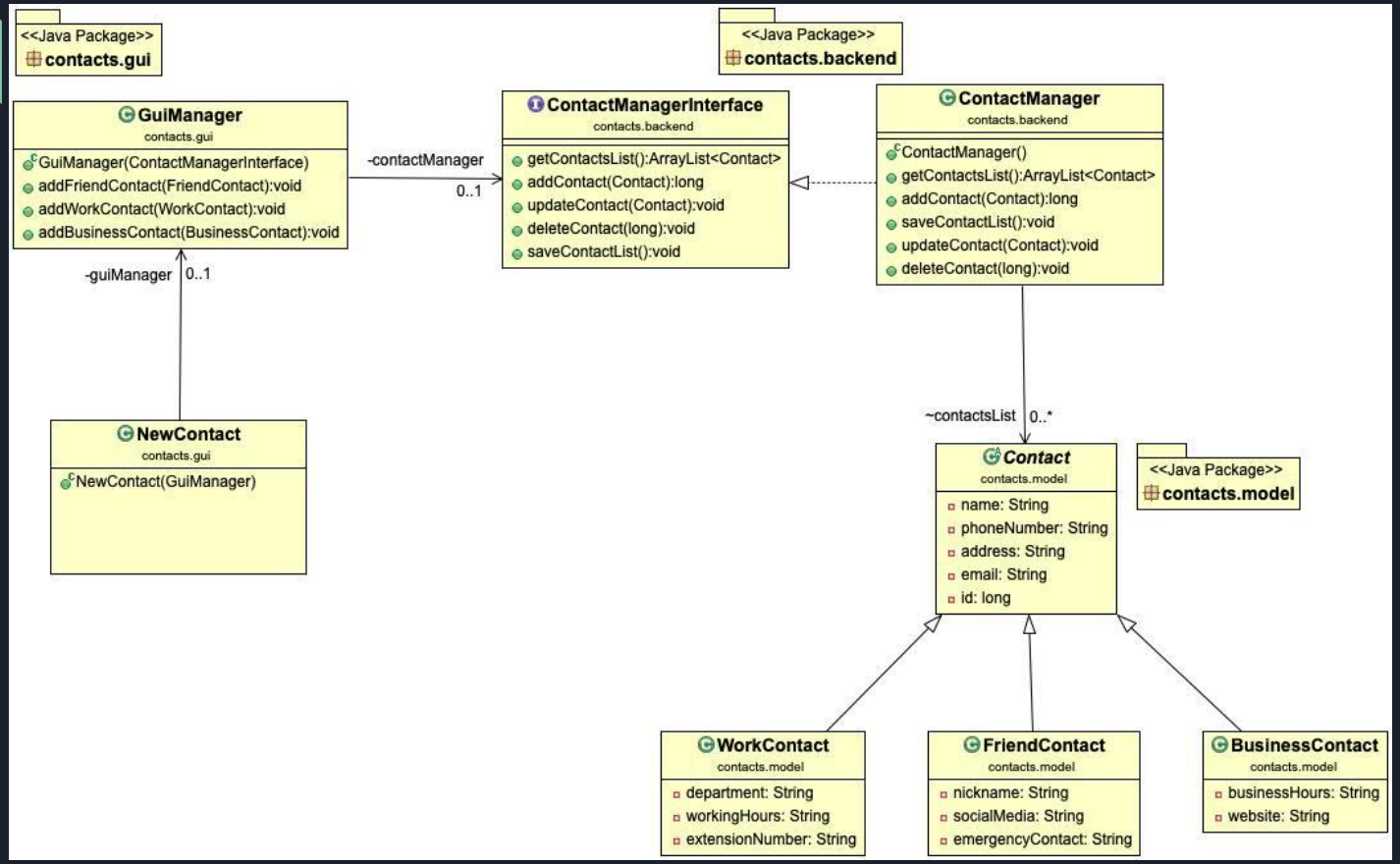
Contact Manager Application

Arjun Tyagi

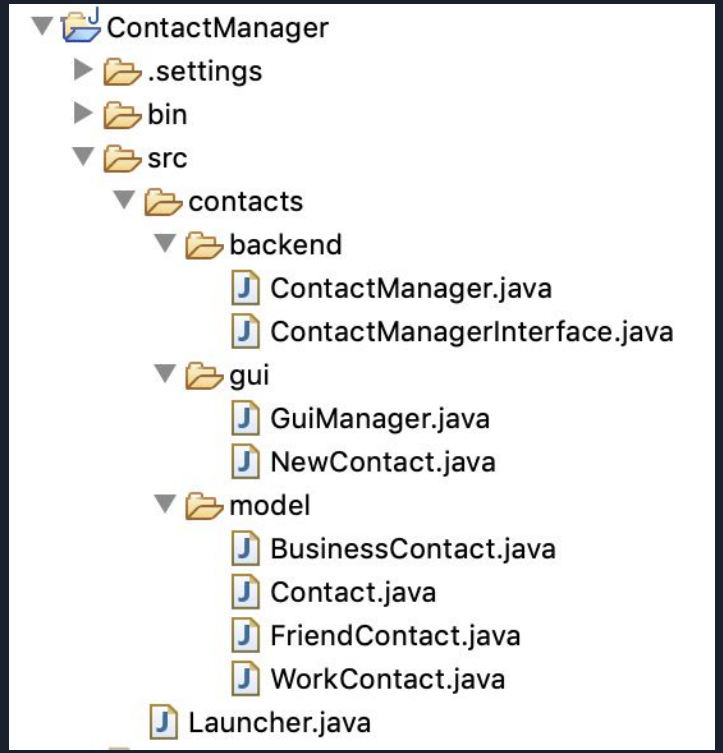
UML Class Diagram



Source Code Organization

The classes are is organized into three packages:

- contacts.model.*
  - Represents relationships among data
- contacts.backend.*
  - Performs file I/O and holds data in memory
  - Updates the stored data based on UI operations
- contacts.gui.*
  - Provides a user interface and functionality for buttons



Package contacts.model

- Utilizes inheritance to encapsulate common data in a base class.
- Subclasses are defined to represent data specific to a different contact types

```java
package contacts.model;


public abstract class Contact {
    private String name;
    private String phoneNumber;
    private String address;
    private String email;
    private long id;

    public Contact(String name, String phoneNumber, String address, String email) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.address = address;
        this.email = email;
    }
```

```java
package contacts.model;

public class BusinessContact extends Contact {

    private String businessHours;
    private String website;

    public BusinessContact(String name, String phoneNumber, String address, String email, String businessHours, String website) {
        super(name,phoneNumber,address,email);
        this.businessHours = businessHours;
        this.website = website;

    }
```

Package contacts.model (Cont.)

- Each class provides its string representation for writing to the file by overriding toString() utilizing StringBuilder

```
42      //toString() for FriendContact Class
43      @Override
44      public String toString() {
45          StringBuilder recordBuilder = new StringBuilder("0").append("|").append(super.toString()).append("|")
46                  .append(getNickname()).append("|").append(getSocialMedia()).append("|").append(getEmergencyContact());
47          return recordBuilder.toString();
48      }
```

```
// .toString() is invoked on the base class polymorphically
String contactRecordString;
for (Contact aContact:contactsList) {
    contactRecordString = aContact.toString();
    printWriter.println(contactRecordString);
}
```

Package contacts.backend

- ContactManagerInterface - Provides interface to be used by GUI to access backend.
- ContactManager - Provides concrete implementation of the ContactManagerInterface

```
1  package contacts.backend;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5
6  import contacts.model.Contact;
7
8  public interface ContactManagerInterface {
9
10     ArrayList<Contact> getContactsList();
11
12     long addContact(Contact aNewContact);
13
14     void updateContact(Contact updatedContact);
15
16     void deleteContact(long id);
17
18     void saveContactList() throws IOException;
19
20 }
```

Package contacts.backend (cont.)

- Utilizes concepts of:
  - File I/O
  - ArrayLists
  - Exception Handling

```
1  package contacts.backend;
2
3  import contacts.model.*;
13
14 public class ContactManager implements ContactManagerInterface {
15
16     ArrayList<Contact> contactsList;
17
18     public ContactManager() throws FileNotFoundException {
19         contactsList = new ArrayList<Contact>();
20
21         File contactsFile = new File("contact_data.txt");
22         Scanner filescanner = new Scanner(contactsFile);
23         try {
24             while(filescanner.hasNextLine()){
25                 String contactDataEntry = filescanner.nextLine().trim();
26                 if (contactDataEntry.length() == 0)
27                     continue;
28                 parseContactEntry(contactDataEntry);
29             }
30         }
31         finally {
32             filescanner.close();
33         }
34     }
```

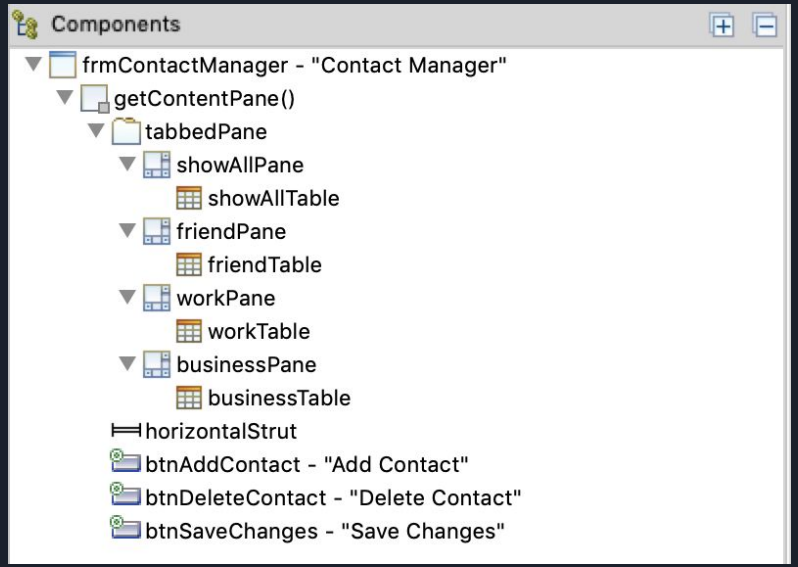
Exception Handling Example (cont.)

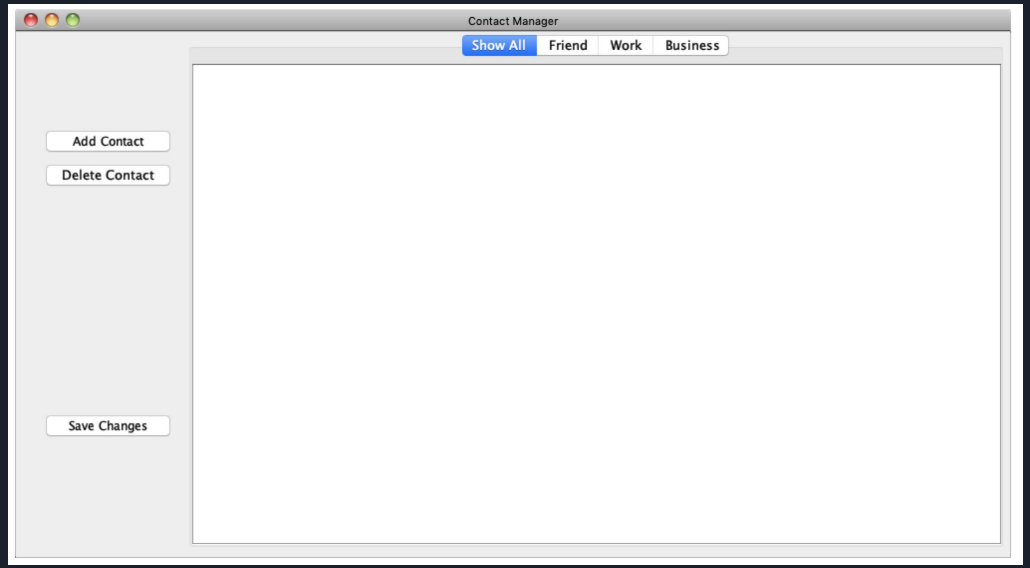
```
 1 import java.awt.*;
 6
 7 public class Launcher {
 8     /**
 9      * Launch the application.
10      */
11     public static void main(String[] args) {
12         EventQueue.invokeLater(new Runnable() {
13             public void run() {
14                 try {
15                     ContactManagerInterface contactManager = new ContactManager();
16                     new GuiManager(contactManager);
17                 } catch (Exception e) {
18                     e.printStackTrace();
19                 }
20             }
21         });
22     }
23 }
```

Package contacts.gui

- Provides classes for drawing the user interface
- Provides add, delete, and save operations for various kinds of contacts.
- Relies on the ContactManager interface to read and write data

Utilizes concepts of: JFrame, GridBagLayout, TabbedPane, and JTable





Package contacts.gui (cont.)

- ActionListeners for the buttons:

```
189     private class AddContactButtonListener implements ActionListener {
190         public void actionPerformed(ActionEvent e) {
191             NewContact myAddcontactPane = new NewContact(GuiManager.this);
192             myAddcontactPane.setVisible(true);
193         }
194     }
```

```
246     private class SaveButtonListener implements ActionListener {
247         public void actionPerformed(ActionEvent e) {
248             try {
249                 contactManager.saveContactList();
250                 JOptionPane.showMessageDialog(null, "Saved contact succesfully!");
251             } catch (IOException e1) {
252                 e1.printStackTrace();
253                 JOptionPane.showMessageDialog(null, "Error, failed to save contact!");
254             }
255
256         }
257     }
```