```
/****
 *
 * Class Model is an abstract parent class for model classes in an MVP design.
 * See Fisher SE lecture notes for further discussion of the MVP design
 * methodology.
 *
 */

#ifndef modelIncluded
#define modelIncluded

#include "std-macros.h"

class View;


class Model {
 public:
    Model(View* v = NULL);
    /*
     * Construct a model with the given View.
     *
     * post: this'->v = v;
     */

    ~Model();
    /*
     * Shallow distruct this.  I.e., do not delete its view;
     *
     * post: freed(this');
     */

    void SetView(View* v);
    /*
     * Set the view of this to the given view, if the view is not already set.
     * This SetView function is used if this must be constructed before its
     * companion view is constructed, and therefore the view will not be
     * available to pass to the constructor.
     *
     * Models with multiple and/or dynamically changeable views must manage
     * view changes with addtional data members.  This' canonical view can be
     * set only once, with either the constructor or one call to SetView.
     *
     * pre: this->v == NULL;
     *
     * post: this'->v == v; */

    View* GetView();
    /*
     * Return the view of this.
     *
     * post: return == v;
     */

 protected:
    View* v;                    // The cannonical view for this model.  Models
                                // with multiple views will add additional view
                                // data members as necessary.

    Model(Model &);             // Standard protected copy constructor
    Model& operator=(Model &);  // Standard protected op =
};

#endif
```