Examen Práctico de la Asignatura Programación Orientada a Objetos
Grado en Ingeniería Informática. Enero 2018

NORMAS DEL EXAMEN:

1. La corrección del examen se llevará a cabo ejecutando las pruebas unitarias proporcionadas en cada ejercicio. Si no sabes hacer algún método escribe su prototipo y déjalo vacío o devolviendo cualquier valor para que el programa compile y puedan ejecutarse el resto de los test. Ya que aunque algún test falle, puntuarán los test que estén correctos. En cambio, si el programa no compila se considerará mal todo el ejercicio y puntuará 0 puntos.
2. Si codificas una clase utilizando solamente el fichero .h, crea su correspondiente fichero .cc dejándolo vacío para que funcione bien el makefile. Fíjate en el makefile para poner los nombres adecuados a los ficheros fuente.
3. Una vez codificadas las clases y los métodos en sus correspondientes ficheros fuente, puedes compilar el primer ejercicio ejecutando make test1, el segundo ejercicio ejecutando make test2, y así sucesivamente. Ejecutando simplemente make, compila todos los ejercicios.
4. Una vez compilado, se puede ejecutar el test del ejercicio 1 ejecutado ./test1, el test del ejercicio 2 ejecutando ./test2, y así sucesivamente.
5. Toda la actividad del alumno será logueada y cualquier actividad diferente a la del uso del editor, compilador, programa unzip, depurador y programa make supondrá el suspenso total de la asignatura tanto de la teoría como de la práctica.
6. En el directorio HOME se encuentra el fichero examen.zip que debes descomprimir con unzip. Se crea un subdirectorio que se llama examen en el que debes realizar todo el examen (para la corrección no se mirará en ningún otro directorio). Crear otro directorio y/o cambiar a cualquier otro directorio supondrá el suspenso definitivo de la asignatura tanto de la teoría como de la práctica.

EJERCICIOS:

1.-La clase **Mueble** gestiona los siguientes atributos:

- id:id del Mueble (tipo entero)
- nombre: cadena con el nombre del mueble (Ej: “mesa”, “silla”, etc.)
- modelo: cadena con el nombre del mueble (Ej, “sam06”, “lev45”, etc.)
- fecha: una cadena con la fecha de alta en formato “DD/MM/AAAA” (OJO: no será necesario comprobar este formato, se admitirá cualquier cadena)
- color: una cadena que describe el color básico del mueble (“rojo”, “verde”, etc.)

Codifica los ficheros mueble.cc y mueble.h con los siguientes métodos:

a) Constructor que recibe como parámetro obligatorio el id, y parámetros opcionales el nombre, el modelo, la fecha y el color del mueble que tendrán como valor por defecto: “NO_ASIGNADO”. Si el id recibido es menor que cero, el id tomará el valor 0.

b) Observadores de cada atributo individual: getId(), getNombre(), getModelo(), getFecha() y getColor().

c) Modificador setId(): recibe un entero con el nuevo id. Si el nuevo id es mayor o igual que cero, se asignará y se devolverá verdadero. En caso contrario, no se asignará y se devolverá falso.

d) Modificadores: setNombre(), setModelo(), setFecha() y setColor(). Reciben una cadena y ninguno permitirá la asignación de una cadena vacía. Si se recibe una cadena vacía, no se modificaría el valor actual del atributo y se devolverá falso. Si no es la cadena vacía, se asignará y se devolverá verdadero (en el caso de la fecha, se admitirá como válida cualquier cadena).

e) Observador getDatos() que devuelve una cadena con la concatenación de los siguientes atributos y en este orden: nombre, modelo, fecha y color separados por un guión (“.”). (OJO: en la devolución no se usa el id).
Ejemplo de devolución de este método: “mesa-sam06-08/01/2018-rojo”

PUNTUACIÓN: 2 puntos

2.- La clase **Mesa** deriva de la clase Mueble y además incluye el atributo 'patas' que es un entero con el número de patas de la mesa. Y el atributo 'material' que es una cadena (Ej: “madera”, “hierro”, “forja”, “plástico”, etc.). Codifica el fichero mesa.cc y mesa.h con los siguientes métodos:

a) Constructor que recibe como un único parámetro con el id de la mesa. Además, el constructor debe asignar el atributo “nombre” del Mueble el valor “mesa”, el atributo “patas” el valor 0, y al atributo “material” el valor “NO_ASIGNADO”.
b) Observador para los atributos: getPatas() y getMaterial().
c) Modificadores: setPAtas(), setMaterial().
d) El método checkEstado() devuelve verdadero si todos los atributos de la mesa tienen asignado valores que no sean los valores por defecto, (es decir, los datos entero no contienen 0 y los datos de tipo string no contienen “NO_ASIGNADO”), y falso, en caso contrario.

PUNTUACIÓN: 2 puntos

3.-La clase **Pedido** gestiona una lista de mesas. Utiliza las listas de la STL de C++ y codifica el fichero pedido.h y pedido.cc de manera que cumpla la siguiente especificación:

a) Constructor que no recibe ningún parámetro ni hace nada particular.
b) addPedido() que añade la mesa recibida como parámetro a la lista.
c) size() que devuelve el número de mesas de la lista.
d) deletePedido(): recibe una cadena y borra de la lista la mesa cuyo modelo se ha recibido como parámetro y devuelve verdadero. Si no existe la mesa, no borre nada de la lista y devuelve falso.
e) deletePedido(): recibe una mesa y borra de la lista la mesa cuy modelo es igual al modelo de la mesa que se ha recibido como parámetro y devuelve verdadero.
f) write(): escribe en un fichero de texto cuyo nombre debe ser “salida.txt” la información de cada mesa de la lista en el siguiente formato:

posición, id, nombre, modelo, material
posición, id, nombre, modelo, material
posición, id, nombre, modelo, material
…

De modo que si, por ejemplo, tenemos 3 meses en la lista, el fichero “salida.txt” podría quedar de la siguiente forma (los datos son de ejemplo):

1,976,mesa,ceti99,madera
2,77,mesa,pet78,plástico
3,87,mesa,oiu87,hierro

PUNTUACIÓN: 3 puntos

4.- La clase **Punto** dispone de un entero para la coordenada X y otro para la coordenada Y de un punto en el plano. Codifica en los ficheros punto.h y punto.cc los siguientes métodos para esta clase:

a) Constructor con los valoresi niciales de X e Y como parámetros obligatorios.
b) Observador getX() y getY() para cada coordenada del punto.
c) Sobrecarga las operadores =, +, y ++ (tanto prefijo como sufijo).
d) Operador + que devuelve un objeto de tipo Punto con la suma entre un objeto de tipo Punto y un entero (en ese orden). El punto devuelto tiene las coordenadas iguales al punto que hace

de sumando incrementadas cada una el valor del entero.
Ejemplo p + 3

e) Operador + que devuelve un objeto de tipo Punto con la suma entre un entero y un objeto de tipo Punto (en ese orden). El punto devuelto tiene las coordenadas iguales al punto que hace de sumando incrementadas cada una el valor del entero.
Ejemplo p + 3

PUNTUACIÓN: 3 puntos