Business and Point of Sale App
Using Classes and Lists to Organize Data
The Coffee App

Description:

This app uses a multiple class structure to create an interface where a user can select options for ordering a given product. The goals of this App are to design class that will keep track of the data points surrounding a product, design a class that will keep a database of the instances of the product as they are ordered, and design a user interface where the user to will select the points of data for the product during a Point of Sale Experience. We will also use a new xml object called a Spinner that allows us to pick items from a list (like a drop down menu).

For this example we will use coffee as the Product.

Overview of App:

In designing a Point of Sale App, we need to define the product and the fields that describe the product. For our coffee product, we will define the following:

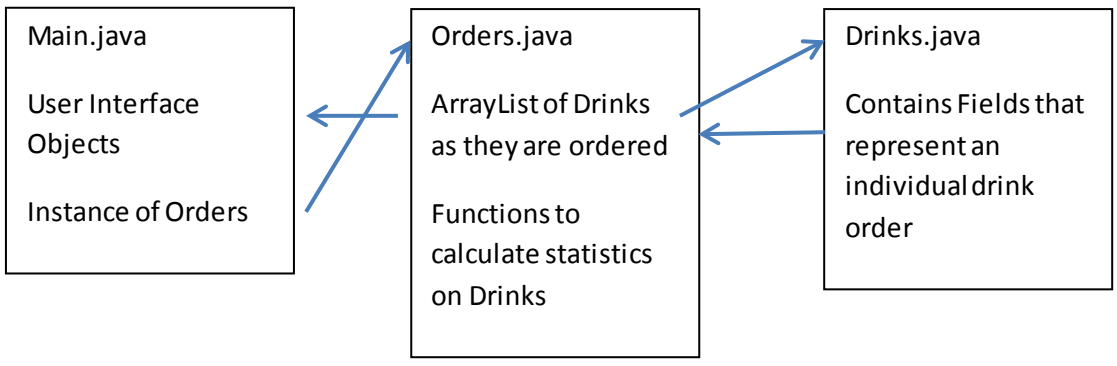
Class Drink

Fields:
Boolean hot: Hot or Cold Drink
String type: Coffee, Tea, Frappuccino, Espresso . . .
String flavor: Mocha, Vanilla . . .
String dairy: Type of dairy product; Milk, Half and Half . . .
Integer size: small, medium, large . . .
Date date: time ordered
Boolean served: was the drink served to the customer

Constructors:
Simple Constructor (No Fields initialized)

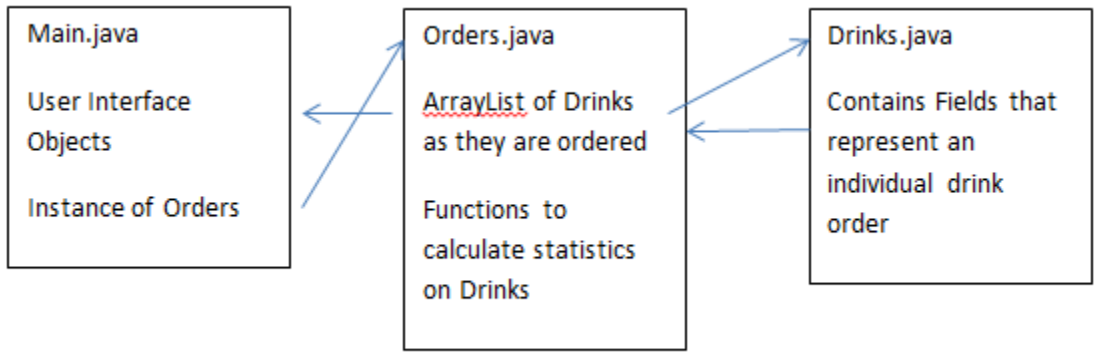
Functions:
Getters and Setters for all Fields

The Drink class will be the core object that the app will track. We will have two additional classes that will manipulate Drink instances:

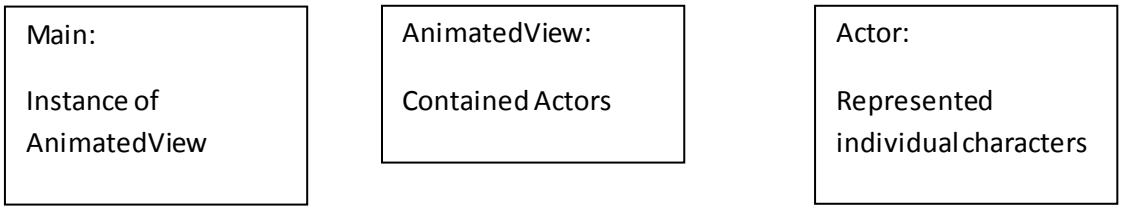


Think of this as being similar to the Main, AnimationView, Actor setup we had in animated games:

Drink App Classes:



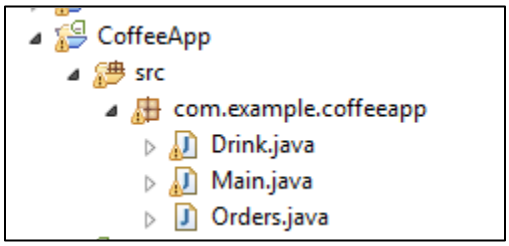
Bouncing App Classes:



Process:

Part 1: Create New App and classes Main, Orders, and Drink. Define the Drink class fields, accessors, and modifiers.

1. Open Eclipse and start a new Android Application Project.
    a. Name the Project: "Lastname Coffee App"
    b. Name the main activity "Main"
    c. Everything else is default.
2. Expand the src folder and create two additional classes (right click on package name and select new class). Create the classes 'Orders' and 'Drink"



3. Open the Drink class and define the fields for the Drink object:

```
 4
 5 public class Drink {
 6     // Fields
 7     private boolean hot;  // Hot or Cold
 8     private String type;  // Coffee, Tea . . .
 9     private String flavor;// Mocha . . .
10     private String topping;// Drizzle . ..
11     private String dairy;  // Milk, Soy . . .
12     private int size;      // In ounces
13     private String instructions; // Special instructions
14     private Date date;     // Date and Time ordered
15     private boolean served; // Was this drink served
16
```

4. Write the constructor for Drink. Note that we will not initialize any fields here.

```
16
17     // Constructor - no initialization of fields
18     public Drink() {
19
20     } // End Constructor
21
```

5. Write a second constructor that will take parameters and initialize the fields.

```
21
22      // Second Constructor with fields
23      public Drink(boolean h, String t, String f, String tp,
24                   String d, int sz, String ins) {
25          hot = h;
26          type = t;
27          flavor = f;
28          topping = t;
29          dairy = d;
30          size = sz;
31          instructions = ins;
32          served = false;
33      } // End Second constructor
34
```

6. Write the Modifiers for the Fields:

```
36      // Modifiers
37      public void setHot(boolean h) {
38          hot = h;
39      }
40
41      public void setType(String t) {
42          type = t;
43      }
44
45      public void setFlavor(String f) {
46          flavor = f;
47      }
48
49      public void setDairy(String d) {
50          dairy = d;
51      }
52
53      public void setSize(int s) {
54          size = s;
55      }
56
57      public void setInstructions(String i) {
58          instructions = i;
59      }
60
61      public void setDate(Date d) {
62          date = d;
63      }
```

7. Write the Accessors for the Fields:

```
65      // Accessors
66      public boolean getHot() {
67          return hot;
68      }
69
70      public String getType() {
71          return type;
72      }
73
74      public String getFlavor() {
75          return flavor;
76      }
77
78      public String getTopping() {
79          return topping;
80      }
81
82      public String getDairy() {
83          return dairy;
84      }
85
86      public int getSize() {
87          return size;
88      }
89
90      public Date getDate() {
91          return date;
92      }
93
94      public boolean getServed() {
95          return served;
96      }
97
98      public void setServed(boolean s) {
99          served = s;
100     }
101
102 } // end class Drink
103
```

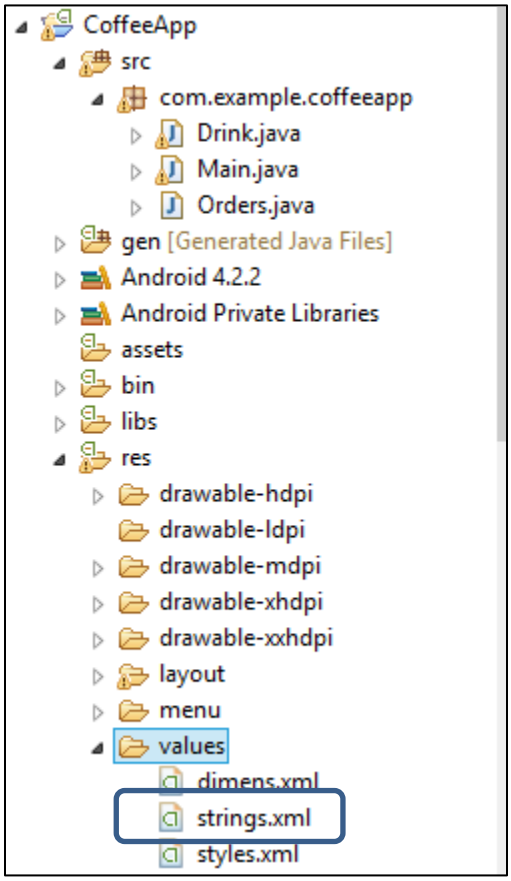
8. This finished the Drink class. Note that the Drink class holds the data about the drinks we will create during the App. Now we need to write a class that will 'hold' all the drinks.

Part 2: Define the Orders class and string resources for Spinner objects and build User Interface

9. Go to the Orders class. This class is short as it revolves around an ArrayList object that will store the drinks. Enter the code below:

```
package com.example.coffeeapp;

import java.util.ArrayList;
import java.util.List;

public class Orders {
	// Field (Only the List)
	private List <Drink> drinks;

	// Constructor
	public Orders() {
		drinks = new ArrayList <Drink> (0);
	} // end constructor

	// Accessor
	public List<Drink> getDrinks() {
		return drinks;
	}

	// Add a drink to the List
	public void addDrink(Drink d) {
		drinks.add(d);
	}

	// Get the total drinks served
	public int getNumServed() {
		int total = 0;
		for (int d = 0; d < drinks.size(); d++) {
			if (drinks.get(d).getServed() == true) {
				total++;
			}
		}
		return total;
	}

	// Get an individual order
	public Drink getDrink(int i) {
		return drinks.get(i);
	}

	// Get the Drinks ordered
	public int getNumDrinks() {
		return drinks.size();
	}

} // End class Orders
```

10. We now need to write some XML to store the values that the Spinners will have. Find the res/values/strings.xml file and open it.

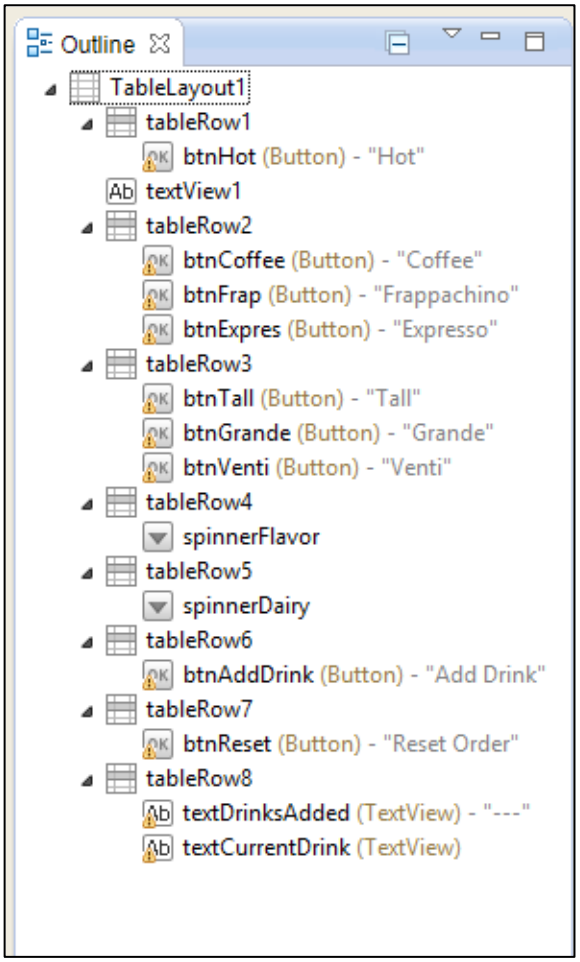


11. The strings.xml file holds words (String objects) that the App can refer to while it is running. This allows you to modify String objects used around the App without having to reach into each class. Add the lines of code corresponding to the <string-array> tags:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">CoffeeApp</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

    <string-array name = "flavor_array">
        <item>None</item>
        <item>Mocha</item>
        <item>Vanilla</item>
        <item>Caramel</item>
    </string-array>

    <string-array name = "dairy_array">
        <item>None</item>
        <item>Half and Half</item>
        <item>Skim</item>
        <item>Soy</item>
    </string-array>

</resources>
```

12. Save the strings.xml file and go to the activity_main.xml to start work on the User Interface.

13. Note the design and layout of the User Interface below. Set the layout to TableLayout and use TableRow objects to create a user interface with the objects as named in the outline view.



14. We now need to set the android:onClick values for the Buttons to call functions we will write in the Main class. The entire xml is shown below. Add the android:onClicks that are highlighted.

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/TableLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Main" >

    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnHot"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick = "btnHotClicked"
            android:text="Hot" />

    </TableRow>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <TableRow
        android:id="@+id/tableRow2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnCoffee"
            android:layout_height="wrap_content"
            android:layout_weight="0.2"
            android:onClick = "coffeeClicked"
            android:text="Coffee" />
```

```
        <Button
            android:id="@+id/btnFrap"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.2"
            android:onClick = "frapClicked"
            android:text="Frappachino" />

        <Button
            android:id="@+id/btnExpres"
            android:layout_height="wrap_content"
            android:layout_weight="0.2"
            android:onClick = "expresClicked"
            android:text="Expresso" />

    </TableRow>


    <TableRow
        android:id="@+id/tableRow3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnTall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick = "tallClicked"
            android:text="Tall" />

        <Button
            android:id="@+id/btnGrande"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick = "grandeClicked"
            android:text="Grande" />

        <Button
            android:id="@+id/btnVenti"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick = "ventiClicked"
            android:text="Venti" />

    </TableRow>
```

```
    <TableRow
        android:id="@+id/tableRow4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Spinner
            android:id="@+id/spinnerFlavor"
            android:layout_height="wrap_content"
            android:layout_weight="0.25" >



         </Spinner>

    </TableRow>

    <TableRow
        android:id="@+id/tableRow5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Spinner
            android:id="@+id/spinnerDairy"
            android:layout_height="wrap_content"
            android:layout_weight="0.25" />

    </TableRow>

    <TableRow
        android:id="@+id/tableRow6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnAddDrink"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="addDrinkClicked"
            android:text="Add Drink" />

    </TableRow>

    <TableRow
        android:id="@+id/tableRow7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
```

```
        <Button
            android:id="@+id/btnReset"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:onClick = "resetDrink"
            android:text="Reset Order" />

    </TableRow>

    <TableRow
        android:id="@+id/tableRow8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textDrinksAdded"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="---" />

        <TextView
            android:id="@+id/textCurrentDrink"
            android:layout_width="125dp"
            android:layout_height="wrap_content"
            android:text="TextView" />

    </TableRow>

</TableLayout>
```

15. Go to the Main class. We now need to define fields for all the XML objects.

```
13
14 public class Main extends Activity {
15     // Field to hold Order data
16     private Orders orders;
17     private Drink currentDrink;
18
19     // Fields for User Interface Objects
20     private Button btnHot;
21     private Button btnCoffee;
22     private Button btnFrap;
23     private Button btnExpres;
24     private Button btnTall;
25     private Button btnGrande;
26     private Button btnVenti;
27     private Spinner spinnerFlavor;
28     private Spinner spinnerDairy;
29     private Button btnAddDrink;
30     private Button btnResetDrink;
31     private TextView textDrinksAdded;
32     private TextView textCurrentDrink;
33
```

16. Go to the onCreate and bind the user interface objects to the XML fields:

```
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_main);
38
39         // Initialize Orders
40         orders = new Orders();
41         currentDrink = new Drink();
42
43         // Bind to XML
44         btnHot = (Button)findViewById(R.id.btnHot);
45         btnCoffee = (Button)findViewById(R.id.btnCoffee);
46         btnFrap = (Button)findViewById(R.id.btnFrap);
47         btnExpres = (Button)findViewById(R.id.btnExpres);
48         btnTall = (Button)findViewById(R.id.btnTall);
49         btnGrande = (Button)findViewById(R.id.btnGrande);
50         btnVenti = (Button)findViewById(R.id.btnVenti);
51         spinnerFlavor = (Spinner)findViewById(R.id.spinnerFlavor);
52         spinnerDairy = (Spinner)findViewById(R.id.spinnerDairy);
53         btnAddDrink = (Button)findViewById(R.id.btnAddDrink);
54         btnResetDrink = (Button)findViewById(R.id.btnReset);
55         textDrinksAdded = (TextView)findViewById(R.id.textDrinksAdded);
56         textCurrentDrink = (TextView)findViewById(R.id.textCurrentDrink);
57
```

17. Stay in onCreate(). We now need to attach the flavor and dairy arrays to the spinnerFlavor and spinnerDairy objects. First, we will create an ArrayAdapter flavorAdapter and attach it to the flavor_array.

```
57
58          // Populate the Spinner for Flavor
59          ArrayAdapter<CharSequence> flavorAdapter = ArrayAdapter.createFromResource(this,
60                  R.array.flavor_array, android.R.layout.simple_spinner_dropdown_item);
61
```

18. Now we will set the format for the dropdown view:

```
61
62          // Specify the layout to use when the list of choices appears
63          flavorAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
64
```

19. We now attach the flavorAdapter to the spinnerFlavor.

```
64
65          // Apply to the Spinner
66          spinnerFlavor.setAdapter(flavorAdapter);
67
```

20. Repeat the process for the spinnerDairy:

```
67
68          // Populate the Spinner for Dairy
69          ArrayAdapter<CharSequence> dairyAdapter = ArrayAdapter.createFromResource(this,
70                  R.array.dairy_array, android.R.layout.simple_spinner_dropdown_item);
71
72          dairyAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
73
74          spinnerDairy.setAdapter(dairyAdapter);
75
76      } // end onCreate
77
```

21. Write the function for the Hot/Cold button click event.

```
84
85      // Function for Hot/Cold Button
86      public void btnHotClicked(View v) {
87          if (btnHot.getText() == "Hot") {
88              btnHot.setText("Cold");
89              btnHot.setBackgroundColor(Color.BLUE);
90              currentDrink.setHot(false);
91          } // end if
92          else {
93              btnHot.setText("Hot");
94              btnHot.setBackgroundColor(Color.RED);
95              currentDrink.setHot(true);
96          } // end else
97      } // end btnHotClicked
98
```

22. Write the functions for the button events for drink type.

```
98
99      // Functions to react to Drink Types
100     public void coffeeClicked(View v) {
101         currentDrink.setType("Coffee");
102         btnCoffee.setBackgroundColor(Color.YELLOW);
103         btnFrap.setBackgroundColor(Color.LTGRAY);
104         btnExpres.setBackgroundColor(Color.LTGRAY);
105     }
106
107     public void frapClicked(View v) {
108         currentDrink.setType("Frappacino");
109         btnCoffee.setBackgroundColor(Color.LTGRAY);
110         btnFrap.setBackgroundColor(Color.YELLOW);
111         btnExpres.setBackgroundColor(Color.LTGRAY);
112
113     }
114
115     public void expresClicked(View v) {
116         currentDrink.setType("Expresso");
117         btnCoffee.setBackgroundColor(Color.LTGRAY);
118         btnFrap.setBackgroundColor(Color.LTGRAY);
119         btnExpres.setBackgroundColor(Color.YELLOW);
120     }
121
```

23. Write the functions for the drink size button click events:

```
121
122     // Functions for Drink Sizes
123     public void tallClicked(View v) {
124         currentDrink.setSize(8);
125         btnTall.setBackgroundColor(Color.GREEN);
126         btnGrande.setBackgroundColor(Color.LTGRAY);
127         btnVenti.setBackgroundColor(Color.LTGRAY);
128     }
129
130     public void grandeClicked(View v) {
131         currentDrink.setSize(12);
132         btnTall.setBackgroundColor(Color.LTGRAY);
133         btnGrande.setBackgroundColor(Color.GREEN);
134         btnVenti.setBackgroundColor(Color.LTGRAY);
135     }
136
137     public void ventiClicked(View v) {
138         currentDrink.setSize(20);
139         btnTall.setBackgroundColor(Color.LTGRAY);
140         btnGrande.setBackgroundColor(Color.LTGRAY);
141         btnVenti.setBackgroundColor(Color.GREEN);
142     }
143
```

24. The addDrinkClicked() function sets the flavors of the currentDrink object and then passes this object into the orders object. The function then displays the drink order number and data and then resets the UI.

```
143
144     // Button to add drink
145     public void addDrinkClicked(View v) {
146         // Set Flavor and Dairy from Spinners
147         currentDrink.setFlavor(String.valueOf(spinnerFlavor.getSelectedItem()));
148         currentDrink.setDairy(String.valueOf(spinnerDairy.getSelectedItem()));
149         // Add Drink to Orders
150         orders.addDrink(currentDrink);
151         // currentDrink = new Drink(); // Erase and load new drink
152         textDrinksAdded.setText(String.valueOf(orders.getNumDrinks()));
153         displayDrink(orders.getNumDrinks()-1);
154         resetDrink(v);
155     }
156
```

25. Now code the reset button event:

```
156
157     // Button for reset
158     public void resetDrink(View v) {
159         currentDrink = new Drink();
160         btnCoffee.setBackgroundColor(Color.LTGRAY);
161         btnFrap.setBackgroundColor(Color.LTGRAY);
162         btnExpres.setBackgroundColor(Color.LTGRAY);
163
164         btnTall.setBackgroundColor(Color.LTGRAY);
165         btnGrande.setBackgroundColor(Color.LTGRAY);
166         btnVenti.setBackgroundColor(Color.LTGRAY);
167     }
168
```

26. Finally, write the function to display a drink to the textCurrentDrink object.

```
168
169     private void displayDrink(int i) {
170         String sOrder = "Just ordered: ";
171         Drink dDrink = orders.getDrink(i);
172         sOrder += String.valueOf(dDrink.getSize()) + " ounces of ";
173         sOrder += dDrink.getType() + " with ";
174         sOrder += dDrink.getFlavor() + " and ";
175         sOrder += dDrink.getDairy() + ".";
176         // Display the Drink
177         textCurrentDrink.setText(sOrder);
178     }
179
180 } // end class Main
181
```

27. This app does not completely fill out the fields in the Drink object. Work to add Buttons and functions to allow for more detail.
28. Given this as a model, try creating your own ordering app for a product of your choice.