```
/****
 *
 * Implementation of rolodex-menu-ui.h
 *
 */

#include "edit-menu.h"
#include "file-menu.h"
#include "rolodex-menu-ui.h"
#include "rolodex-menu.h"
#include "rolodex-tool.h"
#include "white-background-tray.h"
#include <InterViews/border.h>
#include <InterViews/glue.h>
#include <InterViews/message.h>
#include <InterViews/tray.h>


RolodexMenuUI::RolodexMenuUI(Screen* s, RolodexTool* rt) :
        RolodexUI(s, rt) {
    /*
     * Construct the file menu, passing its File model.
     */
    fm = new FileMenu(new String("File"), rt->GetFile());

    /*
     * Construct the edit menu, passing its Edit model.
     */
    em = new EditMenu(new String("Edit"), rt->GetEdit());

    /*
     * Construct the edit menu, passing its Edit model.
     */
    rm = new RolodexMenu(new String("Rolodex"), rt->GetRolodex(), this);

    /*
     * For convenience, save a copy of the model as a downcast value.  Hence,
     * this->m == this->rt, where the former is of type Model* and the latter
     * is of type RolodexTool*.
     */
    this->rt = rt;
}

RolodexMenuUI::~RolodexMenuUI() {
}

void RolodexMenuUI::Compose() {

    /*
     * Make the window of this a WhiteBackgroundTray, which isspecialization of
     * an InterViews Tray.  A Tray is a standard outermost interactor for a
     * top-level interface since it provides a opaque background.  An HBox or
     * VBox by itself has a transparent background, which means that if it's
     * used as the outmost interactor, it will "absorb" whatever window
     * background is underneath it.  Hence, it is standard practice to use a
     * Tray for the solid background.  Trays have other useful purposes, but
     * here we're simply using one for its background.
     */
    w = new WhiteBackgroundTray(vb = new VBox());

    /*
     * Compose the menubar and insert it into this' window.  The border
     * provides a visible separator under the menubar.  By default a menubar
     * has no visible border.
     */
    vb->Insert(ComposeMenuBar());
    vb->Insert(new HBorder());

    /*
     * Compose the data entry area and insert into the window.
     */
    vb->Insert(ComposeDataArea());

    /*
     * Make the start-up contents of the data entry empty.
     */
    d->FlipTo(EMPTYPOSITION);

    /*
     * Set the name of the top-level interactor, which name will appear in the
     * titlebar of the window.
     */
    w->SetName("Rolodex Tool");
}

MenuBar* RolodexMenuUI::ComposeMenuBar() {
    mb = new MenuBar();

    /*
     * Compose the file menu view and insert its window into the bar.
     */
    mb->Include((PulldownMenu*)ComposeFileMenu()->GetWindow());

    /*
     * Stick some glue between the file and edit menus.
     */
    mb->Insert(new HGlue(
        round(0.2*inch), round(0.15*inch), 0));
     // ^^^^^^^^^^^^^^^  ^^^^^^^^^^^^^^^  ^ Stretchibilty,
     // ^^^^^^^^^^^^^^^  ^^^^^^^^^^^^^^^
     // ^^^^^^^^^^^^^^^  ^^^^^^^^^^^^^^^ Shrinkibility, i.e., minimum size
     // ^^^^^^^^^^^^^^^
     // ^^^^^^^^^^^^^^^ Natural size, i.e., initial size

    /*
     * Compose the edit pulldown and insert it into the bar.
     */
    mb->Include((PulldownMenu*)ComposeEditMenu()->GetWindow());

    /*
     * Stick some glue between the edit and rolodex menus.
     */
```

```
113      mb->Insert(new HGlue(round(0.2*inch), round(0.15*inch), 0));
114
115      /*
116       * Compose the rolodex pulldown and insert it into the bar.
117       */
118      mb->Include((PulldownMenu*)ComposeRolodexMenu()->GetWindow());
119
120      /*
121       * Stick some glue on the right edge of the menubar.
122       */
123      mb->Insert(new HGlue(round(0.15*inch), 0, hfil));
124
125      return mb;
126  }
127
128  HBox* RolodexMenuUI::ComposeDataArea() {
129
130      /*
131       * Allocate a deck to hold the operation-specific dialogs.
132       */
133      d = new Deck();
134
135      /*
136       * Allocate and comppose each of the dialogs and insert each into the deck.
137       * Note that the model for each of the dialogs is an appropriate components
138       * of the rolodex tool workspace
139       */
140      ComposeDialogs();
141
142      /*
143       * Put the deck in an HBox with some glue on each end.  This outer hbox
144       * with its glue will allow horizontal stretchibility.  Note here how the
145       * HBox constructor can take more than one argument -- it can take up to 7
146       * arguments.
147       */
148      return new HBox(
149          new HGlue(round(0.2 * inch), 0, hfil),
150          d,
151          new HGlue(round(0.2 * inch), 0, hfil)
152      );
153  }
154
155  FileMenu* RolodexMenuUI::ComposeFileMenu() {
156
157      /*
158       * Compose the file menu, which entails inserting each of its items.
159       */
160      fm->Compose();
161
162      /*
163       * Return the composed menu.
164       */
165      return fm;
166  }
167
168  EditMenu* RolodexMenuUI::ComposeEditMenu() {
169
170      /*
171       * Compose the edit menu, which entails inserting each of its items.
172       */
173      em->Compose();
174
175      /*
176       * Return the composed menu.
177       */
178      return em;
179  }
180
181  RolodexMenu* RolodexMenuUI::ComposeRolodexMenu() {
182
183      /*
184       * Compose the rolodex menu, which entails inserting each of its items.
185       */
186      rm->Compose();
187
188      /*
189       * Return the composed menu.
190       */
191      return rm;
192  }
193
194  void RolodexMenuUI::ComposeDialogs() {
195
196      /*
197       * Position 1 -- Add.
198       */
199      acd = new AddCardDialog(rt->GetRolodex(), this);
200      ComposeAndInsert(acd);
201
202      /*
203       * Position 2 -- Delete.
204       */
205      dcd = new DeleteCardDialog(rt->GetRolodex(), this);
206      ComposeAndInsert(dcd);
207
208      /*
209       * Position 3 -- Confirm Delete.
210       */
211      cdd = new ConfirmDeleteDialog(rt->GetRolodex(), this);
212      ComposeAndInsert(cdd);
213
214      /*
215       * Position 4 -- Change.
216       */
217      ccd = new ChangeCardDialog(rt->GetRolodex(), this);
218      ComposeAndInsert(ccd);
219
220      /*
221       * Position 5 -- Confirm Change.
222       */
223       cncd = new ConfirmChangeDialog(rt->GetRolodex(), this);
224       ComposeAndInsert(cncd);
```

```
225
226      /*
227       * Position 6 -- Find.
228       */
229       fcd = new FindCardDialog(rt->GetRolodex(), this);
230       ComposeAndInsert(fcd);
231
232      /*
233       * Position 7 -- Found.
234       */
235       fcsd = new FoundCardsDialog();
236       ComposeAndInsert(fcsd);
237
238      /*
239       * Position 8 -- Error.
240       */
241      emd = new ErrorMessageDialog();
242      ComposeAndInsert(emd);
243
244      /*
245       * Position 9 -- Empty.
246       */
247      d->Insert(new Tray());
248  }
249
250  void RolodexMenuUI::ComposeAndInsert(View* v) {
251      v->Compose();
252      d->Insert(v->GetWindow());
253  }
254
255  void RolodexMenuUI::ShowAddCardDialog() {
256      d->FlipTo(curpos = ADDPOSITION);
257  }
258
259  void RolodexMenuUI::ShowDeleteCardDialog() {
260      d->FlipTo(curpos = DELPOSITION);
261  }
262
263  void RolodexMenuUI::ShowChangeCardDialog() {
264      d->FlipTo(curpos = CHANGEPOSITION);
265  }
266
267  void RolodexMenuUI::ShowFindCardDialog() {
268      d->FlipTo(curpos = FINDPOSITION);
269  }
270
271  void RolodexMenuUI::ShowFoundDialog() {
272      d->FlipTo(curpos = FOUNDPOSITION);
273  }
274
275  void RolodexMenuUI::ShowErrorDialog(String* s) {
276      emd->SetMessage(s);
277      d->FlipTo(ERRORPOSITION);
278  }
279
280  void RolodexMenuUI::ShowErrorDialog(StrList* sl) {
281      emd->SetMessages(sl);
282      d->FlipTo(ERRORPOSITION);
283  }
284
285  void RolodexMenuUI::ShowEmptyDialog() {
286      d->FlipTo(curpos = EMPTYPOSITION);
287  }
```