```
/****
 *
 * Implementation of rolodex.h.
 *
 */

#include "rolodex.h"
#include "std-macros.h"
#include <stdio.h>


Rolodex::Rolodex(View* v) : Model(v) {
    cl = new CardList();
    aie = new AddInputErrors();
    die = new DeleteInputError();
    cie = new ChangeInputErrors();
}

Rolodex::~Rolodex() {
    delete cl;
}

void Rolodex::Add(Card* c) {
    /*
     * Clear out the input error data.
     */
    aie->Clear();

    /*
     * Throw a precond violation if a card of the given id is already in
     * this->data.
     */
    if (cl->Find(c->GetId())) {
        aie->SetAlreadyThereError();
        throw aie;
    }

    /*
     * Throw a precond violation if the validy check fails for one or more of
     * the card fields.  The exeption value is a tuple of values for each of
     * the invalid fields.
     */
    if (ValidateAddInput(c)->AnyErrors()) {
        throw aie;
    }

    /*
     * If preconditions are met, add the given card to this->data using the
     * inherited List::Put function.
     */
    cl->Put(new CardListElem(c));
}

void Rolodex::Delete(Id id) {
    /*
     * Throw a precond violation if a card of the given id is not in
     * this->data.
     */
    if (not cl->Find(id)) {
        throw die;
    }

    /*
     * If the precondition is met, delete the given card from this->data using
     * the inherited List::FindPos and List::DelNth functions.
     */
    cl->DelNth(cl->FindPos((ListElemKey*) id));
}

void Rolodex::Change(Id id, Card* c) {
    /*
     * Clear out the input error tuple.
     */
    cie->Clear();

    /*
     * Throw a precond violation if a card of the given id is not already in
     * this->data.
     */
    if (not cl->Find(id)) {
        cie->SetNotThereError();
        throw cie;
    }

    /*
     * Throw a precond violation if a card of the given id is not already in
     * this->data.
     */
    if (cl->Find(id)->Equal(c)) {
        cie->SetIdenticalToExtantError();
        throw cie;
    }

    /*
     * Throw a precond violation if the validy check fails for one or more of
     * the card fields.  The exeption value is a tuple of values for each of
     * the invalid fields.
     */
    if (ValidateAddInput(c)->AnyErrors()) {
        throw cie;
    }

    /*
     * If preconditions are met, delete the extant card and add the new one.
     */
    Delete(id);
    Add(c);
}

CardList* Rolodex::Find(Name* n) {
    Card* c;
    CardList* rtn = new CardList();
```

```
113
114     while (c = cl->Enum()) {
115         if (*(c->GetName()) == *n)
116             rtn->Put(new CardListElem(c));
117     }
118     return rtn;
119 }
120
121 void Rolodex::Print() {
122     Dump();
123 }
124
125 void Rolodex::Dump() {
126     cl->Print();
127 }
128
129 Sex Rolodex::StringToSex(String* s) {
130     if (*s == "m")
131         return Male;
132     else
133         return Female;
134 }
135
136 String* Rolodex::SexToString(Sex s) {
137     if (s == Male)
138         return new String("m");
139     else
140         return new String("f");
141 }
142
143 AddInputErrors* Rolodex::ValidateAddInput(Card* c) {
144     if (c->GetName()->Len() > 30)
145         aie->SetNameError();
146
147     if ((c->GetId() < 100000000) or (c->GetId() > 999999999))
148         aie->SetIdError();
149
150     if ((c->GetAge() < 0) or (c->GetAge() > 200))
151         aie->SetAgeError();
152
153     if (c->GetSex() == InvalidSex)
154         aie->SetSexError();
155
156     if (c->GetAddr()->Len() > 40)
157         aie->SetAddressError();
158
159     return aie;
160 }
```