un “indice” de tipo entero
      - Devuelve una referencia al Socio que ocupa el lugar señalado por “indice”.

  - Métodos de modificación para los atributos propios
    - setNombreOng
    - setVectorSocios

  - Sobrecarga del operador “=”

  - Funciones de lectura o escritura

- escribirOng
  - Escribe por pantalla el nombre y numero de socios de la Ong y los datos de sus socios.
- leerOng
  - Lee del teclado el nombre y numero de socios de la Ong y los datos de sus socios

- grabarOngEnFichero:
  - Recibe como parámetro el nombre de un fichero
  - Escribe en el fichero los datos de la Ong:
    - La primera línea del fichero debe contener el nombre de la Ong
    - La segunda línea debe contener el número de Socios
    - Las demás líneas deben contener los datos de cada Socio

---

- **Observaciones**
  - Las clases definidas deberán permitir la ejecución de los tres programas de prueba:
    - test1.cpp: comprobación de la clase Persona
    - test2.cpp: comprobación de la clase Socio
    - test3.cpp: comprobación de la clase Ong

  - Todos los ficheros del examen se crearán en un único directorio.
  - Puede añadir los atributos y métodos privados que considere necesarios, además de los indicados en la especificación de cada ejercicio.

  - Nota de ayuda:
    - Si no sabe hacer alguna de las funciones, al menos, declárela y déjela sin código para que el programa pueda compilar.
    - Se evaluará la parte del programa que funcione.

  - Se valorará
    - La utilización de un espacio de nombres
    - La claridad del código
    - La documentación utilizando doxygen

```cpp
// Fichero: test1.cpp
// Ficheros de cabecera
#include <iostream>
#include <string>

#include "persona.hpp"
// Espacio de nombres para la entrada y salida de datos
using namespace std;

using namespace poo;

int main(void)
{

  Persona p;

  cout << "Lectura de los datos de una persona" << endl;
  p.leerPersona();
  cout << endl;

  cout << "Escritura de los datos de una persona" << endl;
  p.escribirPersona();
  cout << endl;

  cout << endl;
  cout << "El test 1 ha finalizado correctamente" << endl << endl;

  // Fin del programa

  return 0;
}

// Fichero: test2.cpp
// Ficheros de cabecera
#include <iostream>
#include <string>

#include "socio.hpp"

// Espacio de nombres para la entrada y salida de datos
using namespace std;

using namespace poo;

int main(void)
{

  Socio s1;

  cout << "Lectura de los datos de un socio" << endl;
  s1.leerSocio();
  cout << endl;

  cout << "Escritura de los datos de un socio" << endl;
  s1.escribirSocio();
  cout << endl << endl;

  // Uso del constructor de copia

  Socio s2(s1);

  // Uso del extractor e insertador
  cout << " Datos del socio nº 2" << endl;
  cout << s2;
  cout << endl << endl;

  cout << "Modificación de los datos del socio nº 2" << endl;
  cin >> s2;
  cout << endl;

  cout << " Datos modificados del socio nº 2" << endl;
  cout << s2;
  cout << endl << endl;
```

```
    cout << "El test 2 ha finalizado correctamente" << endl << endl;

  // Fin del programa

  return 0;
}

// Fichero: test3.cpp
// Ficheros de cabecera
#include <iostream>
#include <string>

#include "ong.hpp"

// Espacio de nombres para la entrada y salida de datos
using namespace std;

using namespace poo;

int main(void)
{
  int n;

  Ong o1;

  cout << "Número de socios --> ";
  cin >> n;

  o1.setVectorSocios(n);

  cout << "Lectura de los datos de una Ong" << endl;
  o1.leerOng();
  cout << endl;

  cout << "Escritura de los datos de una Ong" << endl;
  o1.escribirOng();
  cout << endl;

  cout << "Comprobación del operador []" << endl;
  n = o1.getNumeroSocios();

  if (n > 0)
   {
     cout << "El último socio es --> " << endl;
     o1[n-1].escribirSocio();
   }

  cout << endl;

   //////////////////

   Ong o2;

  // Uso del operador de asignación =
   o2 = o1;

  cout << "Escritura de los datos de la nueva Ong 2" << endl;
  o2.escribirOng();
  cout << endl;

  string nombreFichero;

  cout << "Introduzca en nombre del fichero para grabar los datos de la ONG --> ";
  cin >> nombreFichero;

  o2.grabarEnFichero(nombreFichero);

  cout << endl;
  cout << "El test 3 ha finalizado correctamente" << endl << endl;

  // Fin del programa

  return 0;
}
```