Appendix A:
Script for Gild User Study

Participant Reads/Signs Consent Form

Participant Fills out Pre Study Questionnaire

Introduce Gild

- Resource View
- Editor
- Task View
- Console View
- Debugger

Demonstrate Use of Gild for Programming

- All of these steps are to be done by the researcher.
- Tell the participant that they can ask questions any time
- Talk aloud while doing all tasks
- In an empty workspace, add a new project, call it Test
- Add a file to that project, call it Test.java (emphasize need for .java)
- Type in the hello world program on the

- Save program
- Build program
- Run program

- Create and fix the following errors (and use save, build, run):
  - change Test to test
  - change Emergency to emergency
  - remove the first quotation mark in the string
  - remove the last quotation mark in the string

Instruct Participant to Use of Gild for Programming

Encourage the participant to talk aloud and ask any questions or request help if they become confused during this next stage.

Instruct the Participant repeat the steps just performed but use name Test2 for the project name.

Guide the user to create and fix the following different bugs (**used for Participant 1 only since they did not know Java**):

- change main to man
- change println to bogus
- remove the semicolon at the end of the println statement
- remove the last brace in the program

Demonstrate Use of Gild's Debug View

- Import the archived project Lemonade.zip
- Explain purpose of the program
- Explain that debuggers are useful for stepping through hard to understand code
- Quickly read through the code
- Set a break point at the beginning
- Run the Debugger, step through the program commenting on what's happening.

Direct the Participant to use Debugger

- Change the conditions of today to different values
- Ask the participant to predict the result of the program
- Direct the participant to use the debugger to step through the program and explain the new results
- Ask the participant to look away from the screen and enter a logical error into the code (insert a space in one of the string literals located in an if statement that should fire true)
- Tell the participant that you've introduced a logical error and to use the debugger to find it.

(For advance users) Direct the Participant to use solve a logical problem

- Set up the PowersOfTwo problem – code is expected to calculate and output the powers of two, from 0 to 16.
- Code compiles and runs, but a logical error exists in both the way the output is produced on the screen, and the way the powers of two are calculated.
- Direct the participant to find and fix the problem.

Appendix B
Lemonade.java Code

```
/**
 *
 * This program is used to estimate the amount of business our lemonade stand
 * will have each day.  To use it, change the daily conditions to reflect the
 * current circumstances.
 *
 */
public class LemonadeStand {

        public static void main(String args[]) {

                //On an average day we get 15 customers;
                int numberOfCustomers = 15;

                //We charge $2 per drink
                int costPerDrink = 2;

                System.out.println("On an average day we make $" + numberOfCustomers *
costPerDrink);


                //These are todays conditions
                String weather = "sunny";
                String dayOfWeek = "Sunday";
                String moodOfStaff = "happy";


                if (weather.equals("sunny ")) {
                        //we get 5 extra customers on sunny days
                        numberOfCustomers = numberOfCustomers +5;
                        //we charege more on sunny days
                        costPerDrink = costPerDrink +1;
                }

                if (dayOfWeek.equals("Saturday")) {
                        //we get 10 extra customers on Saturday
                        numberOfCustomers = numberOfCustomers + 10;
                }

                if (dayOfWeek.equals("Sunday")) {
                        //we get 2 extra customers on Saturday
                        numberOfCustomers = numberOfCustomers + 2;
                        //we always charge $1 on Sundays
                        costPerDrink = 1;
                }

                if (dayOfWeek.equals("Monday")) {
                        //we get 5 less customers on Monday
                        numberOfCustomers = numberOfCustomers -5;
                }

                if (moodOfStaff.equals("happy")) {
                        //we get 10 more customers when our staff are happy
                        numberOfCustomers = numberOfCustomers + 10;
                }

                System.out.println("Today we  made $" + numberOfCustomers * costPerDrink);

        }
}
```

Appendix C
PowerOfTwo.java Code

```java
/**
 * A simple program that prints all of the powers of two from 0 up to
 * and including 32. After this is done, it prints 32 as a power of two.
 */
public class PowerOfTwo {
        /**
         * A field that represent a power of two.
         */
        protected static int powerOfTwo = 1;

        /**
         * The main method.
         */
        public static void main(String[] args) {

                //a counter that will give us the exponent of the power of two.
                int i = 0;

                //a loop to display the power of two.
                while (powerOfTwo < 32) {
                        //print which power of two we are currently at.
                        System.out.println("The current power of 2 is " + powerOfTwo);
                        //compute the next power of two.
                        powerOfTwo = getNextPower(powerOfTwo);
                        //update the exponent
                        i++;
                }
                System.out.println("2^" + i + " = " + powerOfTwo);
        }

        /**
         * Given that <code>currentPower</code> is a power of two,
         * <code>getNextPower(int)</code> returns the next power of two after
         * <code>currentPower</code>.
         * @param currentPower a power of two.
         * @return the next power of two after <code>currentPower</code>.
         */
        protected static int getNextPower(int currentPower) {
                return currentPower + 2;
        }

}
```