Lab 7 Project

In this lab we will study:

- Model-View-Controller Architecture
- Constructors vs. methods
- Parameter Passing (data types and objects)
- Debugging

Lab Project: Program X

1.Open BlueJ and create a new project, ProgramX
2.Create the following four classes as shown in Figure 1. Remember only the ProgramX class will be created with the main statement. It is the entry point for the project.

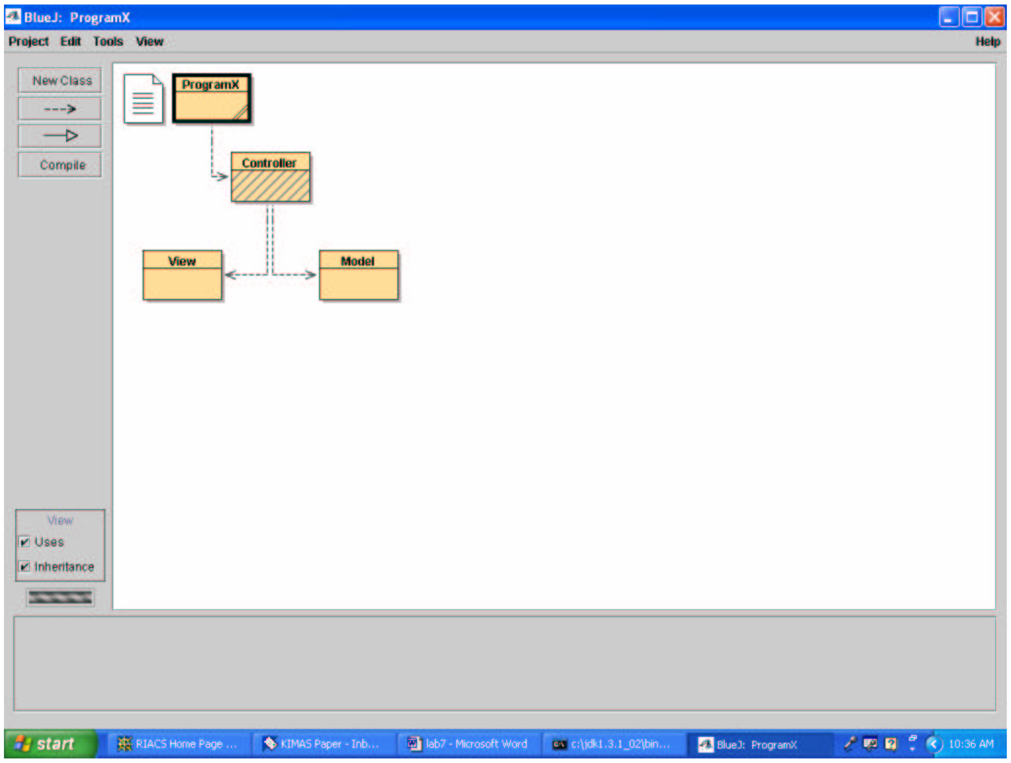


Figure 1

3.Open the **ProgramX** class and add the following code. *Make sure that this is the ONLY code in the class. (Take out all of the code supplied by BlueJ.)*

```
// program entry point
public class ProgramX
{   public static void main(String [] args)
 {     Controller c = new Controller();  // instantiate a new controller object "c"
        c.execute();  // use the "c" Controller object to call an execute method
                                   // from the Controller class
 }
}
```

4.Compile this class to insure that it is syntactically correct
5.Open the **View** class and add the following code

```
import javax.swing.*;
import java.text.*;

// The VIEW class - what you see and interact with!
public class View
{  JOptionPane jop = new JOptionPane(); //instantiate a new JOptionPane object "jop"
   DecimalFormat dec = new DecimalFormat("$###,##0.00");
   // instantiate a new DecimalFormat object
   // "dec" and pass it the desired format

// getInput method that asks the user to enter a number (actually a String until converted)
   public String getInput()
   { return jop.showInputDialog("Enter the Number!");
   }

   // showOutput method takes a double value "num", formats it and show it to the user within
   // a JOptionPane dialog
   public void showOutput(double num)
   { jop.showMessageDialog(null,"My number is: " + dec.format(num));
   }
}
```

6.Compile the View class to insure that it is syntactically correct

7.Open the **Model** class and add the following code

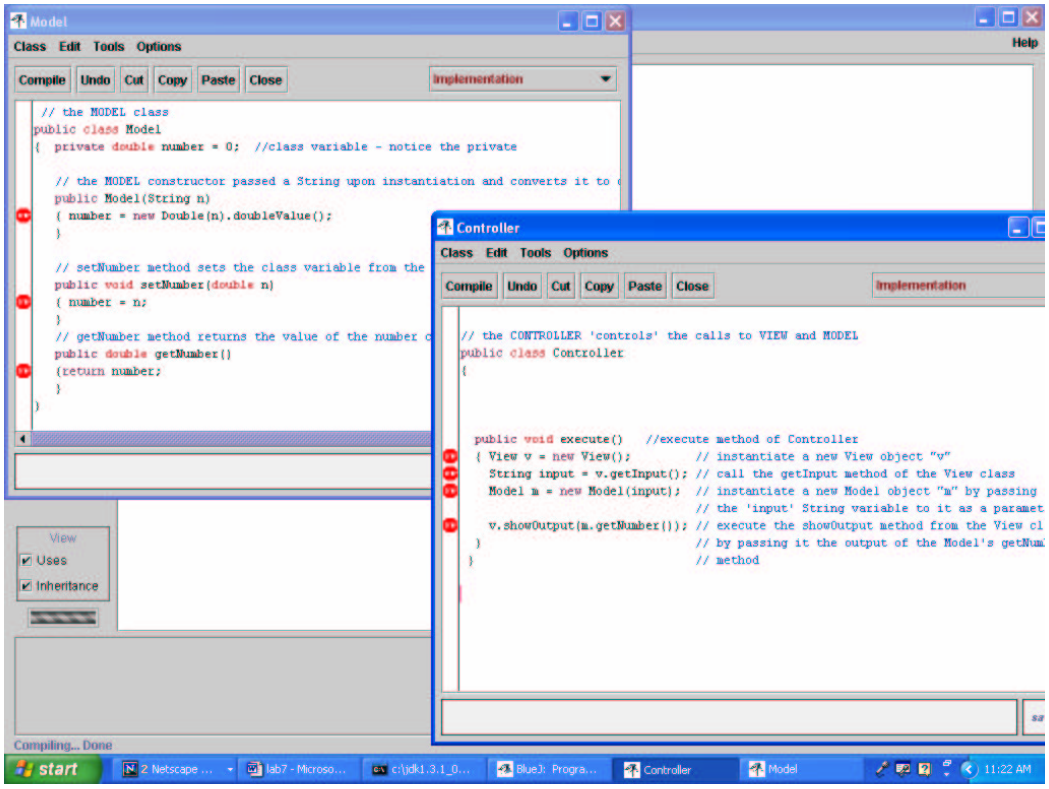
```
public class Model
{  private double number = 0;  //class variable - notice the private

   // the MODEL constructor passed a String upon instantiation and converts it to double
   public Model(String n)
   { number = new Double(n).doubleValue();
   }

   // setNumber method sets the class variable from the parameter "n"
   public void setNumber(double n)
   { number = n;
   }
   // getNumber method returns the value of the number class variable
   public double getNumber()
   {return number;
   }
}
```

8.Compile the Model class to insure that it is syntactically correct

9.Open the Controller class and add the following code

```
// the CONTROLLER 'controls' the calls to VIEW and MODEL
public class Controller
{    public void execute()   //execute method of Controller
     { View v = new View();        // instantiate a new View object "v"
       String input = v.getInput(); // call the getInput method of the View class
       Model m = new Model(input);  // instantiate a new Model object "m" by passing
                                    // the 'input' String variable to it as a parameter
       v.showOutput(m.getNumber()); // execute the showOutput method from the View class
     }     // by passing it the output of the Model's getNumber method
 }  //end of class
```

10. Compile the Controller class to insure that it is syntactically correct
11. Execute the project to view its execution
12. If the compile and execution work, then we will look at debugger
13. First go to the Help menu and open the tutorial file. When it is open read chapter 6 (pages 20-23) on debugging.
14. Set “break points” in your code by clicking on the left side of the screen (in the rectangle) as shown in Figure 2. When a breakpoint is set the execution will pause at that point in the program execution for you see the value of the relevant variables.



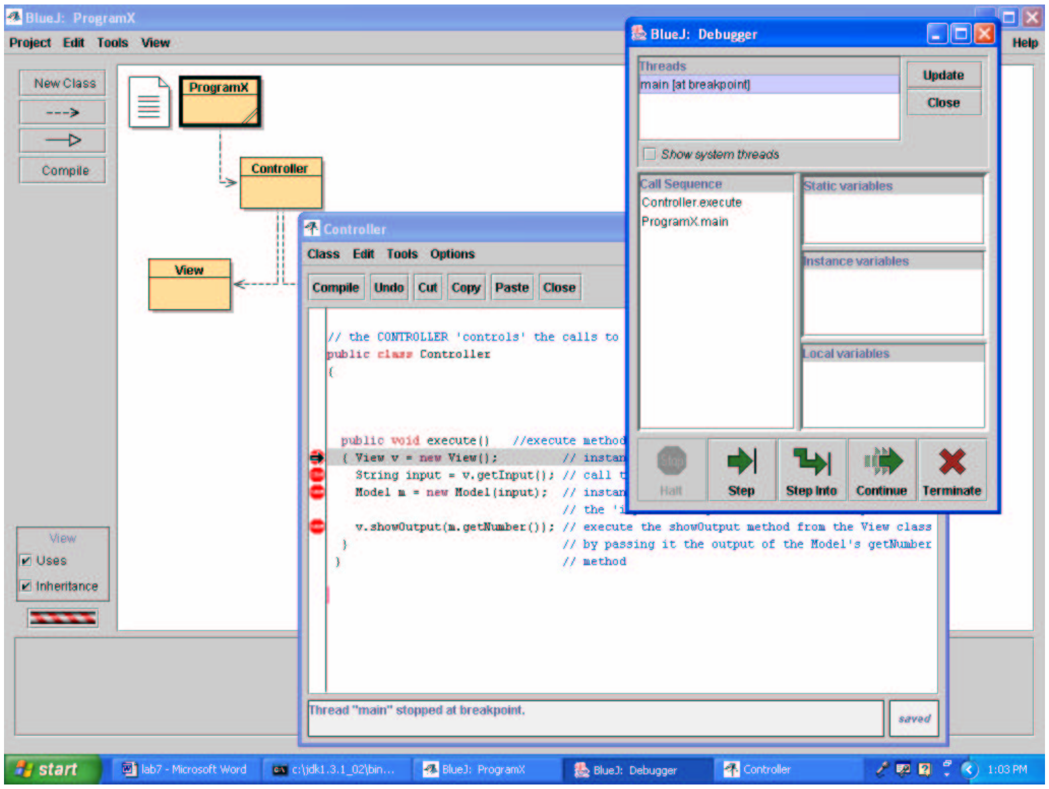
```
 // the MODEL class
public class Model
{  private double number = 0;  //class variable - notice the private

   // the MODEL constructor passed a String upon instantiation and converts it to
   public Model(String n)
   { number = new Double(n).doubleValue();
   }

   // setNumber method sets the class variable from the
   public void setNumber(double n)
   { number = n;
   }
   // getNumber method returns the value of the number c
   public double getNumber()
   {return number;
   }
}
```

```
// the CONTROLLER 'controls' the calls to VIEW and MODEL
public class Controller
{



  public void execute()   //execute method of Controller
  { View v = new View();          // instantiate a new View object "v"
    String input = v.getInput(); // call the getInput method of the View class
    Model m = new Model(input);  // instantiate a new Model object "m" by passing
                                 // the 'input' String variable to it as a paramet
    v.showOutput(m.getNumber()); // execute the showOutput method from the View cl
  }                              // by passing it the output of the Model's getNum
 }                               // method
```

Figure 2

15.Execute the program

16.You will see the program stop on your first breakpoint as shown in Figure 3.



```
// the CONTROLLER 'controls' the calls to
public class Controller
{


  public void execute()    //execute method
  { View v = new View();              // instan
     String input = v.getInput(); // call t
     Model m = new Model(input);  // instan
                                          // the 'i
     v.showOutput(m.getNumber()); // execute the showOutput method from the View class
  }                                       // by passing it the output of the Model's getNumber
 }                                        // method
```

Thread "main" stopped at breakpoint.

Figure 3

17.Step through your program several times to get a feel for how the execution flows and also how the debugger works. Use the STEP, STEP INTO, CONTINUE and TERMINATE buttons and discover what they do. Use different entry data to see what happens and how the data flows.

18.Change the setNumber method in the Model class to take the input double and return it taken to the 2nd power.

19.Once again recompile all classes and reset breakpoints. Notice what happened to the previous breakpoints when you re-compiled the code.

20.Execute several times again. Investigate, through the use of the debugger, the change that you just made.

21.To complete the lab, show the TA that you know how debugging works by taking them through a trace of the program and explaining what is happening.