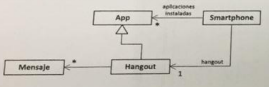
Deusto
Facultad de Ingeniería

0. LOS PRELIMINARES...

Se parte de la clase **`Mensaje`** *ya creada (con sus correspondientes ficheros de cabecera .h y de implementación .cpp) y un fichero* ***"main.cpp"*** *que contiene el programa principal. Este programa crea una instancia de la clase* **`Mensaje`** *e imprime por pantalla sus datos. Antes de comenzar el examen, prueba que este mínimo programa funciona correctamente de modo que estés seguro que el entorno de desarrollo se encuentra bien configurado.*

Una vez comprobado esto, el objetivo del examen es implementar parte de un programa que administra las aplicaciones que se instalan en distintos teléfonos móviles. Tras la fase de diseño se ha llegado a la conclusión de que el dominio mencionado requiere de las clases **`Smartphone`**, **`App`**, **`Hangout`** *y* **`Mensaje`**, *con las siguientes relaciones:*



1. CLASE APP (0,25 puntos)

1.1 App. *Crea una nueva clase* **`App`** *(con su correspondiente fichero de cabecera y de implementación) que represente a cualquier aplicación que pueda ser instalada en un teléfono móvil.*

1.2 Atributos de la clase. *Esta clase deberá contener dos atributos: el nombre de la aplicación y su versión (por ejemplo, "Twitter 6.4"). Adicionalmente debe declararse un atributo estático privado que indique a partir de qué número de versión cualquier aplicación del sistema será suficientemente vieja para ser considerada obsoleta. Asumimos en este examen que existe un número de versión común de obsolescencia para todas las aplicaciones.*

1.3 Métodos de la clase. *Añade en la clase* **`App`**, *al menos, los siguientes métodos:*

- **`constructores y métodos get/set`**: incluye aquellos básicos que vayas viendo que necesitas durante el examen para la inicialización y acceso a las instancias de esta clase.
- **`void info()`**: su cometido es visualizar por pantalla el nombre y la versión de la aplicación. En el apartado 4.5 tienes varios ejemplos del formato que se espera en esta visualización.

2. CLASE HANGOUT (0,75 puntos)

2.1 Hangout. *Crea una nueva clase* **`Hangout`** *(con su fichero de cabecera y de implementación) que represente a la aplicación concreta de Hangout. Esta clase deberá* ***heredar*** *de la clase* **`App`**.

2.2 Atributos de la clase. *Además de los atributos heredados, deberá incluir el conjunto de mensajes que han sido producidos por esa aplicación de Hangout (deberás hacer uso de la clase* **`Mensaje`** *proporcionada). Asumimos que, como mucho, la aplicación únicamente mantendrá almacenados los 5 mensajes más recientes generados.*

2.3 Métodos de la clase. *Añade en la clase* **`Hangout`**, *al menos, los siguientes métodos:*

Deusto
Facultad de Ingeniería

- **`constructores y métodos get/set`**: incluye aquellos básicos que vayas viendo que necesitas durante el examen para la inicialización y acceso a las instancias de esta clase.
- **`void chatear(const char*, const char*)`**: crea un nuevo mensaje de chat a través de Hangout, cuyo emisor es el nombre que aparece como primer parámetro y su texto el segundo. Deberás hacer uso de la clase **`Mensaje`** dentro de este método y recuerda que el código deberá almacenar únicamente los 5 mensajes más recientes producidos, descartando los más antiguos.
- **`void info()`**: debe redefinir el método correspondiente de la clase **`App`**. Su cometido es visualizar por pantalla el nombre de la aplicación, su versión y cada uno de los mensajes producidos (sólo los 5 últimos). En el apartado 4.5 tienes un ejemplo del formato deseado. Se espera que reutilices la funcionalidad común que está implementada en el método equivalente de la clase padre y no olvides que la clase **`Mensaje`** ya tiene un método para imprimir.

3. CLASE SMARTPHONE (1,5 puntos)

3.1 Smartphone. Crea una nueva clase **`Smartphone`** (con su fichero de cabecera y de implementación) que represente a un teléfono móvil en el que instalar y gestionar aplicaciones.

3.2 Atributos de la clase. Además del nombre del modelo del móvil (por ejemplo, "iPhone S6"), deberá contener el conjunto de aplicaciones instaladas. Cada móvil tendrá un atributo más (un entero) que es el tamaño de la memoria y que determina el número máximo de aplicaciones que puede tener instaladas. Por ejemplo, si su valor es 3, el Smartphone sólo puede tener tres aplicaciones instaladas, como máximo. Finalmente, de entre el conjunto de aplicaciones instaladas, cada Smartphone tendrá un puntero a la aplicación de Hangout (si es que estuviera instalada). Pues será esta aplicación la que le permitirá enviar mensajes de chat.

3.3 Métodos de la clase. Añade en la clase **`Smartphone`**, al menos, los siguientes métodos:

- **`constructores y métodos get/set`**: incluye aquellos básicos que vayas viendo que necesitas durante el examen para la inicialización y acceso a las instancias de esta clase. **Ayuda:** al crear un nuevo Smartphone, se indicará su modelo y el tamaño de su memoria.
- **`void instalarApp(<App>)`**: añade una nueva aplicación al conjunto de aplicaciones instaladas en el Smartphone. En caso de que se haya superado el número de aplicaciones máximo permitido por la memoria del Smartphone, la aplicación no se instalará y se visualizará el mensaje "No suficiente memoria para instalar la app".
- **`void desinstalarApp(char*)`**: elimina del conjunto de aplicaciones instaladas en el Smartphone aquella cuyo nombre coincida con el pasado como parámetro. Implicará que en el Smartphone hay hueco para una aplicación más.
- **`void ampliarMemoria(int)`**: reconfigura el número máximo de aplicaciones que pueden instaladas (asumimos que siempre se incrementa). **Ayuda:** implicará reservar memoria para nuevo número de aplicaciones y reubicar las que ya estaban instaladas en la nueva memor
- **`void setHangout(<Hangout>)`**: asigna la aplicación de Hangout pasada como parám como la aplicación a usar por ese Smartphone para el envío de mensajes de chat.

Deusto
Facultad de Ingeniería

- **`void chatear(const char*, const char*)`**: genera un mensaje de chat que tiene como *emisor y texto el contenido del primer y segundo parámetro, respectivamente. Debes usar la aplicación de Hangout instalada en el Smartphone para el envío del mensaje. En caso de que no esté la aplicación instalada, el método visualizará el mensaje "Hangout no disponible".*
- **`void info()`**: *visualiza por pantalla toda la información contenida en ese Smartphone. En este caso sería su modelo, el tamaño de su memoria (número máximo de aplicaciones instalables) y la información de cada una de sus aplicaciones instaladas. En el apartado 4.5 tienes un ejemplo del formato e información a visualizar.*
- **`void buscarAppObsoletas(<numero-apps>, <array-apps>)`**: *este método devuelve como segundo parámetro el conjunto de aplicaciones instaladas en el Smartphone que consideramos obsoletas (aquellas cuyo número de versión es inferior al almacenado en el atributo estático con la versión obsoleta de la clase* **`App`**). *Además, el método deja en la variable pasada como primer parámetro el número de aplicaciones encontradas.*

NOTA: *los parámetros representados como <TYPE> para las distintas funciones a implementar son orientativos, es decir, no se detalla en ningún caso si éstos deben ser punteros, clases, referencias, colecciones,... Será el alumno quien deba decidir y actuar en consecuencia.*

4. PROGRAMA PRINCIPAL (0,5 puntos)

Partir del fichero ***"main.cpp"****, e ir incorporando el código necesario para cumplir con esta funcionalidad.*

4.1 Crear un Smartphone y cuatro aplicaciones. *Crear un Smartphone del modelo "Samsung Galaxy S7" y con tamaño de memoria igual a 1 (solo podrás instalar una aplicación). Después crear cuatro aplicaciones, con los siguientes nombres y versiones: Hangout 1.7, Twitter 5.1, Calc 1.5 y Calendar 0.4. Dos de ellas debes crearlas en memoria estática y otras dos reservando memoria dinámica.*

4.2 Instalar las aplicaciones en el Smartphone. *Intenta instalar las 4 aplicaciones en el Smartphone. Al intentar instalar la segunda, debería generarse el siguiente mensaje de error:*

```
No suficiente memoria para instalar la app.
```

Amplía la memoria del Smartphone hasta 5 aplicaciones. Después termina de instalar las aplicaciones hasta tener las cuatro instaladas. Hacer notar que la primera de las aplicaciones era un Hangout y por tanto debes asignarlo al Smarphone (con el método **`setHangout`**) *para el envío posterior de mensajes.*

4.3 Enviar mensajes de chat con el Smartphone. *Enviar 6 mensajes de chat usando la aplicación de Hangout instalada en el Smartphone. El contenido de los mensajes debería ser el siguiente:*

- [Emisor: Pedro, Texto: ¡Hola!]
- [Emisor: Maria, Texto: ¿Que tal?]
- [Emisor: Maria, Texto: ¡Cuanto tiempo!]
- [Emisor: Pedro, Texto: ¡Muy bien!]
- [Emisor: Pedro, Texto: ¿Y tu?]
- [Emisor: Maria, Texto: Haciendo un examen...]

4.4 Desinstalar una aplicación. Desinstalar la aplicación con el nombre "Calendar" del Smartphone.

Deusto
Facultad de Ingeniería

4.5 Mostrar los datos del Smartphone. Usando el método **`info`** del Smartphone debería visualizarse la siguiente información (el orden en que se muestran las aplicaciones es irrelevante y hacer notar que sólo se muestran los 5 últimos mensajes de Hangout, de los 6 que había):

```
SMARTPHONE: Samsung Galaxy S7
MEMORIA: 5 aplicaciones
APPS INSTALADAS:
------
NOMBRE APP: Hangout
VERSION: 1.7
[Emisor: Maria, Texto: ¿Que tal?]
[Emisor: Maria, Texto: ¡Cuanto tiempo!]
[Emisor: Pedro, Texto: ¡Muy bien!]
[Emisor: Pedro, Texto: ¿Y tu?]
[Emisor: Maria, Texto: Haciendo un examen...]
------
NOMBRE APP: Calc
VERSION: 1.5
------
NOMBRE APP: Twitter
VERSION: 5.1
```

4.6 Listar aplicaciones obsoletas. Modificar desde el programa principal el atributo estático de la clase **`App`** que marca el umbral de las versiones obsoletas para que contenga el valor 2.0. Posteriormente, usar el método **`buscarAppObsoletas`** para encontrar las aplicaciones obsoletas y visualizarlas por pantalla. El resultado sería (nuevamente el orden de las aplicaciones es irrelevante):

```
APPS OBSOLETAS:
Hangout (v1.7)
Calc (v1.5)
```