```
/****
 *
 * Class ChangeCardDialog contains a string editor for holding the name of the
 * card(s) to be changed.  Above the string editor is the change dialog
 * message.  Below the string editors are OK and Cancel buttons.
 *
 */

#ifndef changecarddialogIncluded
#define changecarddialogIncluded

#include "rolodex.h"
#include "view.h"
#include <InterViews/box.h>
#include <InterViews/button.h>
#include <InterViews/streditor.h>

class CancelChangeButton;
class OKChangeButton;
class RolodexMenuUI;            // Avoid circular include


class ChangeCardDialog : public View {
  public:
    ChangeCardDialog(Rolodex* r, RolodexMenuUI* rmui);
    /*
     * Construct the dialog to enter the id of a card to be changed rolodex.
     * The given Rolodex is the companion model.  The given RolodexMenuUI is
     * the parent view.
     */

    ~ChangeCardDialog();

    virtual void Compose();
    /*
     * Compose this by inserting a labeled string editor for entering the card
     * id.
     */

    String* GetId();
    /*
     * Return the text value of the id string editor.
     */

  protected:

    void ComposeButtons();
     /*
      * Compose the OK and Cancel dialog buttons.
      */


    StringEditor* seid;         // String editor for id field
    ButtonState* bs;            // Dummy (i.e., unused) button state for all
                                // string editors

    VBox* vb;                   // VBox for holding the label, str ed, and
                                // button box.
    HBox* hb;                   // Hbox for holding the buttons.

    OKChangeButton* okcb;       // OK button
    CancelChangeButton* ccb;    // Cancel button

};

class OKChangeButton : public PushButton {

  public:
    OKChangeButton(Rolodex* r, ChangeCardDialog* ccd, RolodexMenuUI* rmui);
    /*
     * Construct this with the given rolodex model parent dialog view, and
     * grandparent rolodex view.  Access to the model is for calling its Change
     * function.  Access to the parent dialog is for gathering data to be sent
     * to Rolodex::Change.  Access to the grandparent view is for reporting
     * input error messages.  See the specs for Press below for further
     * discussion.
     */

  protected:
    virtual void Press();
    /*
     * Respond to a press of this button by calling Rolodex::Change.  The id
     * of the card sent as the parameter to Change is gathered from the string
     * editor for the card id field.  This editor is contained in the parent
     * dialog, and is hence a view sibling of this button.
     */

    Rolodex* r;                         // The companion model
    ChangeCardDialog* ccd;              // The parent view
    RolodexMenuUI* rmui;                // The grandparent view
};

class CancelChangeButton : public PushButton {

  public:
    CancelChangeButton(RolodexMenuUI* rmui);
    /*
     * Construct this with the given grandparent rolodex view.  Access to the
     * parent view is for the purposes of effecting the cancelation, which in
     * terms of the view means to replace the change card dilaog with an empty
     * data entry area.  Note that no model access is necessary in this button
     * since pressing this has no effect on the rolodex, only on the rolodex
     * view.
     */

  protected:
    virtual void Press();
    /*
     * Respond to a press of this button by calling
     * RolodexMenuUI::ShowNoDialog.
     */

```

```
113      RolodexMenuUI* rmui;               // The grandparent view
114  };
115  #endif;
```