```
/****
 *
 * Class DeleteCardDialog contains a string editor for holding the id of a card
 * to be deleted.  Above the string editor is the delete dialog message.  Below
 * the string editors are OK and Cancel buttons.
 *
 */

#ifndef deletecarddialogIncluded
#define deletecarddialogIncluded

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

class CancelDeleteButton;
class OKDeleteButton;
class RolodexMenuUI;            // Avoid circular include


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

    ~DeleteCardDialog();

    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.

    OKDeleteButton* okdb;       // OK button
    CancelDeleteButton* cdb;    // Cancel button

};

class OKDeleteButton : public PushButton {

  public:
    OKDeleteButton(Rolodex* r, DeleteCardDialog* dcd, RolodexMenuUI* rmui);
    /*
     * Construct this with the given rolodex model and parent dialog view.
     * Access to the model is for calling its Delete function.  Access to the
     * parent dialog is for gathering data to be sent to Rolodex::Delete.
     * Access to the grandparent ui is for input error messaging See the specs
     * for Press below for further discussion.
     */

  protected:
    virtual void Press();
    /*
     * Respond to a press of this button by calling Rolodex::Delete.  The id
     * of the card sent as the parameter to Delete 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
    DeleteCardDialog* dcd;              // The parent view
    RolodexMenuUI* rmui;                // The grandparent view
};

class CancelDeleteButton : public PushButton {

  public:
    CancelDeleteButton(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 delete 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.
     */

    RolodexMenuUI* rmui;                // The grandparent view
```

```
113  };
114  #endif;
```