Programación Orientada a Objetos
Práctica 2

Juan A. Romero 2014, aromero@uco.es

LEER HASTA EL FINAL ANTES DE COMENZAR EL EJERCICIO

La clase Persona gestiona el DNI, nombre, apellidos, dirección, localidad, provincia y país de una persona (todos datos de tipo string).

1. Crear modificadores (set) y observadores (get) para cada dato.
2. Un constructor que recibe como parámetro de forma obligatoria el DNI, y de forma opcional el resto de datos con un valor por defecto igual a "" (cadena vacía).
3. Un método getApellidosyNombre() que devuelve una cadena con el formato: "apellidos, nombre". Usar la concatenación (operador +) de la clase string.

Los test para esta clase se encuentran en el fichero `persona_unittest.cc` que se proporciona y que tiene los siguientes tests:

a) test del constructor con parámetros por defecto
b) test del constructor con parámetros obligatorios
c) test del constructor de copia (se usará el constructor de copia por defecto)
d) test del operador = (se usará el operador = por defecto)

Codificar la clase Persona de forma que pase dichos tests.
Es importante observar que el constructor de copia por defecto y el operador = por defecto que proporciona C++ es suficiente para el caso de la clase persona. Analizar esta cuestión. ¿Por qué no es necesario definir un constructor de copia a medida en este caso?

La clase Crupier hereda de la clase *Persona* y guarda un código alfanumérico de empleado del casino. El constructor debe recibir forzosamente DNI y código de empleado, y el resto de parámetros tienen un valor por defecto igua a "". Tener en cuenta que se le deben pasar los parámetros correspondientes al constructor de la clase base (*Persona*) mediante iniciadores de la clase base.
Añadir también los métodos getCodigo() y setCodigo().
Realizar un test unitario en el fichero `crupier_unittest.cc` análogo al de la clase Persona pero ampliado para la clase *Crupier*, es decir, con test análogos e incluyendo tests para el código del empleado y un test diferente para cada uno de sus métodos.
Del mismo modo, es importante observar que el constructor de copia por defecto y operador = por defecto de C++ son también suficientes en este caso.

La clase Jugador hereda de la clase *Persona*, tiene un dinero en euros para apostar (dinero_, de tipo int, no se admiten fracciones de euro en las apuestas) y tiene un código alfanumérico de jugador. Además, cada jugador tiene una lista de apuestas. Cada elemento de la lista guarda el tipo de apuesta, el valor de la apuesta y la cantidad apostada. Los tipos de apuesta son:

a) Tipo 1, apuesta sencilla. Se apuesta a un número entre 0 y 36, y si sale, se gana 35 a 1 (se puede apostar y ganar al 0).
b) Tipo 2, apuesta rojo o negro. Se apuesta a un color y se paga 1 a 1. Si sale el cero, se pierde.
c) Tipo 3, apuesta par o impar. Se apuesta par o impar y se paga 1 a 1. Si sale el cero, se pierde.
d) Tipo 4, apuesta alto o bajo. Bajo es entre 1 y 18; alto entre 19 y 36. Se paga 1 a 1. Si sale el cero, se pierde.

La clase Jugador debe cumplir los siguientes requisitos:

1. El constructor debe recibir forzosamente DNI y código de jugador, el resto de parámetros tienen un valor por defecto igua a “”. El dinero debe ser inicializado siempre a 1000.
2. Observadores y modificadores para código de jugador (get/setCodigo())y dinero (get/setDinero()).
3. La lista de apuestas debe ser dinámica usando la clase *list* de la STL de C++ (C++ STL *list*) donde cada elemento de la lista guarda los tres datos antes mencionados: tipo, valor y cantidad.
4. Un método, *getApuestas()*, que devuelve la lista de apuestas (también podría hacerse pasando como parámetro una referencia a una lista de apuestas a la que se le asignan las apuestas del jugador, pero no es necesario ya que un objeto *list* puede devolverse en una función y se copia bien sin problemas en otro objeto de tipo *list* que reciba el valor devuelto).
5. Un método, *setApuestas()*, que borra las apuestas actuales y lee del fichero DNI.txt las nuevas apuestas, siendo DNI el DNI del jugador. El fichero tiene formato texto y la siguiente estructura:
   CÓDIGO-APUESTA,VALOR,CANTIDAD
   CÓDIGO-APUESTA,VALOR,CANTIDAD
   …
   CÓDIGO-APUESTA,VALOR,CANTIDAD
   Usar un editor de texto plano para crear un fichero de texto con algunas apuestas siguiendo el formato descrito.

Hacer tests para probar la clase *Jugador* en el fichero `jugador_unittest.cc` y un pequeño programa principal (`jugador-ppal.cc`) que pida los datos de un jugador, lea del fichero de texto correspondiente sus apuestas y las muestre por pantalla. En realidad, si se diseñan bien los test, el programa principal de prueba no sería necesario (analizar esta afirmación).
Para hacer la lista REVISAR ANTES el ejemplo de STL list proporcionado en la web de la asignatura.
Igualmente, para el manejo de ficheros en C++ REVISAR ANTES los ejemplos y documentación aportada en la web de la asignatura sobre ficheros en C++ y la función getline() para leer un fichero texto con delimitadores, como es el caso del fichero del apartado 5 de la práctica.

NOTA:
Todos los ejercicios deben hacerse con los estándares de calidad que estamos aprendiendo en las clases de teoría (esto es más importante que la corrección del ejercicio en sí).