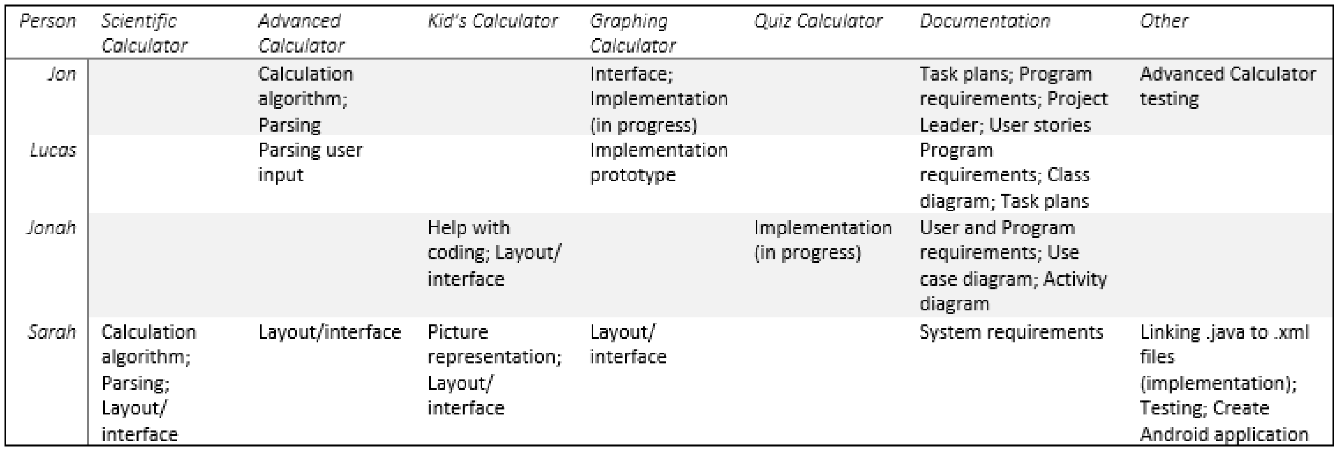
Midterm Evaluation for Calculator Project

Sarah Schulteis, Jon Jones, Jonah WaldKoenig, Lucas Garges

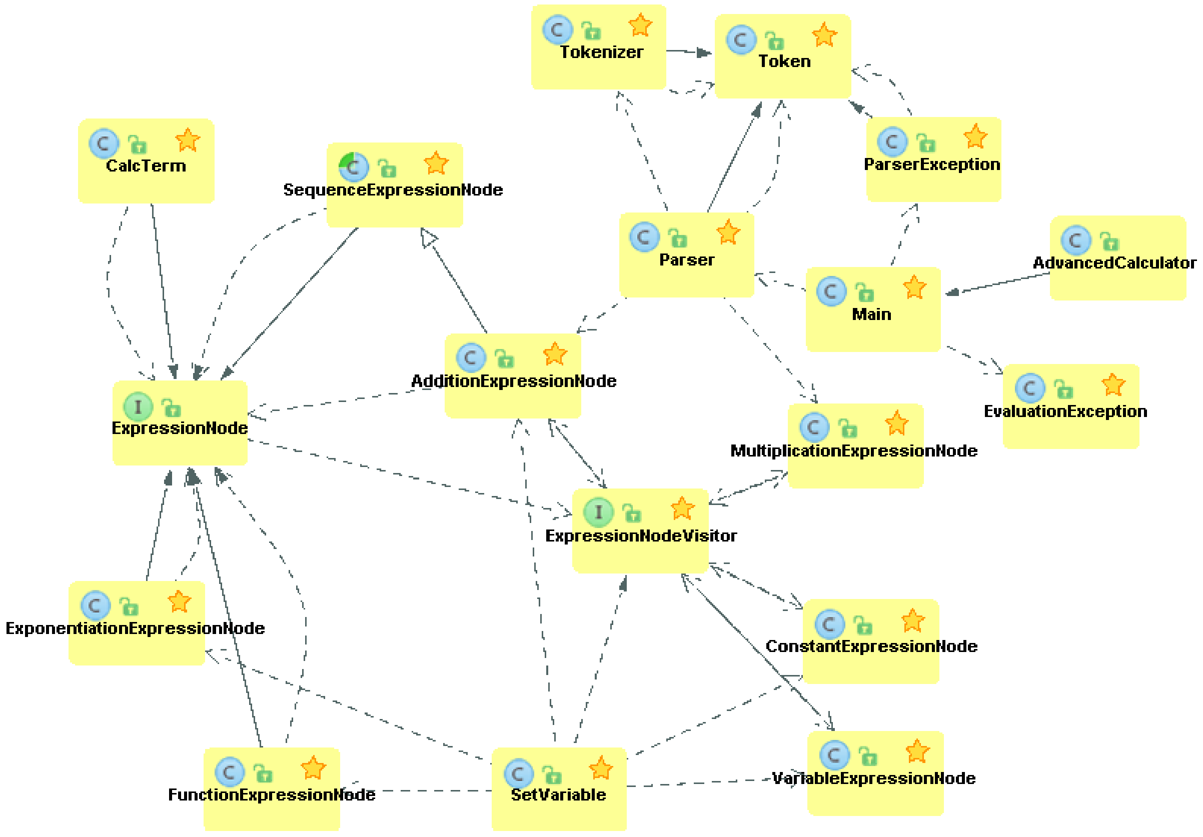
April 8, 2017

1 Group Contributions

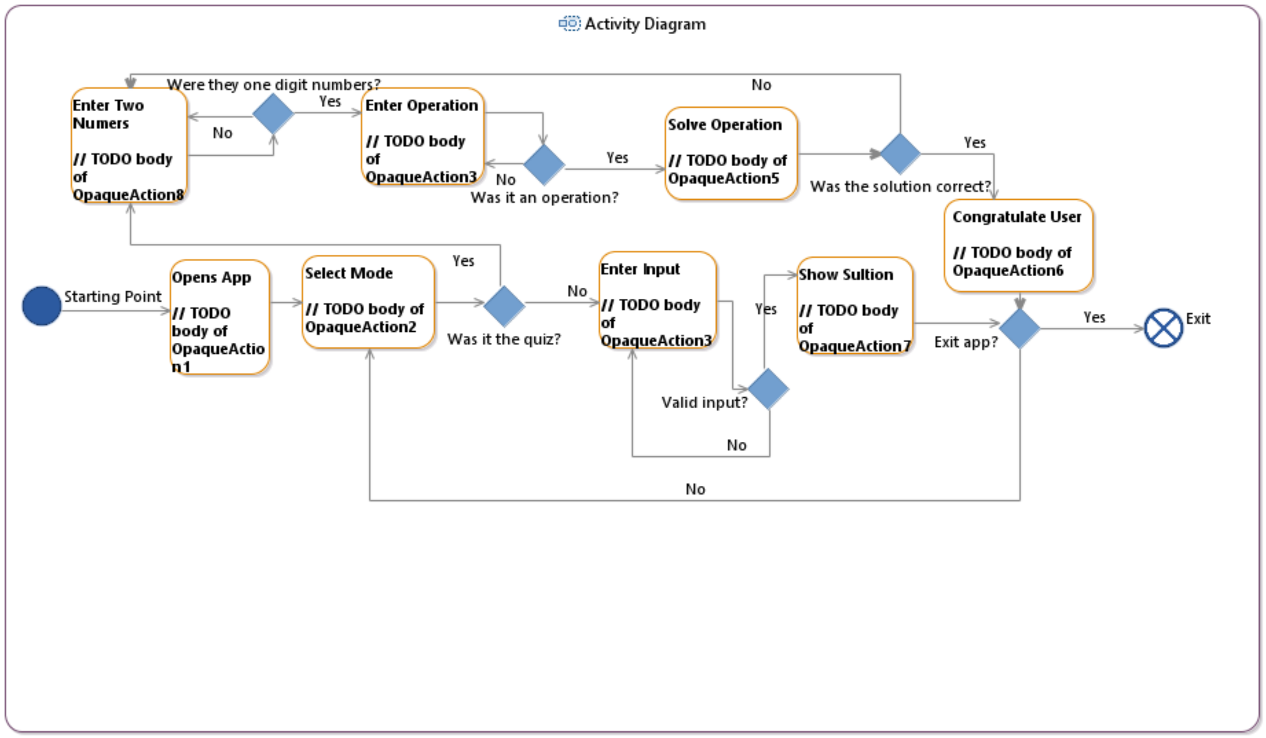


Person	*Scientific Calculator*	*Advanced Calculator*	*Kid's Calculator*	*Graphing Calculator*	*Quiz Calculator*	*Documentation*	*Other*
Jon		Calculation algorithm; Parsing		Interface; Implementation (in progress)		Task plans; Program requirements; Project Leader; User stories	Advanced Calculator testing
Lucas		Parsing user input		Implementation prototype		Program requirements; Class diagram; Task plans	
Jonah			Help with coding; Layout/ interface		Implementation (in progress)	User and Program requirements; Use case diagram; Activity diagram	
Sarah	Calculation algorithm; Parsing; Layout/ interface	Layout/interface	Picture representation; Layout/ interface	Layout/ interface		System requirements	Linking .java to .xml files (implementation); Testing; Create Android application

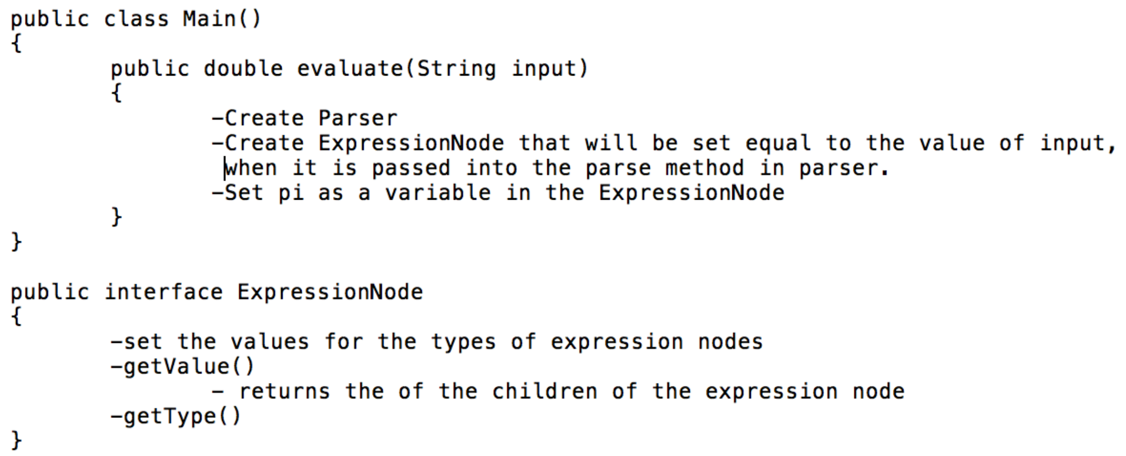
2 Class Diagram



3 Activity Diagram

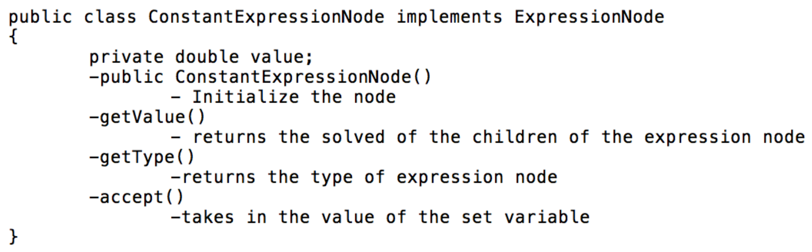

Activity Diagram
Were they one digit numbers?
Enter Two Numers
// TODO body of OpaqueAction8
No
Yes
Enter Operation
// TODO body of OpaqueAction3
No
Was it an operation?
Yes
Solve Operation
// TODO body of OpaqueAction5
No
Was the solution correct?
Yes
Congratulate User
// TODO body of OpaqueAction6
Starting Point
Opens App
// TODO body of OpaqueAction1
Select Mode
// TODO body of OpaqueAction2
Yes
Was it the quiz?
No
Enter Input
// TODO body of OpaqueAction3
Yes
Valid input?
No
Show Sultion
// TODO body of OpaqueAction7
Exit app?
Yes
Exit
No

4 Pseudo-Code for Math Algorithms

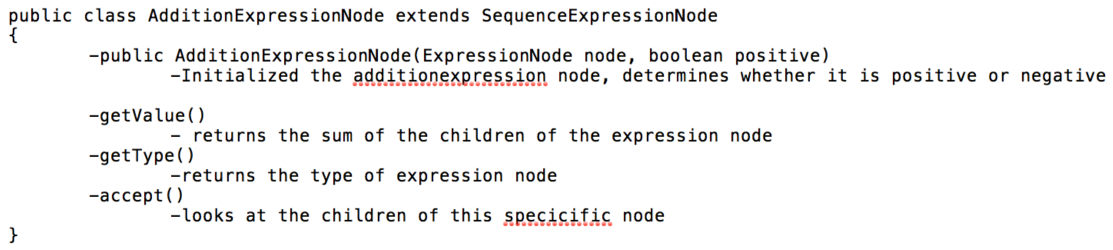


```
public class Main()
{
        public double evaluate(String input)
        {
                -Create Parser
                -Create ExpressionNode that will be set equal to the value of input,
                 when it is passed into the parse method in parser.
                -Set pi as a variable in the ExpressionNode
        }
}

public interface ExpressionNode
{
        -set the values for the types of expression nodes
        -getValue()
                - returns the of the children of the expression node
        -getType()
}
```



```
public class ConstantExpressionNode implements ExpressionNode
{
        private double value;
        -public ConstantExpressionNode()
                - Initialize the node
        -getValue()
                - returns the solved of the children of the expression node
        -getType()
                -returns the type of expression node
        -accept()
                -takes in the value of the set variable
}
```



```
public class AdditionExpressionNode extends SequenceExpressionNode
{
        -public AdditionExpressionNode(ExpressionNode node, boolean positive)
                -Initialized the additionexpression node, determines whether it is positive or negative

        -getValue()
                - returns the sum of the children of the expression node
        -getType()
                -returns the type of expression node
        -accept()
                -looks at the children of this specicific node
}
```

5 Major Java-class code Explanation

CalculatorActivity.java

When the app is launched, the CalculatorActiviy is launched. It displays a grid of buttons for basic calculations and an output for an equation to be entered into it. When a button is pushed, it calls the onClicked method which displays the corresponding text. When the equals button is pushed, it goes into the onClicked method which in turn calls the InfixToPostfix java class to display the result. The startKids, startAdvanced, startGraphing, and startQuiz methods are called when their corresponding buttons are clicked.

AdvancedCalculator.java

When the Advanced button is clicked in the CalculatorActivity, it calls the AdvancedCalculator.java file and launches the interface. Similarly to the previous one, there is an onClicked method that takes in and displays values. When the equals button is clicked, this calls the onClicked button which in turn calls the Main.java class to perform the execution.

KidsCalculator.java

As above, the KidsCalculator opens when the corresponding button is pressed on the CalculatorActivity. This class has a numbClick and a symClick methods which function very similarly to the onClicked explained in the previous .java classes. When each of these is called, they not only display the number, but also an equivalent number of pictures (apples) as a visual representation, done in setPic. The clearClick and calculation methods are helper methods to both clear the screen and display the output of the calculation.

GraphingCalculator.java

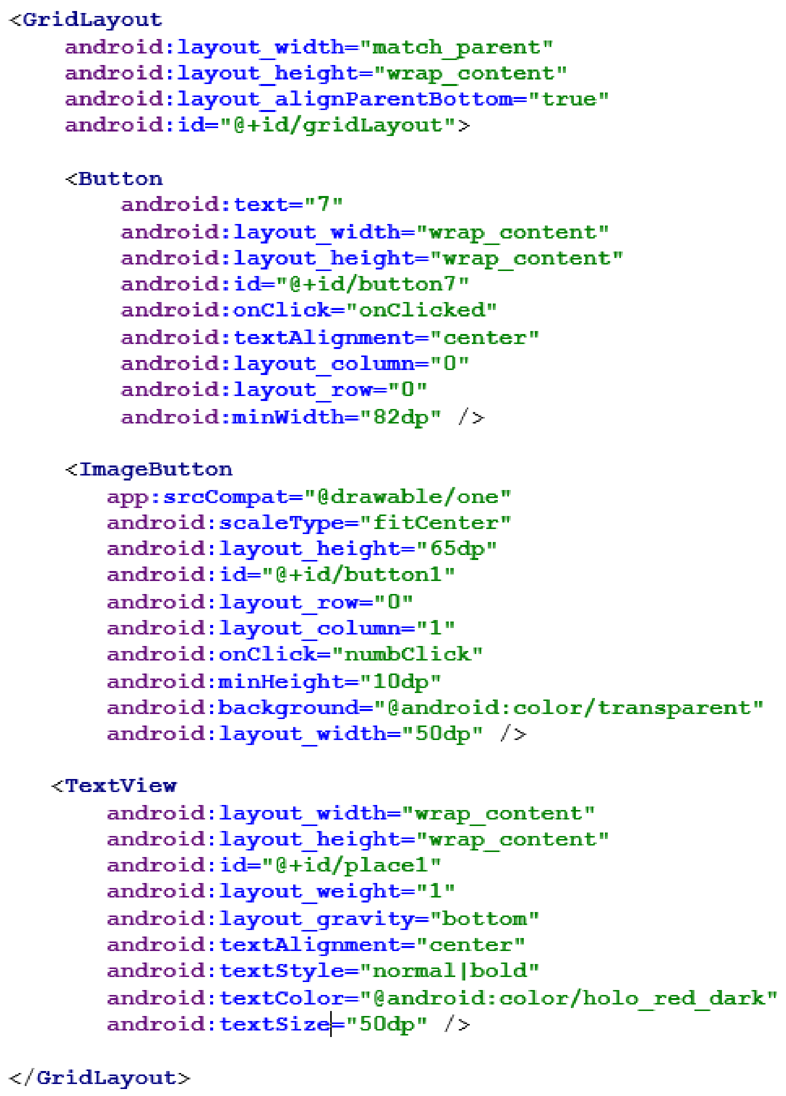
When the GraphingCalculator file is called, it displays a very basic graph. This calculator is still in the process of being created, but it currently does rudimentry point plotting.

User Interface (.xml) Explanation:

Listed below are a few pieces of code from a .xml file. The .xml files are the layout files which generate the user interface of an Android Application. To add to an interface, first a layout is chosen, such as GridLayout or HorizontalLayout and the specifications are set. In the below example, it is a GridLayout with parameters of match_parent and wrap_parent for the width and height. This means the width will stretch as far as possible and the height will keep as small as possible (based on what is put inside of it). It is also aligned at the bottom and given the name gridLayout.

Next is an example of a button named button7 with the text 7. Both of its layout parameters are set to wrap_content but must be at least 82dp wide. Since this is in the GridLayout, the row and column numbers are specified as 0,0. Setting the onClick parameter to onClicked means whenever the button is clicked, the method onClicked() will be called. The ImageButton shown below this button is very similar but with two exceptions. First, this button is an image found in the drawable file under the name one and the background color has been set.

The last example is a TextView, or just a line of text on the screen. This one is relatively self-explanatory as it has been given a name, height and width specifications, and information about the text. Information is given about the alignment, style, color, and size, along with the gravity (where the text sits in the box).



```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:id="@+id/gridLayout">

    <Button
        android:text="7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button7"
        android:onClick="onClicked"
        android:textAlignment="center"
        android:layout_column="0"
        android:layout_row="0"
        android:minWidth="82dp" />

    <ImageButton
       app:srcCompat="@drawable/one"
       android:scaleType="fitCenter"
       android:layout_height="65dp"
       android:id="@+id/button1"
       android:layout_row="0"
       android:layout_column="1"
       android:onClick="numbClick"
       android:minHeight="10dp"
       android:background="@android:color/transparent"
       android:layout_width="50dp" />

   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:id="@+id/place1"
       android:layout_weight="1"
       android:layout_gravity="bottom"
       android:textAlignment="center"
       android:textStyle="normal|bold"
       android:textColor="@android:color/holo_red_dark"
       android:textSize="50dp" />

</GridLayout>
```

Data Structures Used:

Stack
ArrayList
Queue

Data Structures Used:

Eclipse
Android Studio
Java JDK and JRE