Report of CS 522 Assignment 2

Name: Yunfeng Wei
CWID: 10394963
E-mail: ywei14@stevens.edu
Video: In the Zip archive file,
BookStore App: “Yunfeng_Wei_assignment_2_BookStore.MOV”
Basic Chat App: “Yunfeng_Wei_assignment_2_BasicChat.MOV”

Part 1: Book Store

1. First, modify the “Author.java”. Implement Parcelable interface to Author class, override the describeContents(), writeToParcel() methods and define a Creator<Author> called CREATOR, in the Creator, override the createFromParcel() and newArray methods() to let the Author support Parcelable interface. Finally, define the constructor of Author and override the toString method to display the full name of the Author, as Figure 1.1

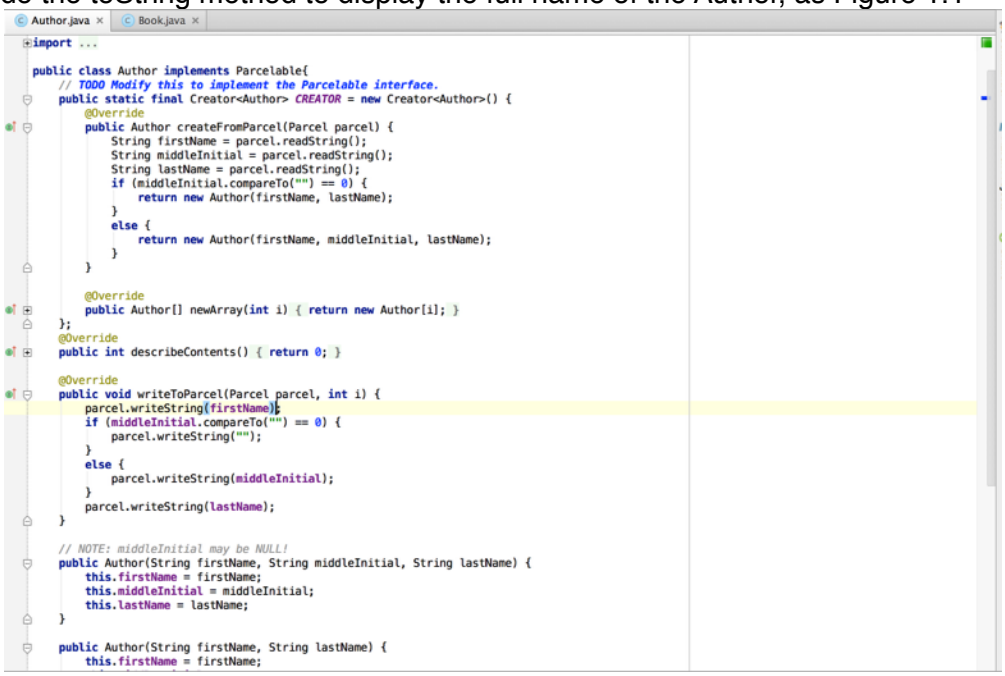


Figure 1.1

2. Modify the “Book.java”. Implement Parcelable interface and override the describeContents(), wirtieToParcels() methods and define a Creator<Book> as the Author class. Override the toString() method to display the title and price of the book as Figure 2.1 and Figure 2.2

```java
public class Book implements Parcelable {

    // TODO Modify this to implement the Parcelable interface.
    public static final Creator<Book> CREATOR = new Creator<Book>() {
        @Override
        public Book createFromParcel(Parcel parcel) {
            int id = parcel.readInt();
            String title = parcel.readString();
            int length = parcel.readInt();
            Author[] authors = new Author[length];
            for (int j = 0; j < length; j++) {
                try {
                    authors[j] = parcel.readParcelable(Author.class.getClassLoader());
                }
                catch (BadParcelableException e) {
                    Log.e(Book.class.getCanonicalName(), "BadParcelableException", e);
                }
            }
            String isbn = parcel.readString();
            String price = parcel.readString();
            return new Book(id, title, authors, isbn, price);
        }

        @Override
        public Book[] newArray(int i) { return new Book[i]; }
    };

    @Override
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeInt(id);
        parcel.writeString(title);
        parcel.writeInt(authors.length);
        for (int j = 0; j < authors.length; j++) {
            parcel.writeParcelable(authors[j], i);
        }
        parcel.writeString(isbn);
        parcel.writeString(price);
    }
```

Figure 2.1

```java
// TODO redefine toString() to display book title and price (why?).
@Override
public String toString() { return "Title: " + this.title + "     Price: " + this.price; }

public int id;

public String title;

public Author[] authors;

public String isbn;

public String price;

public Book(int id, String title, Author[] author, String isbn, String price) {
    this.id = id;
    this.title = title;
    this.authors = author;
    this.isbn = isbn;
    this.price = price;
}
```

Figure 2.2

3. Complete the BookStoreActivity.java. First define a ArrayAdapter<Book> bookArrayAdapter to be used for display the each book in the ListView. Define the BOOK_ID as the book's id. Define the String ID, BOOK_STORE_KEY and SHOPPING_CART as the key for intent. use savedInstatnceState to store and restore the shoppingCart and the BOOK_ID as Figure 3.1.

```
// There is a reason this must be an ArrayList instead of a List.
/unused/
private ArrayList<Book> shoppingCart;

private ArrayAdapter<Book> bookArrayAdapter = null;

public static int BOOK_ID; // Initialize the Book id
private static final String ID = "BOOK_ID";
public static final String BOOK_STORE_KEY = "BOOK_STORE_KEY";
private static final String SHOPPING_CART = "SHOPPING_CART";
```

Figure 3.1

In order to display the book's name, price and authors in two TextViews. Rewrite the ArrayAdapter called BookAdapter, override the getView() method to display the Book's title, price and Book's authors in two TextViews, the code of BoodAdapter is displayed in Figure 3.2.

```
public class BookAdapter extends ArrayAdapter<Book> {
    int resource;
    public BookAdapter(Context context, int textViewResourceId, List<Book> objects) {
        super(context, textViewResourceId, objects);
        resource = textViewResourceId;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LinearLayout linearLayout;
        Book book = getItem(position);
        String title = book.toString();
        String author = "";
        for (int i = 0; i < book.authors.length; i++) {
            author += book.authors[i].toString();
            if (i < book.authors.length - 1) {
                author += "; ";
            }
        }

        if (convertView == null) {
            linearLayout = new LinearLayout(getContext());
            String inflater = Context.LAYOUT_INFLATER_SERVICE;
            LayoutInflater inflater1;
            inflater1 = (LayoutInflater)getContext().getSystemService(inflater);
            inflater1.inflate(resource, linearLayout, true);
        }
        else {
            linearLayout = (LinearLayout)convertView;
        }

        TextView titleView = (TextView)linearLayout.findViewById(R.id.cart_row_title);
        TextView authorView = (TextView)linearLayout.findViewById(R.id.cart_row_author);

        titleView.setText(title);
        authorView.setText(author);
        return linearLayout;
    }
}
```

Figure 3.2

In the onCreate() method, set the layout and setListAdapter for BookStoreActivity, finally register the context view as Figure 3.3

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // TODO check if there is saved UI state, and if so, restore it (i.e. the cart contents)
    if (savedInstanceState != null) {
        shoppingCart = savedInstanceState.getParcelableArrayList(SHOPPING_CART);
        BOOK_ID = savedInstanceState.getInt(ID);
    }
    else {
        shoppingCart = new ArrayList<Book>();
        BOOK_ID = 0;
    }
    // TODO Set the layout (use cart.xml layout)
    setContentView(R.layout.cart);
    // TODO use an array adapter to display the cart contents.
    bookArrayAdapter = new BookAdapter(this, R.layout.cart_row, shoppingCart);
    setListAdapter(bookArrayAdapter);
    registerForContextMenu(getListView());
}
```

Figure 3.3

In order to show details of the book or delete the book, create a context menu and override the onContextItemSelected() method to implement the functions, as Figure 3.4

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, 1, Menu.NONE, "Detail");
    menu.add(0, 2, Menu.NONE, "Delete");
}

@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
    int position = info.position;
    switch (item.getItemId()) {
        case 1:
            Intent mIntent = new Intent(this, BookDetailActivity.class);
            mIntent.putExtra(BOOK_STORE_KEY, shoppingCart.get(position));
            startActivity(mIntent);
            break;
        case 2:
            shoppingCart.remove(position);
            bookArrayAdapter.notifyDataSetChanged();
            break;
    }
    return super.onContextItemSelected(item);
}
```

Figure 3.4

Modify the OnCreateOptionsMenu() and onOpetionsItemSelected() methods to implement the add and checkout functions, as Figure 3.5

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // TODO provide ADD, DELETE and CHECKOUT options - Done
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.bookstore_menu, menu);
    return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    super.onOptionsItemSelected(item);
    // TODO

    // ADD provide the UI for adding a book
    // Intent addIntent = new Intent(this, AddBookActivity.class);
    // startActivityForResult(addIntent, ADD_REQUEST);

    // DELETE delete the currently selected book

    // CHECKOUT provide the UI for checking out
    switch (item.getItemId()) {
        case R.id.add:
            Intent addIntent = new Intent(this, AddBookActivity.class);
            startActivityForResult(addIntent, ADD_REQUEST);
            break;
        case R.id.checkout:
            if (shoppingCart.size() < 1) {
                Toast.makeText(this, "No books", Toast.LENGTH_LONG).show();
                break;
            }
            Intent checkoutIntent = new Intent(this, CheckoutActivity.class);
            checkoutIntent.putExtra(BOOK_STORE_KEY, shoppingCart.size());
            startActivityForResult(checkoutIntent, CHECKOUT_REQUEST);
            break;
    }
    return true;
}
```

Figure 3.5

Modify the onActivityResult() method, receive the resultCode and execute the task according to the requestCode as Figure 3.6

```java
@Override
protected void onActivityResult(int requestCode, int resultCode,
        Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    // TODO Handle results from the Search and Checkout activities.

    // Use SEARCH_REQUEST and CHECKOUT_REQUEST codes to distinguish the cases.

    // SEARCH: add the book that is returned to the shopping cart.

    // CHECKOUT: empty the shopping cart.
    if (requestCode == ADD_REQUEST) {
        if (resultCode == RESULT_OK) {
            Book newBook = intent.getParcelableExtra(AddBookActivity.BOOK_RESULT_KEY);
            shoppingCart.add(newBook);
            bookArrayAdapter.notifyDataSetChanged();
        }
    }
    else if (requestCode == CHECKOUT_REQUEST) {
        if (resultCode == RESULT_OK) {
            shoppingCart.clear();
            BOOK_ID = 0;
            bookArrayAdapter.notifyDataSetChanged();
        }
    }

}
```

Figure 3.6

Finally, save the shopping cart contents and BOOK_ID in onSaveInstanceState() method, and restore it in onCreate() method, as Figure 3.7 and Figure 3.8.

```
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        // TODO save the shopping cart contents (which should be a list of parcelables).
        savedInstanceState.putParcelableArrayList(SHOPPING_CART, shoppingCart);
        savedInstanceState.setClassLoader(getClass().getClassLoader());
        savedInstanceState.putInt(ID, BOOK_ID);
        super.onSaveInstanceState(savedInstanceState);
    }
```

Figure 3.7

```
public class AddBookActivity extends Activity {

    // Use this as the key to return the book details as a Parcelable extra in the result intent.
    public static final String BOOK_RESULT_KEY = "book_result";

    EditText searchTitle = null;
    EditText searchAuthor = null;
    EditText searchIsbn = null;
    Intent intent = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        intent = getIntent();
        setContentView(R.layout.add_book);
        searchTitle = (EditText)findViewById(R.id.search_title);
        searchAuthor = (EditText)findViewById(R.id.search_author);
        searchIsbn = (EditText)findViewById(R.id.search_isbn);
    }
```

Figure 3.8

4. In AddBookActivity.java, setContentView of the Activity in onCreate() method as Figure 4.1.

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // TODO check if there is saved UI state, and if so, restore it (i.e. the cart contents)
        if (savedInstanceState != null) {
            shoppingCart = savedInstanceState.getParcelableArrayList(SHOPPING_CART);
            BOOK_ID = savedInstanceState.getInt(ID);
        }
        else {
            shoppingCart = new ArrayList<Book>();
            BOOK_ID = 0;
        }
        // TODO Set the layout (use cart.xml layout)
        setContentView(R.layout.cart);
        // TODO use an array adapter to display the cart contents.
        bookArrayAdapter = new BookAdapter(this, R.layout.cart_row, shoppingCart);
        setListAdapter(bookArrayAdapter);
        registerForContextMenu(getListView());
    }
```

Figure 4.1

In onCreateOptionsMenu() and onOptionsItemSelected() methods, implement the menu for search the book or cancel the activity as Figure 4.2.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // TODO provide SEARCH and CANCEL options
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.addbook_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    super.onOptionsItemSelected(item);
    // TODO

    // SEARCH: return the book details to the BookStore activity

    // CANCEL: cancel the search request
    switch (item.getItemId()) {
        case R.id.search:
            Book newBook = searchBook();
            intent.putExtra(BOOK_RESULT_KEY, newBook);
            setResult(RESULT_OK, intent);
            finish();
            break;
        case R.id.search_cancel:
            setResult(RESULT_CANCELED);
            finish();
            break;
    }
    return false;
}
```

Figure 4.2

Modify the searchBook() method, build a object with the search criteria and return the Book as Figure 4.3

```java
public Book searchBook(){
    /*
     * Search for the specified book.
     */
    // TODO Just build a Book object with the search criteria and return that.
    String title = searchTitle.getText().toString();
    String author = searchAuthor.getText().toString();
    String[] nameList = author.split(";");
    Author[] authors = new Author[nameList.length];
    for (int i = 0; i < nameList.length; i++) {
        String[] name = nameList[i].split(" ");
        if (name.length == 3) {
            authors[i] = new Author(name[0], name[1], name[2]);
        }
        else if (name.length == 2) {
            authors[i] = new Author(name[0], name[1]);
        }
    }
    String isbn = searchIsbn.getText().toString();
    return new Book(BookStoreActivity.BOOK_ID++, title, authors, isbn, "UNKNOWN");
}

}
```

Figure 4.3

5. Complete the CheckoutActivity.java. Modify the onCreateOptionsMenu() and onOptionsItemSelected() methods to implement the checkout order or cancel order functions, as Figure 5.1

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    // TODO display ORDER and CANCEL options.
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.checkout_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    super.onOptionsItemSelected(item);
    // TODO

    // ORDER: display a toast message of how many books have been ordered and return

    // CANCEL: just return with REQUEST_CANCELED as the result code
    switch (item.getItemId()) {
        case R.id.checkout_order:
            int number = intent.getIntExtra(BookStoreActivity.BOOK_STORE_KEY, 0);
            String temp;
            if (number == 1) {
                temp = " book ordered";
            }
            else {
                temp = " books ordered";
            }
            Toast.makeText(this, number + temp, Toast.LENGTH_LONG).show();
            setResult(RESULT_OK);
            finish();
            break;
        case R.id.checkout_cancel:
            setResult(RESULT_CANCELED);
            finish();
            break;
    }
    return false;
}
```

Figure 5.1

6. In order to show the detail of a book, create a new Activity called BookDetailActivity, modify and layout file as Figure 6.1, the xml code is as Figure 6.2. Override the onCreate() method to display the Book in the layout view as Figure 6.3.

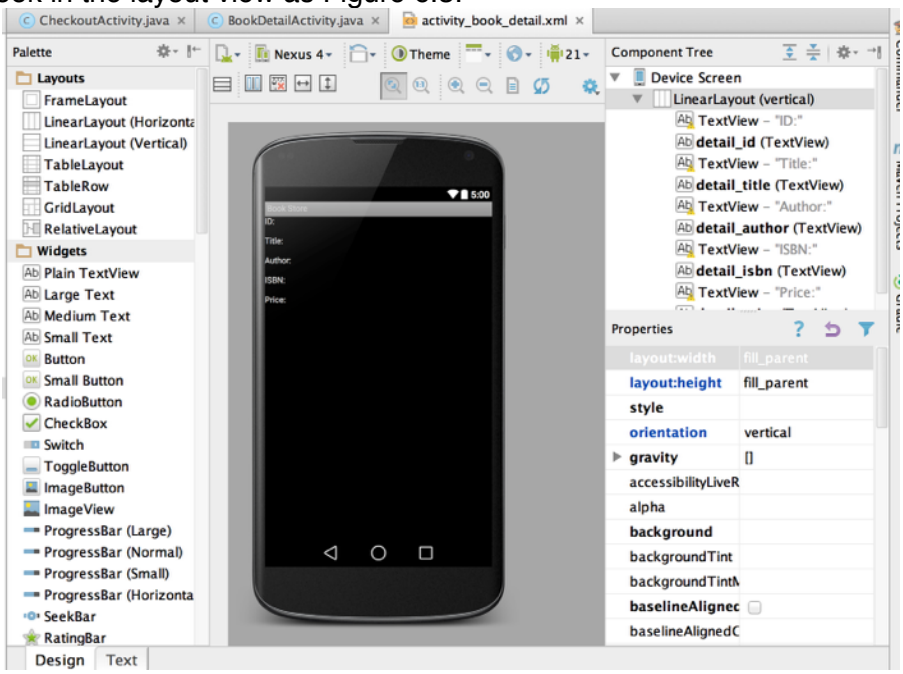
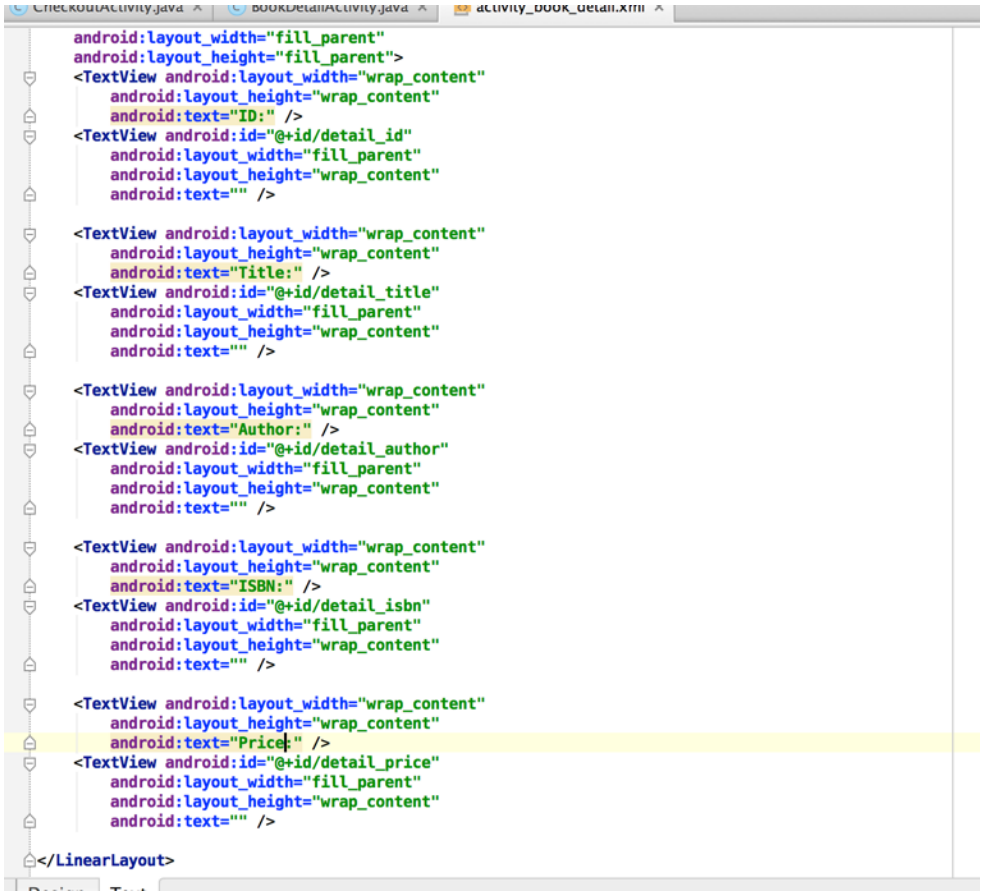


Figure 6.1



```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ID:" />
    <TextView android:id="@+id/detail_id"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Title:" />
    <TextView android:id="@+id/detail_title"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Author:" />
    <TextView android:id="@+id/detail_author"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ISBN:" />
    <TextView android:id="@+id/detail_isbn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Price:" />
    <TextView android:id="@+id/detail_price"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

</LinearLayout>
```

Figure 6.2

```
public class BookDetailActivity extends Activity {
    TextView id = null;
    TextView title = null;
    TextView author = null;
    TextView isbn = null;
    TextView price = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_detail);
        id = (TextView)findViewById(R.id.detail_id);
        title = (TextView)findViewById(R.id.detail_title);
        author = (TextView)findViewById(R.id.detail_author);
        isbn = (TextView)findViewById(R.id.detail_isbn);
        price = (TextView)findViewById(R.id.detail_price);
        Intent intent = getIntent();
        Book book = intent.getParcelableExtra(BookStoreActivity.BOOK_STORE_KEY);
        id.setText(String.valueOf(book.id));
        title.setText(book.title);
        isbn.setText(book.isbn);
        price.setText(book.price);
        Author[] authors = book.authors;
        String author_name = "";
        for (int i = 0; i < authors.length; i++) {
            author_name = author_name + authors[i].toString();
            if (i != authors.length - 1){
                author_name += "; ";
            }
        }
        author.setText(author_name);
    }

}
```

Figure 6.3

7. Run the App, and each activity is shown as Figure 7.1, Figure 7.2, Figure 7.3, Figure 7.4, Figure 7.5, Figure 7.6 and Figure 7.7.

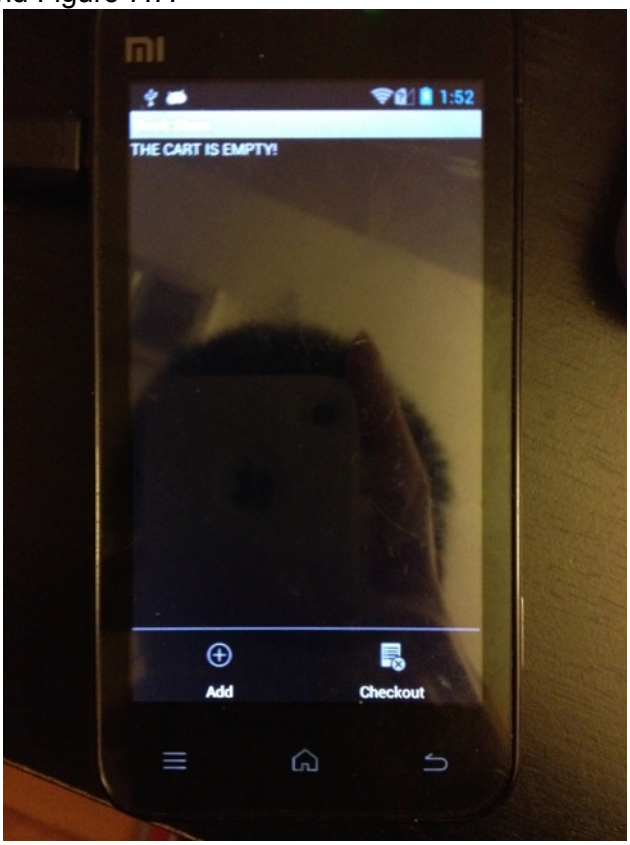


Figure 7.1

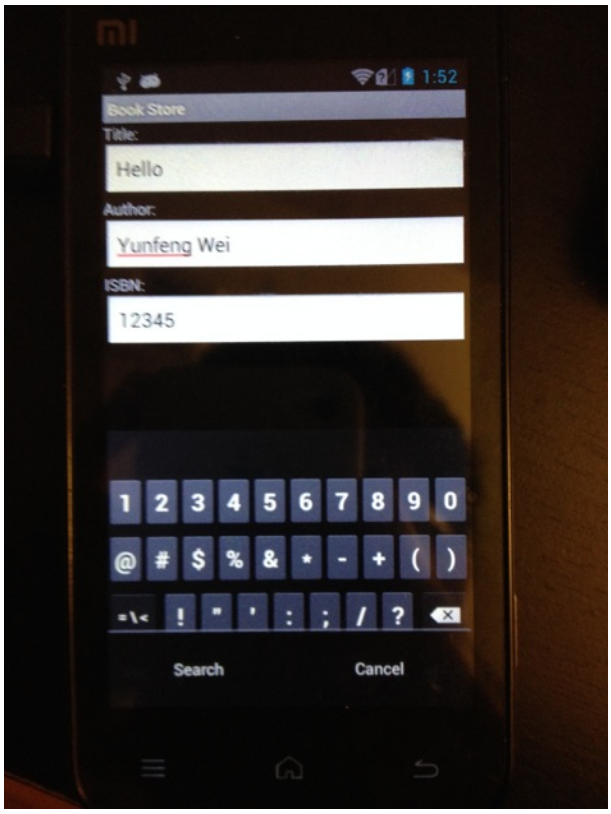


Figure 7.2

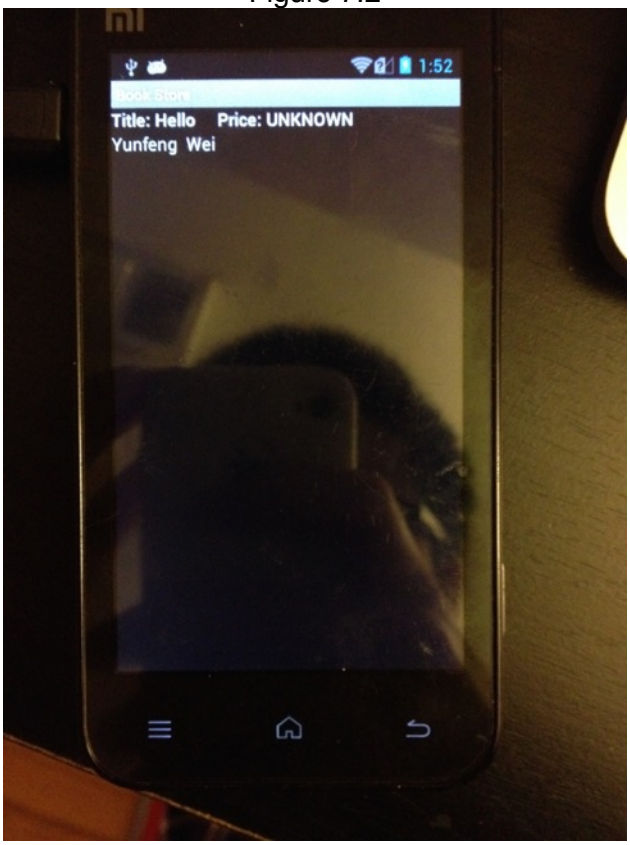
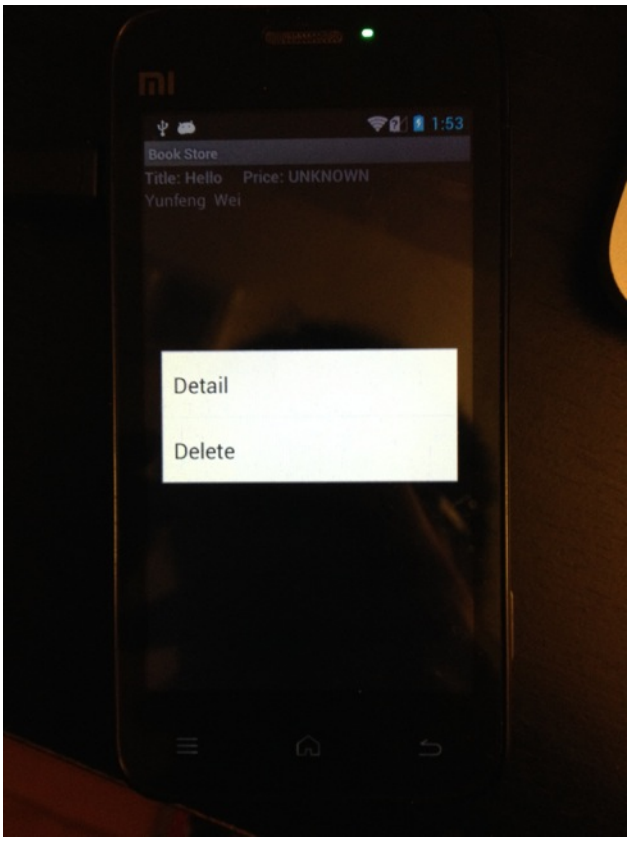
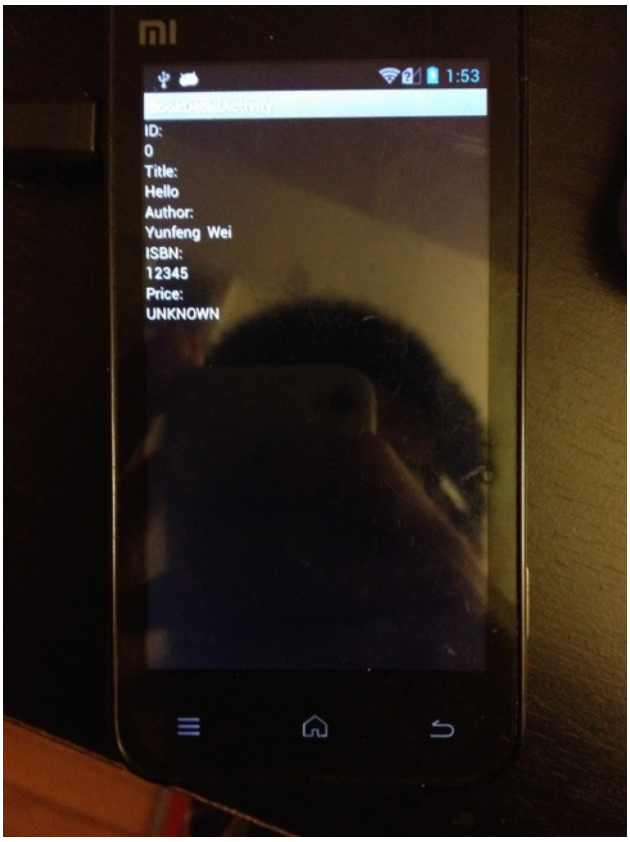


Figure 7.3


1:53
Book Store
Title: Hello Price: UNKNOWN
Yunfeng Wei
Detail
Delete
Figure 7.4


1:53
ID:
0
Title:
Hello
Author:
Yunfeng Wei
ISBN:
12345
Price:
UNKNOWN
Figure 7.5

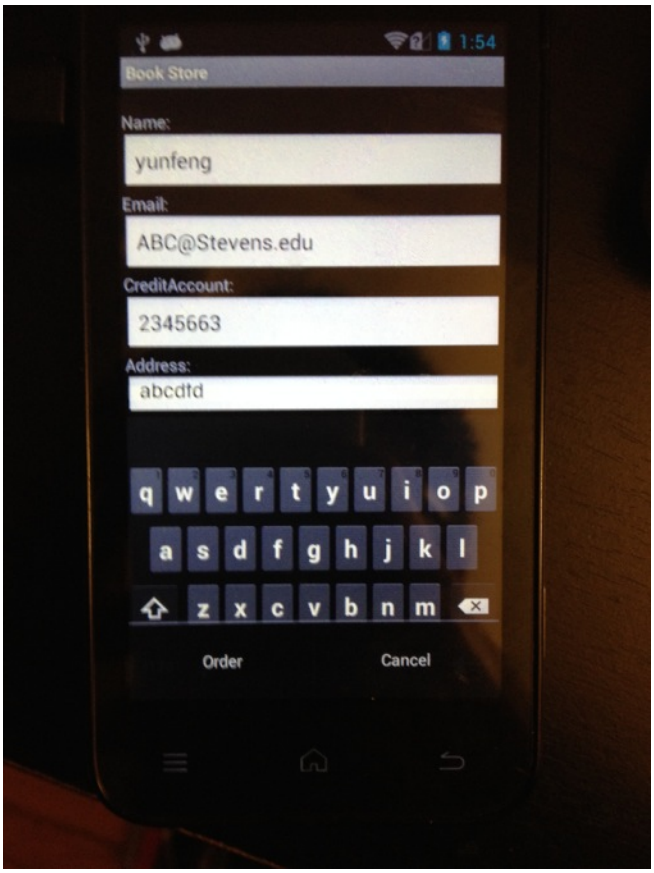

1:54
Book Store
Name:
yunfeng
Email:
ABC@Stevens.edu
CreditAccount:
2345663
Address:
abcdfd
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
Order
Cancel

Figure 7.6

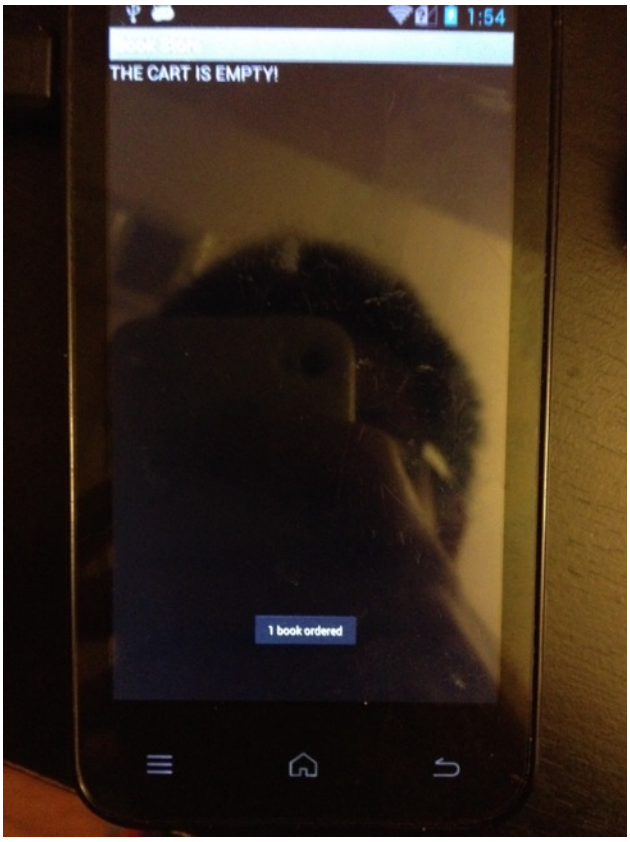


Figure 7.7

Part 2: Basic Chat App

(1) Chat-Server-Oneway Project

1. In ChatServer.java, first declare the ListView, define an ArrayList<String> for storing the message and an ArrayAdapter<String> to show messages in the ListView, as Figure 1.1

```java
/*
 * Socket used both for sending and receiving
 */
private DatagramSocket serverSocket;

/*
 * True as long as we don't get socket errors
 */
private boolean socketOK = true;

/*
 * TODO: Declare UI.
 */
private ListView messageList;

private ArrayList<String> list;
private ArrayAdapter<String> arrayAdapter;

/*
 * End Todo
 */

Button next;
```

Figure 1.1

In the onCreate() method, first initialize the UI, set the adapter for the ListView as Figure 1.2

```java
/*
 * Called when the activity is first created.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /**
     * Let's be clear, this is a HACK to allow you to do network communication on the main thread.
     * This WILL cause an ANR, and is only provided to simplify the pedagogy.  We will see how to do
     * this right in a future assignment (using a Service managing background threads).
     */
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

    try {
        /*
         * Get port information from the resources.
         */
        int port = Integer.parseInt("6666");
        serverSocket = new DatagramSocket(port);
    } catch (Exception e) {
        Log.e(TAG, "Cannot open socket" + e.getMessage());
        return;
    }

    /*
     * TODO: Initialize the UI.
     */
    messageList = (ListView)findViewById(R.id.msgList);
    next = (Button)findViewById(R.id.next);
    list = new ArrayList<String>();
    arrayAdapter = new ArrayAdapter<String>(this, R.layout.message, list);
    messageList.setAdapter(arrayAdapter);


    /*
     * End Todo
     */

}
```

Figure 1.2

Implement the onClick() method, receive the packet from the DatagramSocket, use the getData() method to receive the byte arrays, change it into String and add to the ArrayList, then use notifyDataSetChange() method to tell the arrayAdapter to flush the ListView, as Figure 1.3

```
public void onClick(View v) {

    byte[] receiveData = new byte[1024];

    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

    try {

        serverSocket.receive(receivePacket);
        Log.i(TAG, "Received a packet");

        InetAddress sourceIPAddress = receivePacket.getAddress();
        Log.i(TAG, "Source IP Address: " + sourceIPAddress);

        /*
         * TODO: Extract sender and receiver from message and display.
         */
        receiveData = receivePacket.getData();
        String temp = new String(receiveData, 0, receivePacket.getLength());
        if (!temp.isEmpty()) {
            list.add(temp);
            arrayAdapter.notifyDataSetChanged();
        }

        /*
         * End Todo
         */

    } catch (Exception e) {

        Log.e(TAG, "Problems receiving packet: " + e.getMessage());
        socketOK = false;
    }

}
```

Figure 1.3

Finally, override the onStop() method to implement the closeSocket() method to stop the socket, as Figure 1.4

```
    @Override
    protected void onStop() {
        if (socketIsOK()) {
            closeSocket();
        }
        super.onStop();
    }

    /*
     * Close the socket before exiting application
     */
    public void closeSocket() { serverSocket.close(); }

    /*
     * If the socket is OK, then it's running
     */
    boolean socketIsOK() { return socketOK; }

}
```

Figure 1.4

2. In the main.xml layout file, set the android:onClick attribute of the Button to “onClick” method, as Figure 2.1 and Figure 2.2.

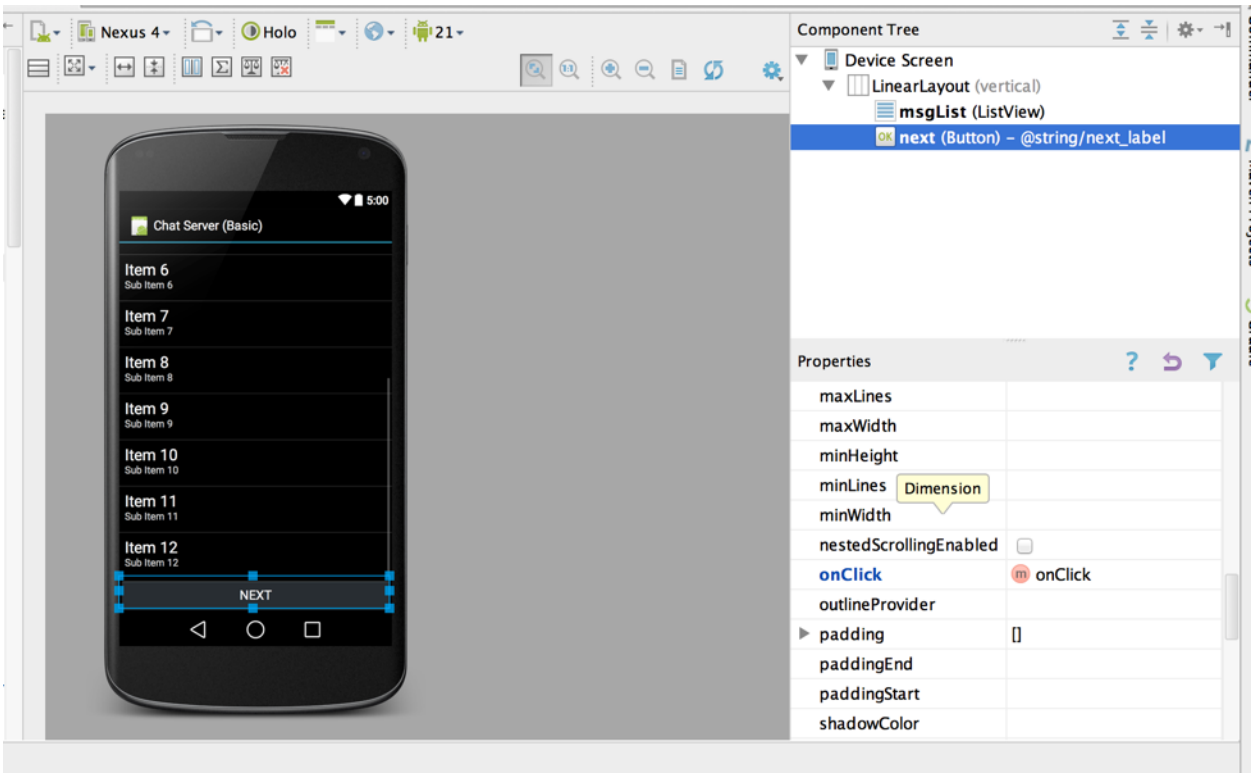

Nexus 4
Holo
21
Component Tree
Device Screen
LinearLayout (vertical)
msgList (ListView)
next (Button) – @string/next_label
5:00
Chat Server (Basic)
Item 6
Sub Item 6
Item 7
Sub Item 7
Item 8
Sub Item 8
Item 9
Sub Item 9
Item 10
Sub Item 10
Item 11
Sub Item 11
Item 12
Sub Item 12
NEXT
Properties
maxLines
maxWidth
minHeight
minLines
Dimension
minWidth
nestedScrollingEnabled
onClick
onClick
outlineProvider
padding
[]
paddingEnd
paddingStart
shadowColor

Figure 2.1

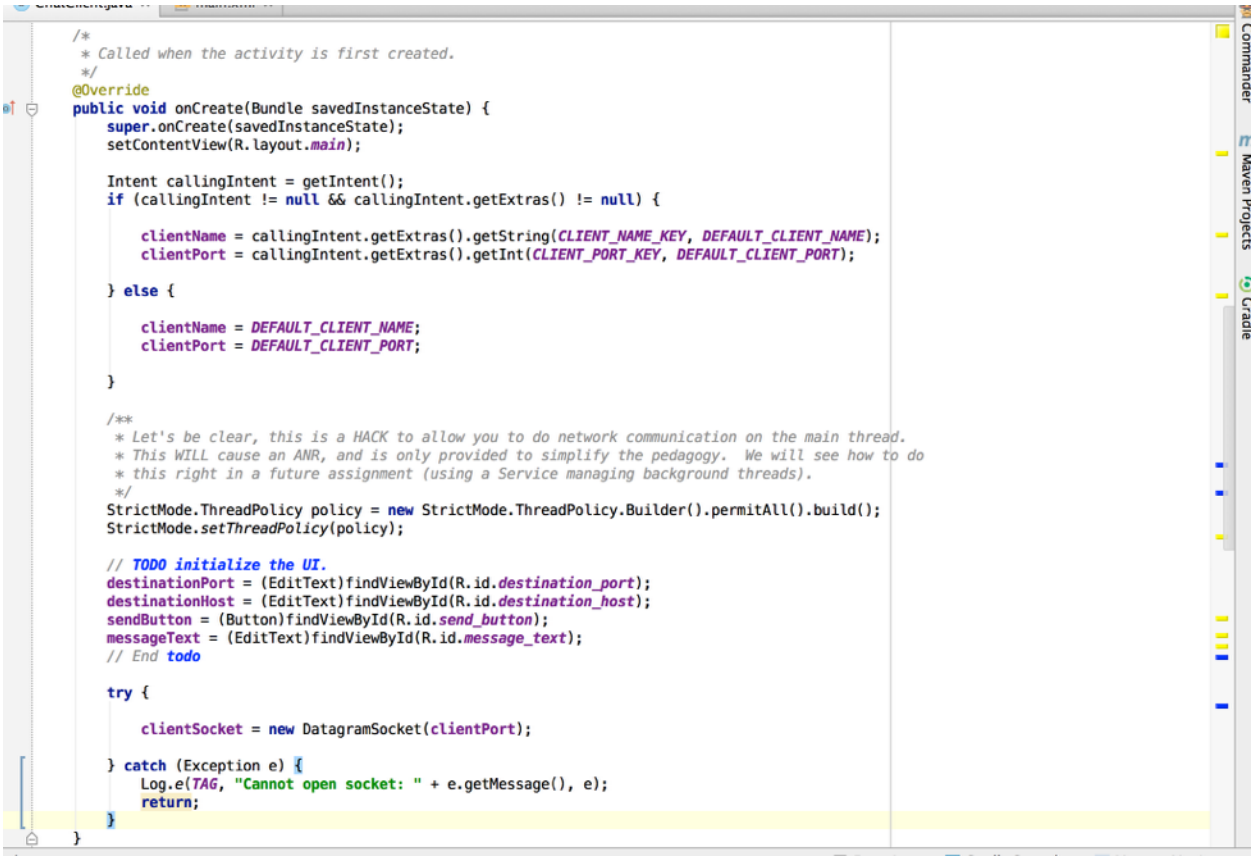
message.xml × main.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/msgList"
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        android:stackFromBottom="true"
        android:transcriptMode="alwaysScroll" >
    </ListView>

    <Button
        android:id="@+id/next"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="NEXT"
        android:onClick="onClick" />

</LinearLayout>
```

Figure 2.2

(2) Chat-Client-Oneway Project

1. In ChatClient.java, initialize the UI in onCreate() method including the destinationPort EditText, destinationHost EditText, messageText EditText and the Button as Figure 1.1



```
/*
 * Called when the activity is first created.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Intent callingIntent = getIntent();
    if (callingIntent != null && callingIntent.getExtras() != null) {

        clientName = callingIntent.getExtras().getString(CLIENT_NAME_KEY, DEFAULT_CLIENT_NAME);
        clientPort = callingIntent.getExtras().getInt(CLIENT_PORT_KEY, DEFAULT_CLIENT_PORT);

    } else {

        clientName = DEFAULT_CLIENT_NAME;
        clientPort = DEFAULT_CLIENT_PORT;

    }

    /**
     * Let's be clear, this is a HACK to allow you to do network communication on the main thread.
     * This WILL cause an ANR, and is only provided to simplify the pedagogy.  We will see how to do
     * this right in a future assignment (using a Service managing background threads).
     */
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

    // TODO initialize the UI.
    destinationPort = (EditText)findViewById(R.id.destination_port);
    destinationHost = (EditText)findViewById(R.id.destination_host);
    sendButton = (Button)findViewById(R.id.send_button);
    messageText = (EditText)findViewById(R.id.message_text);
    // End todo

    try {

        clientSocket = new DatagramSocket(clientPort);

    } catch (Exception e) {
        Log.e(TAG, "Cannot open socket: " + e.getMessage(), e);
        return;
    }
}
```

Figure 1.1

In the onClick() method, first get the data from the EditText including the destination host, port and the message to be sent as Figure 1.2

```
public void onClick(View v) {
    try {
        /*
         * On the emulator, which does not support WIFI stack, we'll send to
         * (an AVD alias for) the host loopback interface, with the server
         * port on the host redirected to the server port on the server AVD.
         */

        InetAddress destAddr = null;


        int destPort = 0;

        byte[] sendData = null;  // Combine sender and message text; default encoding is UTF-8

        // TODO get data from UI

        destPort = Integer.parseInt(destinationPort.getText().toString());
        String host = destinationHost.getText().toString();
        Log.v("Host", host);
        destAddr = InetAddress.getByName(host);
        Log.v("destAddr", destAddr.getHostName());

        sendData = messageText.getText().toString().getBytes(Charset.forName("UTF-8"));
        // End todo

        DatagramPacket sendPacket = new DatagramPacket(sendData,
                sendData.length, destAddr, destPort);

        clientSocket.send(sendPacket);

        Log.i(TAG, "Sent packet: " + messageText);


    } catch (UnknownHostException e) {
        Log.e(TAG, "Unknown host exception: ", e);
    } catch (IOException e) {
        Log.e(TAG, "IO exception: ", e);
    }


    messageText.setText("");
}
}
```

Figure 1.2

Finally, same as the Chat Server, override onStop() method, implement the closeSocket to close the socket safely, as Figure 1.3

```
@Override
protected void onStop() {
    if (socketIsOK()) {
        closeSocket();
    }
    super.onStop();
}

/*
 * Close the socket before exiting application
*/
public void closeSocket() { clientSocket.close(); }

/*
 * If the socket is OK, then it's running
 */
boolean socketIsOK() { return socketOK; }
}
```

Figure 1.3

2. In the main.xml layout file, set the android:onClick attribute of the Button to “onClick” method, as Figure 2.1 and Figure 2.2.

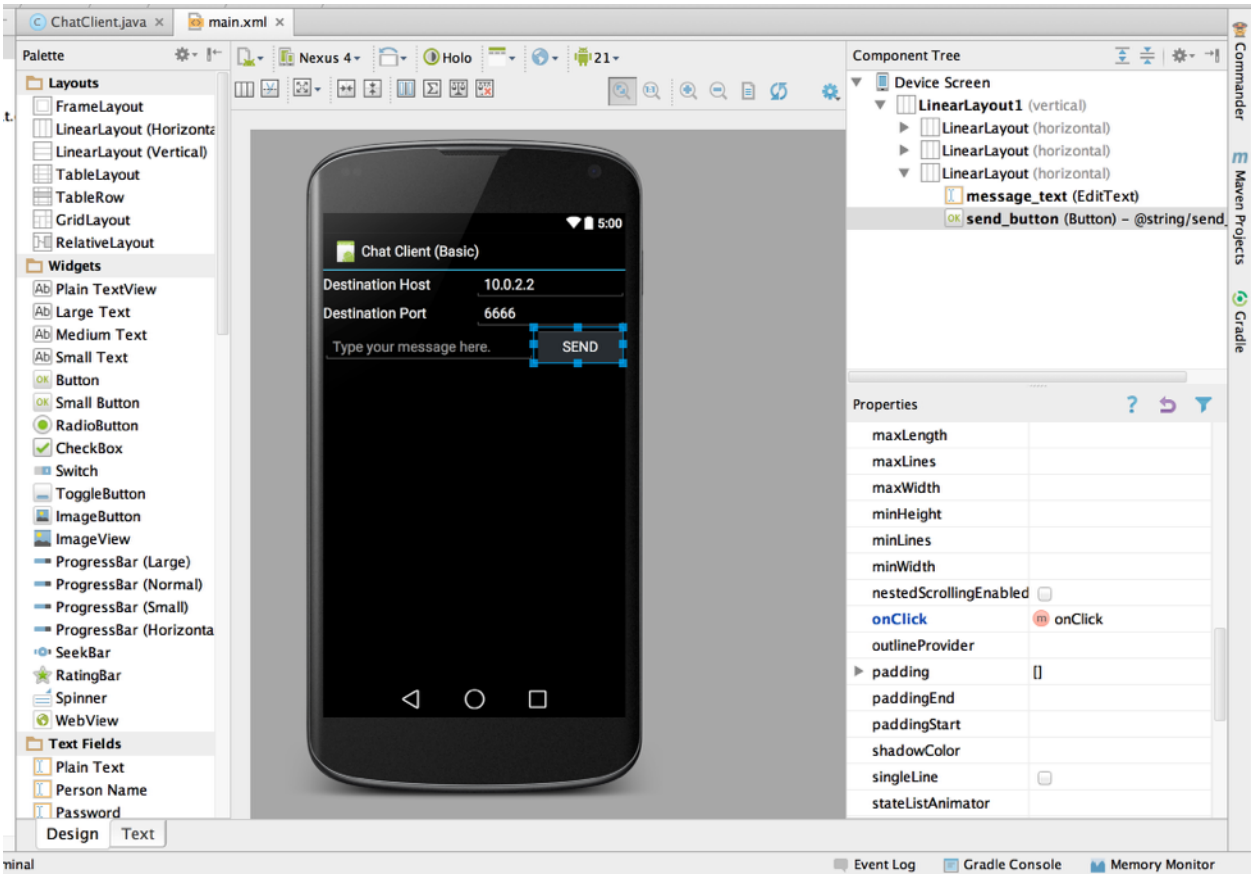


Figure 2.1

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/message_text"
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_weight="0.7"
        android:hint="Type your message here."
        android:inputType="text"
        android:singleLine="true"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <Button
        android:id="@+id/send_button"
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_weight="0.3"
        android:text="SEND"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:onClick="onClick" />

</LinearLayout>
</LinearLayout>
```

Figure 2.2

(3) Test the Server and Client

1. To test the Chat Server and Client, I run the Server App on my real Android Phone where the IP address is 192.168.1.13, and create an Android Virtual Device to run the Client App. The Chat process is displayed as follows:

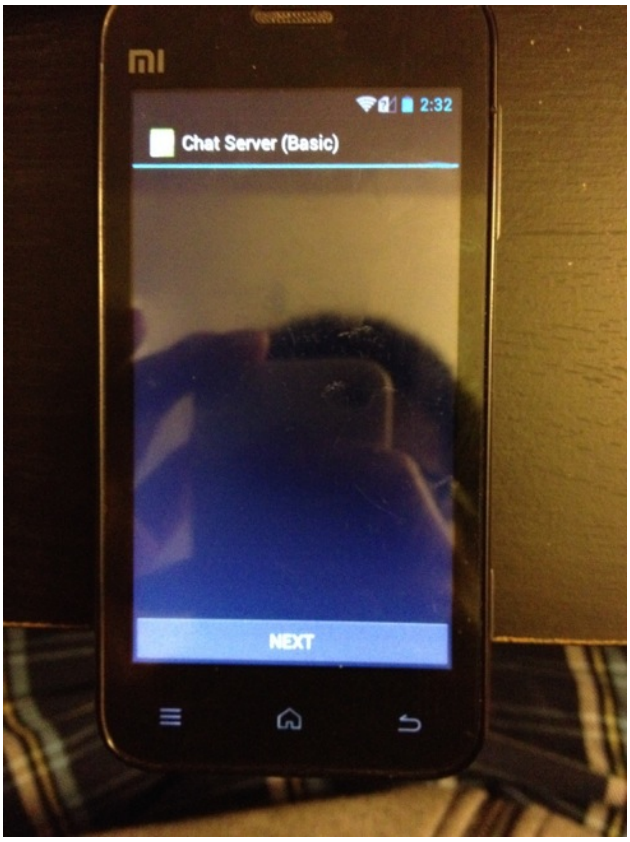


Figure 1.1

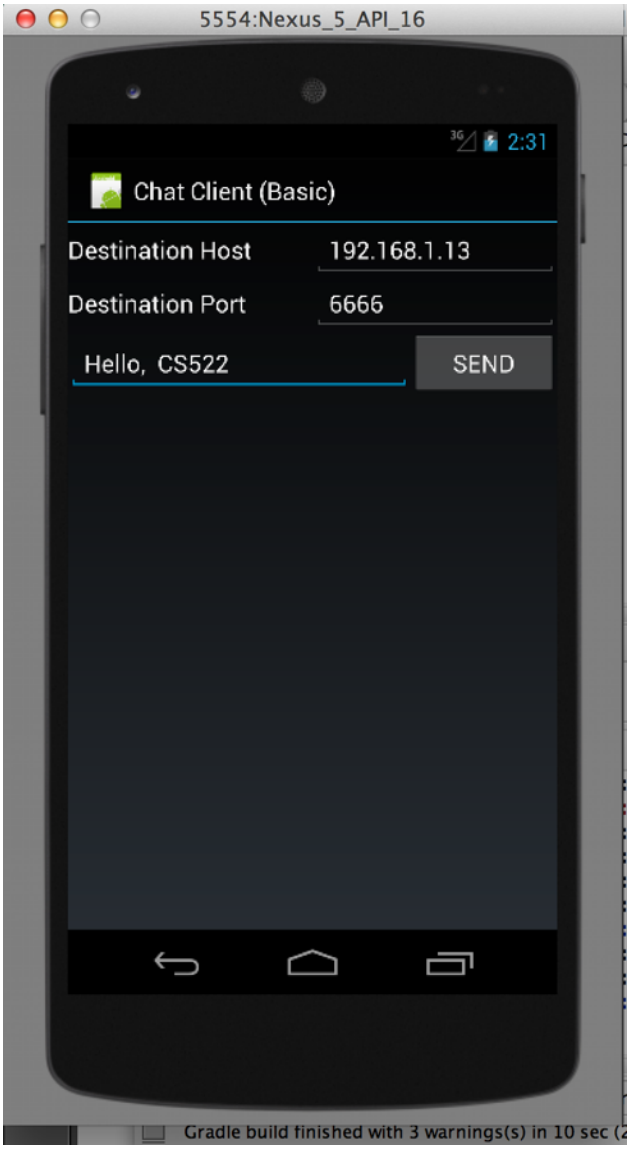

5554:Nexus_5_API_16
2:31
Chat Client (Basic)
Destination Host
192.168.1.13
Destination Port
6666
Hello, CS522
SEND
Gradle build finished with 3 warnings(s) in 10 sec

Figure 1.2

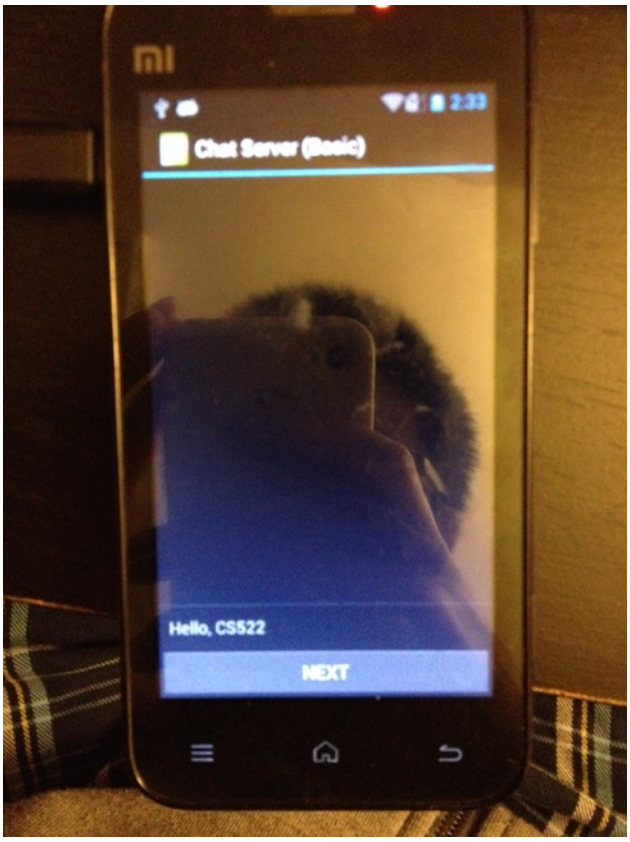


Figure 1.3

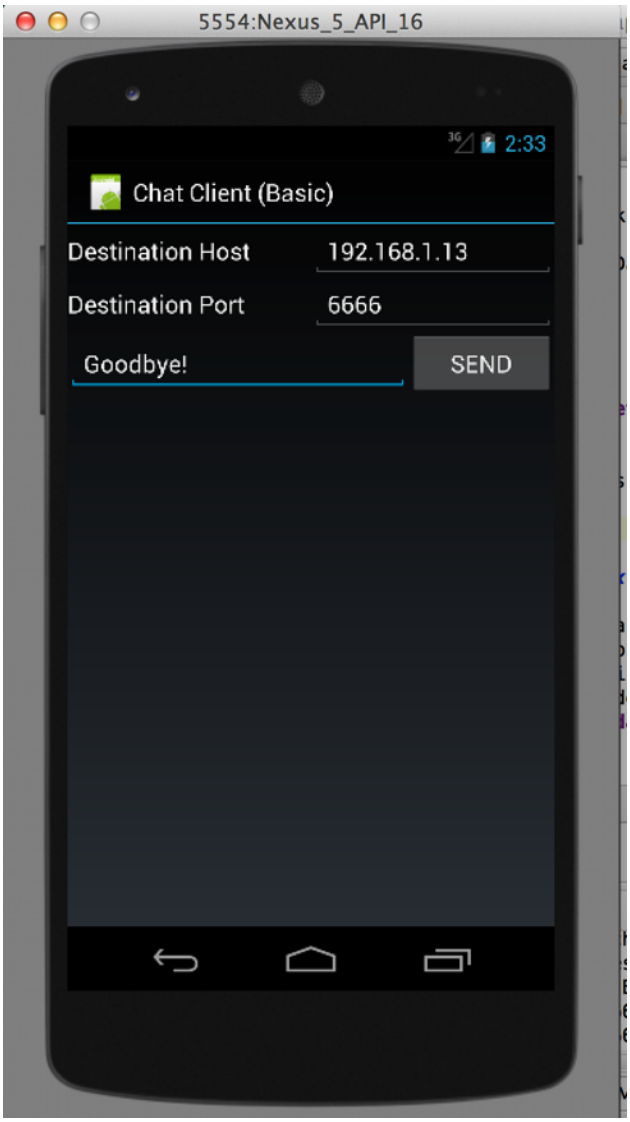

5554:Nexus_5_API_16
2:33
Chat Client (Basic)
Destination Host
192.168.1.13
Destination Port
6666
Goodbye!
SEND

Figure 1.4

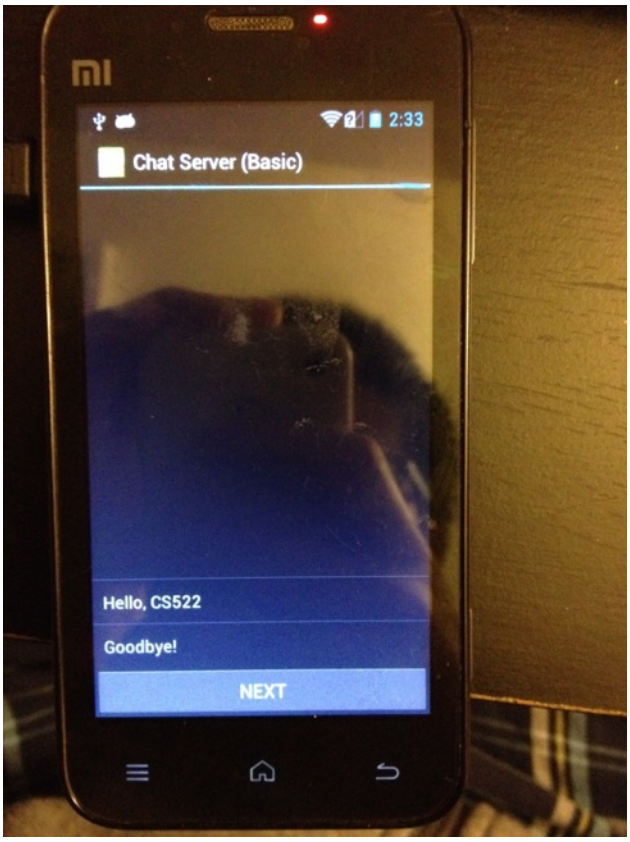


Figure 1.5