Examen Práctico de la Asignatura Programación Orientada a Objetos
Grado en Ingeniería Informática. Convocatoria: Diciembrede 2014

NOTAS IMPORTANTES:

A) La corrección del examen se llevará a cabo ejecutando las pruebas unitarias proporcionados en cada ejercicio. Si no sabes hacer un método de una clase, escribe su prototipo y déjalo vacío para que el programa compile y puedan ejecutarse los tests (aunque algún test falle, sí puntuarán los test válidos). Ya que si el programa no compila se considerará MAL todo el ejercicio y puntuará 0 PUNTOS.

B) Si codificas una clase utilizando solamente el fichero .h, crea su correspondiente fichero .cc dejándolo vacío.

C) 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 tests de todos los ejercicios.

D) Una vez compilado, se puede ejecutar el test del ejercicio 1 ejecutado `test1`, el test del ejercicio 2 ejecutando `test2`, y así sucesivamente

E) 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.

F) 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 de la asignatura tanto de la teoría como de la práctica.

EJERCICIOS

1. La clase Dado gestiona el lanzamiento de 1 dado. Codifica la clase Dado en C++ con los siguientes métodos:

a) El constructor recibe 1 parámetro opcional con el valor inicial para el dado. Si no se recibe el parámetro el valor inicial se asignara a 1. Control de errores: si el valor inicial no es correcto, el valor asignado será igual a 1.

b) Observador get(). Devuelve el valor del dado (función inline).

c) Modificador set(). Recibe un parámetro con el valor entero a asignar al dado. Control de errores: si se produce algún error en el valor asignado (debe estar entre 1 y 6), set() debe devolver false y no modificar el dado, en caso contrario set() modifica el dado y devuelve true.

d) Sobrecarga del operador + para que sume el valor de dos dados y **devuelva un entero** con el resultado.

NOTA: La declaración de la clase debe estar en el fichero dado.h
El cuerpo del constructor, el método set y el operador deben estar en el fichero dado.cc
Puedes compilar el ejercicio ejecutando: `make test1`
Puedes ejecutar los test unitarios ejecutando: `./test1`

2. La clase Dados gestiona el lanzamiento de 2 dados. Codifica la clase Dados en C++ con los siguientes métodos:

a) El constructor recibe dos parámetros opcionales con el valor inicial para el dado 1 y el dado 2 respectivamente. Si los dados no reciben valor inicial, éste se asignara a 1. Control de errores: si algún valor inicial no es correcto, el valor asignado será igual a 1.

b) Observador get(). Tendrá 2 parámetros. El identificador del dado (1 o 2) y una referencia a un entero que almacenará el valor del dado correspondiente. Control de errores: si se produce algún error con el identificador del dado, get() debe devolver false, en caso contrario get() devuelve true.

c) Modificador set(). Tendrá 2 parámetros. Un entero con el identificador del dado que queremos

asignar y el valor asignado. Control de errores: si se produce algún error en el identificador del dado (debe ser 1 o 2) o en el valor asignado (debe estar entre 1 y 6), set() debe devolver false y no modificar el dado, en caso contrario set() modifica el dado correspondiente y devuelve true.

d) Método suma(). Devuelve la suma del valor de los 2 dados.

NOTA: escribe la declaración de las funciones en el fichero dados.h y el cuerpo en dados.cc
Puedes compilar el ejercicio ejecutando: `make test2`
Puedes ejecutar los test unitarios ejecutando: `./test2`

3. La clase Persona gestiona un string con el nombre de una persona y tiene los siguientes métodos:

a) Constructor que recibe el nombre de la persona
b) Observador, getNombre(), que devuelve el nombre de la persona
c) Modificador, setNombre(), del nombre de la persona.

La clase Alumno deriva de la clase Persona y gestiona un entero con el curso del alumno. Tiene los siguientes métodos:

a) Constructor que recibe el nombre del alumno y el curso como parámetros obligatorios. Si el curso proporcionado no está entre 1 y 4, el curso debe asignarse al valor 1.
b) Observador del curso getCurso()
c) Modificador del curso setCurso() que recibe como parámetro un entero con el curso del alumno. Si el valor del parámetro no está entre 1 y 4 devolverá false y no modificará el curso, y en caso contrario devolverá true y sí modificará el curso.

NOTA: Guarda el código en los ficheros persona.h, persona.cc, alumno.h y alumno.cc incluso aunque dejes vacíos los archivos .cc
Puedes compilar el ejercicio ejecutando: `make test3`
Puedes ejecutar los test unitarios ejecutando: `./test3`

4. Realiza una copia del fichero dado.h del ejercicio anterior que se llame dado2.h y realiza una copia del fichero dado.cc del ejercicio anterior al fichero dado2.cc
Añade los siguientes métodos a la clase Dado del ejercicio 1:

a) Método launchs(), devuelve el número de veces que se ha asignado un valor al dado.
b) Método get(), recibe el parámetro i y el parámetro x. Pone en x el i-esimo valor anteriormente asignado al dado con un límite de 5.
   Por ejemplo si al dado se le han asignado sucesivamente los valores [2,5,1,3,5]; get(0) pone en x el valor actual del dado, get(1) el valor 5, get(2) el valor 3 y así sucesivamente hasta un máximo de get(5) que es 2. Además, si i está dentro del rango el método devuelve true, en caso contrario el método devuelve false. Cuando aún no se han producido lanzamientos suficientes, los valores anteriores serán igual a 0.

NOTA: escribe la declaración de las funciones en el fichero dado2.h y el cuerpo en dado2.cc
(CUIDADO CON MACHACAR LOS FICHEROS DEL EJERCICIO 1)
Puedes compilar el ejercicio ejecutando: `make test4`
Puedes ejecutar los test unitarios ejecutando: `./test4`