



**1.- (test 1)** La clase Tool representa una herramienta. Cada herramienta tiene un nombre, su fabricante, su 'id' en los campos 'name', 'maker' e 'id' respectivamente (todos de tipo string).

Una herramienta también tiene un número de existencias en almacén 'existences' de tipo entero y un 'rating' de tipo int que indica la valoración (entre 0 y 5) por parte de los clientes.

Codifica los ficheros tool.h y tool.cc para la clase Tool con la siguiente funcionalidad:

- a) Constructor con 3 parámetros.  
El primer parámetro será el id que debe ser obligatorio.  
El segundo parámetro será el nombre que tendrá como valor por defecto 'no-name'.  
El tercer parámetro será el fabricante y tendrá como valor por defecto 'unknown' controlando los siguientes errores: si el 'id' llega como la cadena vacía, deberá iniciarse a la cadena "error", si nombre o el fabricante llega como una cadena vacía, deberán ponerse a su valor por defecto mencionado. El constructor también inicia el valor de 'rating' a 5 y 'existences' a 0.
- b) Modificador SetId() que recibe como parámetro el nuevo 'id'. Si el nuevo 'id' es la cadena no se hará el cambio y se devolverá false. En caso contrario, se hará el cambio y se devolverá true.
- c) Modificador SetName() que recibe como parámetro el nuevo nombre que si es la cadena no se hará el cambio y se devolverá false. En caso contrario, se hará el cambio y se devolverá true.
- d) Modificador SetMaker() que recibe como parámetro el nuevo fabricante que si es la cadena vacía no se hará el cambio y se devolverá false. En caso contrario, se hará el cambio devolverá true.
- e) Modificador SetExistences() que recibe como parámetro la actualización de las existencia si es un valor menor que cero, no se hará el cambio y se devolverá false. En caso contrario aplicará el cambio y se devolverá true
- f) Modificador SetRating() que recibe como parámetro el nuevo 'rating'. Si el nuevo 'rating' número fuera del rango [0,5], no se hará el cambio y se devolverá false. En caso contrario se hará el cambio y se devolverá true.
- g) Observadores GetId(), GetName(), GetMaker(), GetRating() y GetExistences().
- h) Método GetStatus() que devuelve true si 'id' está asignado con un valor diferente de 'empty'/'name' a un valor diferente de 'no-name' y 'maker' a un valor diferente de 'unknown'. En contrario, devuelve false.

PUNTUACIÓN: 1 punto

**2.-(test2)** La clase Hammer representa un martillo y hereda de la clase Tool del ejercicio anterior. La clase Hammer gestiona además el dato 'madeof' de tipo string que indica el material del que está fabricado.

La clase Drill representa un taladro y también hereda de la clase Tool, Además, la clase Drill tiene un dato 'power' con la potencia en vatios de tipo int.



Escribe el código de la clase Hammer (ficheros hammer.h y hammer.cc) y de la clase Drill (ficheros drill.h y drill.cc) con la siguiente funcionalidad:

- a) Clase Hammer, Constructor que recibe como parámetro obligatorio el id' de la clase Tool. Opcionalmente podrá recibir como parámetro el material del que está fabricado. Si este parámetro no se envía o llega como una cadena vacía se asignará por defecto el valor "error".
- b) Clase Hammer. Modificador SetMadeofO que recibe como parámetro el nuevo 'madeof'. Si el nuevo 'madeof' es la cadena vacía, no se hará el cambio y se devolverá false. En caso contrario, se hará el cambio y se devolverá true.
- c) Clase Hammer. Observador GetMadeof().
- d) Clase Drill. Constructor para la clase Drill que recibe como parámetro obligatorio el id' de la clase Tool. Opcionalmente podrá recibir la potencia en vatios. Si este parámetro no se envía se asignará por defecto el valor 1. Si el valor recibido es menor que 1, se asignará al valor 0.
- e) Clase Drill. Modificador SetPower() que recibe como parámetro el nuevo 'power'. Si el nuevo "power" es menor que 1, no se hará el cambio y se devolverá falso. En caso contrario, se hará el cambio y se devolverá verdadero.
- f) Clase Drill. Observador GetPower().

PUNTUACIÓN: 2 puntos

**3.-(test3)** La clase Set representa un juego de herramientas y tendrá dos atributos de tipo string: un identificador 'id' y un nombre 'name'. La clase Ser dispone de un vector de objetos de tipo Tool (clase Tool del ejercicio 1) denominado 'vt'. Implementa este vector con la clase vector de la STL de C++.

Codifica los ficheros set.h y set.cc con las siguientes funcionalidades:

- a) Constructor que recibe como parámetros obligatorios el identificador id' y su nombre 'name'.
- b) Observadores Getid() y GetNameO
- c) Observador GetTotalTools() que devuelve el número total de herramientas.
- d) El método AddTool() que recibe un objeto de tipo Tool. Si no existe un objeto con el mismo identificador (id), lo añade y devuelve true. Si ya existe un objeto con el mismo identificador, no lo añade y devuelve false.
- e) El método RemoveTool( que recibe un id' y elimina la Tool correspondiente del vector. Si no existe ningún objeto con ese identificador, no hace nada y devuelve false. Si lo encuentra, lo elimina y devuelve true.
- f) El método GetBestO que devuelve el número de tools con rating mayor o igual a 4. El método PrintCatalog() de tipo void, que muestra en pantalla la información de todos los objetos del vector. Una línea por cada Tool. La salida comienza con un salto de línea en blanco. Después, cada línea debe mostrar: la posición de esa Tool en el vector seguido de una "", seguido del id, seguido de dos puntos, seguido del nombre, una coma, el fabricante, una coma y su rating. Después del listado, haz un salto de línea e incluye una nueva



línea que comience por "BEST:" seguido del número de herramientas con rating mayor o igual a 4, un salto de línea y su listado a partir de la siguiente línea. Para verlo mejor, mira el siguiente ejemplo. Para que el apartado se considere correcto, al ejecutar el test PrintCatalog del test3 se debe mostrar en pantalla:

```
0/id1: name1, maker1,2
1/id2: name2, maker2,4
2/id3: name3, maker3,3
3/id4: name4, maker4,5
BEST: 2
1/id2: name2, maker2,4
3/id4: name4, maker4,5
```

PUNTUACIÓN: 3 puntos

**4.- (test4)** La clase Pareja tiene 2 datos de tipo double 'a' y 'b':

Codifica en los ficheros pareja.h y pareja.cc los siguientes métodos para esta clase:

- a) Constructor con los valores iniciales de a y b como parámetros obligatorios sin valores por defecto.
- b) Observadores GetAO y GetB(.
- c) Sobrecarga los operadores + y - (suma o resta por componentes como si fuesen componentes d un vector o unas coordenadas).
- d) Sobrecarga del operador >> (extractor) que saque en pantalla los valores de a y b entre corchetes y separados por una coma sin espacios en blanco. Ejemplo: [2.3,5.7]
- e) Sobrecarga del operador << (insertador) que pida por teclado a y b de la siguiente forma (saldrá en pantalla en la ejecución del test4):

*Introduce a: (el usuario introduce el valor de 'a' por teclado)*

*Introduce b: (el usuario introduce el valor de 'b' por teclado)*

**NOTA:** En la ejecución del test4 debe salir en pantalla [2.3,5.7] y a continuación pide la introducción por teclado de a y de b, como se dice en el enunciado del ejercicio. Y debe salir en pantalla lo que se ha introducido entre corchetes.

PUNTUACIÓN: 2 puntos

**5.-(test5)** Escribe utilizando únicamente el fichero calculator.cc la plantilla de clase Calculator que utilizará como tipo base T de forma que podamos hacer las siguientes definiciones:

```
Calculator<int> intCalc;
```

```
Calculator<double> doubleCalc;
```

Añadir la siguiente funcionalidad:

**605 45 70 99 (Ángel)**

Únicos especialistas en Grado en Ingeniería Informática



1. Función Div que recibirá como primer parámetro el numerador y como segundo parámetro el denominador devolviendo la división entre ambos.

PUNTUACIÓN: 2 puntos

[www.academiamain.es](http://www.academiamain.es)