```cpp
/*
 * Calculateur_Li
 *
 * Une classe pour réaliser le calcul de la valeur Li en
 * utilisant la sensibilité et le gain du capteur Electret
 * MAX4466.
 *
 * Voir les notes de cours "Conception des objets (IIB)" pour les
 * calculs à effectuer.
 *
 * Note: Cette classe contient un objet de classe
 *       Calculateur_VRMS pour calculer la valeur dBV du signal
 *       échantillonné.
 *
 * Convention:
 *  Variables -> camelCase
 *  Classes, fonctions -> PascalCase
 *  Constantes, types utilisateurs -> SNAKE_CASE
 *
 * GPA788 - ETS
 * T. Wong
 * 09-2018
 * 08-2020
 */
#ifndef CALCULATEUR_LI_H
#define CALCULATEUR_LI_H

// Pour pouvoir utiliser un objet de type Calculateur_VRMS
#include "calculateur_vrms.h"

class Calculateur_Li {
public:
  // Pour le microphone Electret une application de 94 dB SPL
  // produit -44 dBV/Pa à sa sortie. Le gain du MAX4466 est par
  // défaut réglé à 125 ou 42 dBV.
  Calculateur_Li(double P = 94.0, double M = -44, double G = 52.0)
    : m_P(P), m_M(M), m_G(G)
  {
    mNb_li = 0;
  }
    // Empêcher l'utilisation du constructeur de copie
  Calculateur_Li(const Calculateur_Li& other) = delete;
  // Empêcher l'utilisation de l'opérateur d'assignation
  Calculateur_Li& operator=(const Calculateur_Li& other) = delete;
  // Empêcher l'utilisation du constructeur par déplacement
  Calculateur_Li(Calculateur_Li&& other) = delete;
  // Empêcher l'utilisation de l'opérateur de déplacement
  Calculateur_Li& operator=(Calculateur_Li&& other) = delete;

  ~Calculateur_Li() = default;  // Destructeur

  /* -------------------------------------------------------------
     Accesseurs des données membres de la classe
     ------------------------------------------------------------- */
  inline double GetLi() const { return m_Li; }
  inline double GetP() const { return m_P; }
  inline double GetM() const { return m_M; }
  inline double GetG() const { return m_G; }
  inline uint32_t GetNbLi() const { return mNb_li;}
```

```
60
61    inline uint16_t GetNbSamples() const { return c.GetnbSamples(); }
62    inline uint16_t GetTotalSamples() const { return c.GetTotalSamples(); }
63    inline double GetVrms() const { return c.GetVrms(); }
64    inline double GetdBV() const { return c.GetdBV(); }
65    inline uint8_t GetAPin() const { return c.GetAPin(); }
66    inline double GetVMax() const { return c.GetVMax(); }
67    inline int16_t GetAdcMax() const { return c.GetAdcMax(); }
68
69    inline void SetAPin(uint8_t A_Pin){ c.SetAPin(A_Pin); }
70    inline void ResetNbLi(){mNb_li = 0;}
71
72    /* -------------------------------------------------------------
73       Services publics offerts
74       ------------------------------------------------------------- */
75    // Utiliser Accumulate() de l'objet de classe Calculateur_VRMS
76    // pour accumuler les valeurs du capteur sonore.
77    // Note: La temporisation est la responsabilité de l'utilisateur.
78    void Accumulate() {
79      c.Accumulate();
80    }
81    // Utiliser Compute() de l'objet de classe Calculateur_VRMS
82    // pour calculer la valeur rms du signal sonore et ensuite
83    // calculer Li du signal.
84    // Note: La temporisation est la responsabilité de l'utilisateur.
85    double Compute() {
86      c.Compute();
87      m_Li = GetdBV() + m_P - m_M - m_G;
88      mNb_li++;
89      return m_Li;
90    }
91
92  private:
93    // Objet de classe Calculateur_VRMS pour réaliser les calculs
94    // Vrms et dBV du signal échantillonné.
95    // La relation entre la classe Calculateur_VRMS et la classe
96    // Calculateur_Li est une relation de "composition".
97    Calculateur_VRMS c;
98    // Pour le calcul de Li
99    double m_Li;                          // Niveau d'énergie sonore au temps ti
100   double m_P;                           // Sensibilité Electret en dB SPL
101   double m_M;                           // Sensibilité Electret en dBV/Pa
102   double m_G;                           // Gain du MAX4466 en dBV
103   uint32_t mNb_li;                       // Nb d'échantillons
104 };
105 #endif
106
```