```
/**
 * @file        MyNumber.h
 * @synopsis    MyNumber-Klasse Deklaration
 * @author      Jan Tammen (FH Konstanz), <jan.tammen@fh-konstanz.de>
 * @date        2005-03-19
 */

#ifndef MYNUMBER_H
#define MYNUMBER_H

#include "Exception.h"
#include "Langzahl.h"
#include <vector>
#include <iterator>
#include <sstream>

using namespace std;

/**
 * @synopsis MyNumber-Klasse. Implementiert das Interface Langzahl.
 * @detail   Implementierung der Langzahldarstellung und teilweiser
 *           Langzahlarithmetik.
 */
class MyNumber : public virtual Langzahl
{
    public:
        /// Konstruktoren + Destruktor
        MyNumber (void);
        MyNumber (const MyNumber& x);
        MyNumber (string number);
        MyNumber (long int number);
        ~MyNumber ();

        /// Ueberladene Operatoren
        MyNumber& operator=  (const MyNumber& z);
        MyNumber  operator+  (const MyNumber& z) const;
        MyNumber  operator-  (const MyNumber& z) const;
        MyNumber  operator*  (const MyNumber& z) const;
        MyNumber& operator+= (const MyNumber& z);
        MyNumber& operator-= (const MyNumber& z);
        MyNumber& operator*= (const MyNumber& z);
        MyNumber  operator-  (void) const;

        /// Vergleichsoperatoren
        bool operator== (const MyNumber& z) const;
        bool operator>  (const MyNumber& z) const;
        bool operator>= (const MyNumber& z) const;
        bool operator<  (const MyNumber& z) const;
        bool operator<= (const MyNumber& z) const;

        bool leq        (const MyNumber& z) const;

        /// Ausgabeoperator
        friend ostream& operator<< (ostream& s, const MyNumber& z);

        /// Zugriffsmethoden
        long int  getFractionLength (void) const { return mFractionLength;  }
        short     getSign (void)           const { return mSign;            }
        void      setSign (const int n)          { mSign = n >= 0 ? 1 : -1; }

        unsigned int getComma (void)       const { return mComma;           }
        void         setComma (int x);

        MyNumber  getKehrwert (void);

        bool      isNegative () const            { return mSign <  0;       }
        bool      isPositive () const            { return mSign >= 0;       }

        void      resize (const unsigned int n)  { mNumber.resize(n);       }
```



```
    private:
        static const unsigned int mMaxDecimalPlaces = 30;
        short int                 mSign;
        unsigned int              mComma;
        vector<int>               mNumber;
        unsigned int              mFractionLength;
};

inline MyNumber abs(const MyNumber& z)
{
    return (z.isPositive()) ? z : (-z);
}

#endif
/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4 foldmethod=marker: */
```