Practica 1. UML Diagrama de clases.

Objetivo: Usando la herramienta de código libre Umbrello, diseñar mediante diagrama de clases UML una interfaz de logeo de mensajes de diferentes prioridades, usando caracterrísticas del lenguaje de diseño de clases UML como lo son las clases, la herencia y las interfaces.

Procedimiento:

1. Comenzar con clase base “StreamEditor”, el cual sera una interfaz con los métodos:

- int open(const string)
  - Descripción: Abre archivo para escritura / lectura. Regresa descriptor de archivo. Establece fichero activo.

- void close(int fd)
  - Descrpción: Cierra el fichero con el descriptor asociado “fd”. En caso de no proporcionar ningún parámetro, cierra el archivo activo (único).

- int write(const string data)
  - Descripción: Escribe cadena ASCII en el archivo activo.

- String read(void)
  - Descripción: Lee una línea ASCII del archivo activo.

2. Seguir con la clase “Logger” la cual implementará lainterfaz “StreamEditor” y contará con los siguientes metodos:

- int open(const string data) overrides
  - Descripción: Implementa método de clase padre. En caso de no contar con ningún argumento, se deberá abrir un archivo predeterminado.
  -
- int write(const string data) overrides
  - Descripción: Implementa método de clase padre. Escribe en el fichero activo. En caso de no contar con ningún archivo previamente abierto, se deberá abrir el archivo predeterminado.

- Int read(const string) overrides
  - Descripción: Implementa método de clase padre.Lee una línea del fichero activo.

- int log(string logData)
  - Descripción: Ecribe mensaje de depuración en el fichero activo. Agrega el tiempo en formato hh:mm:ss:msec al inicio de la linea.

3. Seguir con las clases que heredan de “Logger”, las cuales serán: “DebugLogger”, “WarningLogger” y “ErrorLogger”, las cuales registrarán mensajes de depuración en orden de severidad ascendente, y añadirán los siguientes métodos:

- int debugLog(string logData)
  - Descripción: Escribe mensaje de depuración en el fichero activo, con prioridad baja. Llama el método “log” de la clase padre, agregando la cadena “Debug: ”.

- int warningLog(string logData)
  - Descripción: Escribe mensaje de advertencia en el fichero activo, prioridad intermedia. Llama al método “log” de la clase padre, agregando la cadena “Warning: ”

- int errorLog(string logData)
  - Descripción: Escribe mensaje de advertencia en el fichero activo, prioridad intermedia. Llama al método “log” de la clase padre, agregando la cadena “Error: ”

Además de las implemetaciones de métodos anteriores, las clases diseñadas podrán contar con miembros, constructores / destructores, a consideración del programador.

Posterior al diseño en UML de las clases anteriores, se deberá exportar el diseño en lenguaje C++, se deberá hacer la implementación manual y finalmente elaborar el makefile para la compilación de las clases una vez terminada la implementación.

Evaluación:

Para el propósito de la evaluación, se deberá correr el programa de prueba sustituyendo el nombre del archivo “defaultFile” por uno con el nombre del equipo. El código fuente, junto con el archivo generado y un archivo de texto plano con el nombre de los integrantes del equipo, deberá “subirse” al “branch” designado por el maestro usando Subversion o GIT.