**Data Structures**

* Wolframalpha: The search results from wolframalpha database will be stored in an arraylist.
  + Each entry is the array is a String.
  + Pseudocode for creating/getting info

createQuery(input)

{

Send the input

Get the output

for(!output)

parse thru object hiearchy

filter the XML portions for what we need

}

* Collecting point of gaze data from eye tracker and storing in Arraylist
  + Each entry is a Ellipse2D.Double
  + Pseudocode for parsing and adding point

Parse\_and\_add\_point (API response, screenX, screenY, ArrayList< Ellipse2d.Double > pts

{

Pattern x = Unique pattern for BPOGX and BPOGY

Set up matcher

if (matchFound)

Ellipse2d.Double cir = BPOGX and BPOGY

pts.add(cir)

}

* Training: This object has fields: concept, answers, correct)
  + Pseudocode for creating a puzzle object:

createProblem(concept, answers[4], correct )

return {

set the concept name being work on

set up the answers for multiple choice

set up correct answer

}

**API Interfaces**

**Java APIs**

• Java awt API

• Java swing API

The testing/training/search modules are simply a way to collect data from the user. Search provides whatever it is the user is trying to study. Training/Testing provide different methods of viewing the material and studying it while having eyes tracked.

**Eye Tracker APIs**

• Mirametrix S2 Eye Tracker API

• QuizMeSmart Program (made by previous student) to connect with java

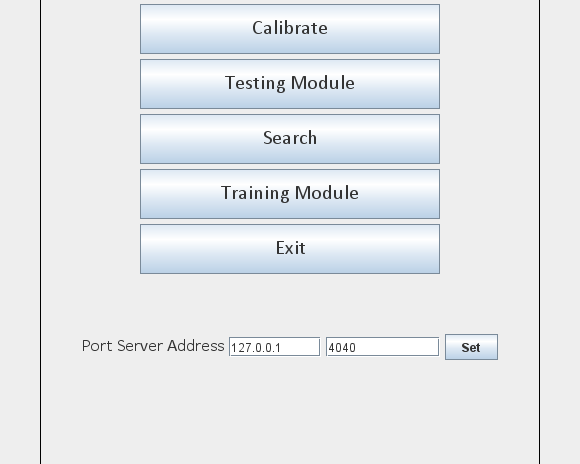
Setting up the TCP/IP server and connecting to it. The eye tracker gets the data regarding the gaze of the user. If the gaze of the user within the program screen, the program will track that time.

**WolframAlpha API**

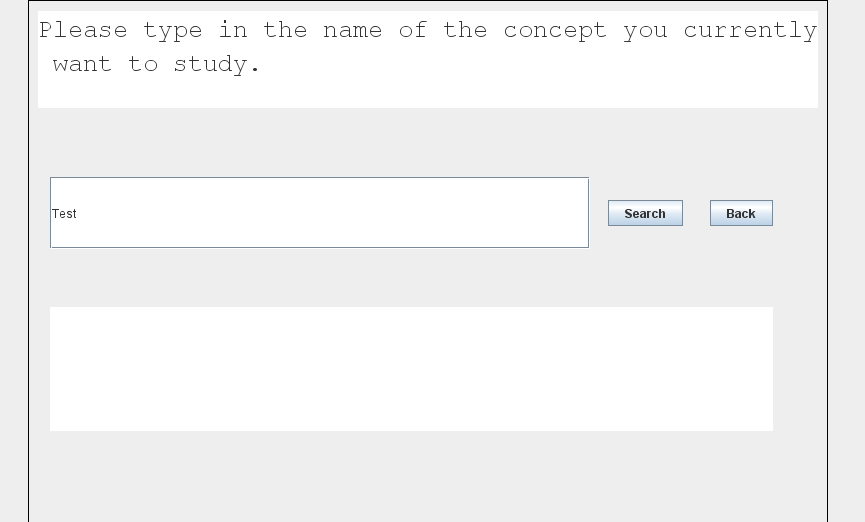
Use the wolfram alpha database to search for concepts and problems.

**User Interface:**

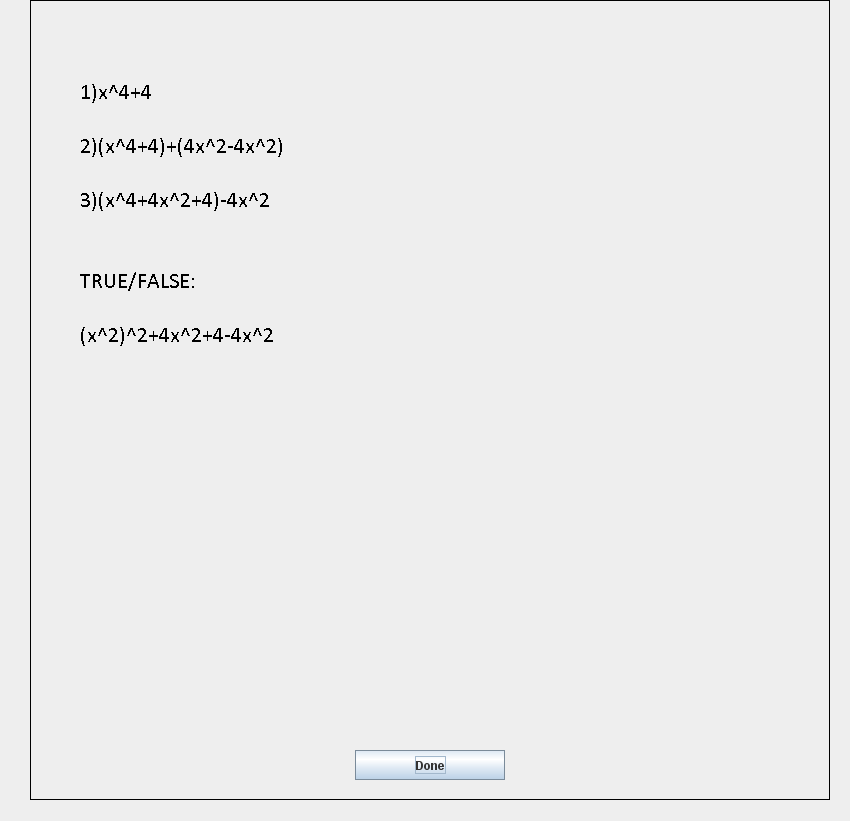
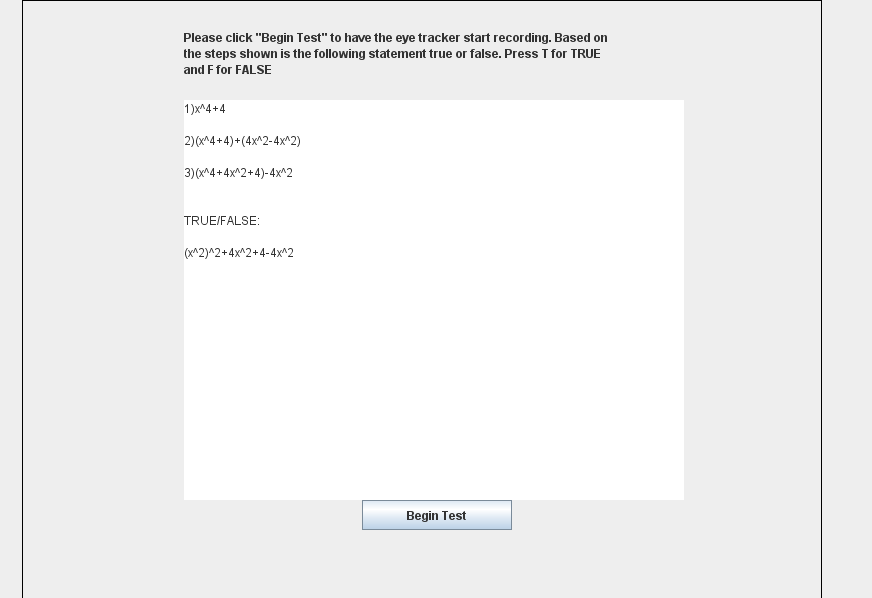
**Step 1: Main Screen**

****

**Step 2: Search**



**Step 3: Testing**



**Step 4: Training**

