Nombre: ..... Fecha: / 10 / 2009

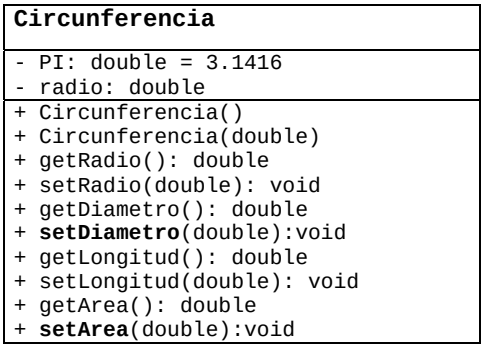
Grupo: 1 ☐ 2 ☐ 3 ☐ 4 ☐

PRÁCTICA 1
DECLARACIÓN Y DEFINICIÓN DE CLASES EN C++

En esta práctica se buscarán los siguientes objetivos. En primer lugar, aprender a traducir la sintaxis propia de los diagramas de clases en UML a C++. Para ello se debe conocer también el uso del entorno de programación Dev-C++ que será el que utilicemos durante el resto del cuatrimestre para trabajar en C++. También se pretende que el alumno se familiarice con la sintaxis propia de C++ para la declaración y definición de clases. Por último, se pretende que el alumno sea capaz de definir clientes (la función "main") que hagan uso de las clases anteriormente creadas.

La forma de conseguir los anteriores objetivos será traduciendo diversos diagramas de UML al código correspondiente en C++. Los ejemplos propios de esta práctica nos servirán en prácticas posteriores para realizar con ellos construcciones más elaboradas.

1. A partir del siguiente diagrama que especifica en UML los atributos y métodos de una clase **Circunferencia**, crea dos ficheros *Circunferencia.h* y *Circunferencia.cpp* donde se declare y defina la clase anterior.



2. Crea un programa principal que actúe como cliente de la clase **Circunferencia** (guárdalo en un fichero *Principal_prac1_1.cpp*). Más concretamente, crea cuatro objetos distintos de la clase **Circunferencia**. Crea primero un objeto de la clase **Circunferencia** haciendo uso del constructor sin parámetros. En segundo lugar, un objeto de la clase **Circunferencia** cuyo radio inicial sea 7.5. En tercer lugar, un puntero a un objeto de la clase **Circunferencia** que se construya haciendo uso del constructor sin parámetros. Por último, crea un puntero a un objeto de la clase **Circunferencia** cuyo radio inicial sea 4.8.

3. Utiliza los métodos de acceso de la clase para mostrar por pantalla el valor del área y la longitud de los objetos que has creado. Observa la diferencia de acceso a los métodos desde punteros ("->") y desde objetos (".").

4. Utiliza los métodos de modificación para hacer que las circunferencias pasen a tener las siguientes propiedades:

La primera circunferencia debe tener área 10

La segunda circunferencia debe tener un radio mayor en tres unidades al definido inicialmente

La tercera circunferencia debe pasar a tener una longitud 25

La cuarta circunferencia debe multiplicar su radio por 3

5. Seguimos con un ejemplo distinto. Imagina que quieres informatizar un sistema de préstamo de películas por parte de una biblioteca. Aparte de los requerimientos más

complejos del sistema, de momento nos quedamos únicamente con que hacen falta dos clases llamadas **Pelicula** y **Usuario** cuya especificación en UML sea la siguiente:

Pelicula
- titulo: string - director: string - duracion: integer
+ Pelicula() + Pelicula(string,string,integer) + getTitulo(): string + setTitulo(string): void + getDirector(): string + **setDirector**(string):void + getDuracion(): integer + setDuracion (integer): void

Usuario
- nombre: string - dni: integer
+ Usuario() + Usuario(string,integer) + getNombre(): string + setNombre(string): void + getDNI(): integer + **setDNI**(integer):void

Guarda las clases en ficheros distintos, *Pelicula.cpp*, *Pelicula.h*, *Usuario.cpp* y *Usuario.h*. La forma más sencilla de codificar en C++ los campos definidos en UML como "string" es usar un tipo "vector de caracteres" (p. ej., "char [30]").

6. Define un cliente de las clases **Película** y **Usuario** en la forma de un programa principal (guarda el fichero como *Principal_prac1_2.cpp*) y comprueba que las clases están bien programadas (es decir, se pueden crear objetos, se puede acceder a los atributos de las mismas, se pueden modificar, ...).

7. Vamos a definir en un nuevo ejemplo una clase para representar puntos en el plano. La codificación de la misma comprende dos coordenadas, una para x y otra para y, y los métodos de acceso y modificación de dicha clase. La representación en UML de la clase **Punto** sería:

Punto
- coord_x: double - coord_y: double
+ Punto() + Punto(double,double) + getX(): double + setX(double): void + getY(): double + **setY**(double):void

8. Define un cliente de dicha clase, de nuevo en forma de un programa principal (guarda el fichero como *Principal_prac1_3.cpp*), que te permita crear objetos de la case **Punto**, acceder a sus atributos, modificarlos, ...

ENTREGAR los archivos *Circunferencia.h*, *Circunferencia.cpp*, *Principal_prac1_1.cpp*, *Pelicula.h*, *Pelicula.cpp*, *Usuario.h*, *Usuario.cpp*, *Principal_prac1_2.cpp*, *Punto.h*, *Punto.cpp*, *Principal_prac1_3.cpp*. Los archivos deben contener **todos los cambios** realizados durante los ejercicios de la práctica. La entrega es a través de Belenus en el repositorio de Julio Rubio. También deben contener unas cabeceras que permitan identificarte:

/* Nombre: ___
Grupo: _____
Nombre del fichero: ____
Descripción: _____*/

Guarda una copia de los ficheros para las prácticas sucesivas

Nota: Dev-C++ es software libre disponible bajo licencia GNU. Puedes descargarlo desde la dirección http://www.bloodshed.net/