```java
public class S3 extends CalcState
{
    public S3(CalcMachine machine, UserData userData, Controller controller, double answer){
        super(machine);

        Screen.displayAnswer(answer, userData);

        machine.setCalcState(new S4(machine, userData, controller));
    }
}
```

```
1  public abstract class screenState
2  {
3      protected  Machine machine;
4      protected int functType;
5      protected UserData userData;
6      // current state
7      public screenState(Machine machine, int functType, UserData userData){
8          this.machine=machine;
9          this.functType=functType;
10         this.userData=userData;
11     }
12
13 }
```

```java
public class S4 extends CalcState
{
    public S4(CalcMachine machine, UserData userData, Controller controller){
        super(machine);

        if( Screen.isExit() ) {
            System.exit(0);
        }
        else {
            machine.setCalcState(new S0(machine, userData, controller));
        }
    }
}
```

```java
public class initialState extends screenState
{
    public initialState(Machine machine, int functType, UserData userData, double answer){
        super(machine, functType, userData);

        if(functType == 1) //is a prompt
        {
            machine.setState(new promptState(machine, functType, userData));
        }
        else if(functType == 0)//is a display
        {
            machine.setState(new displayState(machine, functType, userData, answer));
        }
        else//is a display
        {
            machine.setState(new commandState(machine, functType, userData));
        }
    }



    }
```

```java

/**
 * Data Type Class
 * Kyle and Brayden
 */
public class UserData
{
    private double val1, val2;
    private boolean isPtr1, isPtr2;
    private char opChar;

    public UserData()
    {
        val1 = 0.0;
        val2 = 0.0;
        isPtr1 = false;
        isPtr2 = false;
        opChar = 'N';
    }

    public void setValues(double value1, double value2, boolean isPtr1, boolean isPtr2, char opChar) {
        val1 = value1;
        val2 = value2;
        this.isPtr1 = isPtr1;
        this.isPtr2 = isPtr2;
        this.opChar = opChar;
    }

    public double getVal1() { return val1; }
    public double getVal2() { return val2; }
    public boolean getIsPtr1() { return isPtr1; }
    public boolean getIsPtr2() { return isPtr2; }
    public char getOpChar() { return opChar; }
}

```

```java
import java.util.*;


/**
Kyle
 */
public class Screen
{
    public static void print(String string)
    {
        System.out.println(string + "\n");
    }

    public static void displayAnswer(double answer, UserData userData)
    {
        Machine machine=new Machine();
        machine.setState(new initialState(machine, 0, userData,
answer));
    }

    public static void prompt(UserData userData)
    {
        Machine machine=new Machine();
        machine.setState(new initialState(machine, 1, userData, 0));

    }
    public static boolean isExit()
    {
        System.out.println("Are you finished? (true or false)\n");
        Scanner keyboard = new Scanner(System.in);
        return keyboard.nextBoolean();
    }
    public static void commands(UserData userData)
    {
        Machine machine=new Machine();
        machine.setState(new commandState(machine, 3, userData));
    }
}

```

```java
import java.util.Vector;
import java.util.List;

/**
 * Brayden Burgess
 */
public class ALU
{
    private List<PathItem> operators = new Vector<PathItem>();
    private Instruction instr;

    public ALU()
    {
        instr = new Instruction();
        operators.add(new Adder());
        operators.add(new Subtractor());
        operators.add(new Multiplier());
        operators.add(new Divider());
    }

    public void setInstr(Instruction newInstr) { instr = newInstr; }

    public double getResult(Instruction instr) {
        int index;
        double result;
        switch(instr.getOperation()) {
            case OpCode.ADD:
                index = 0;
                break;
            case OpCode.SUB:
                index = 1;
                break;
            case OpCode.MUL:
                index = 2;
                break;
            case OpCode.DIV:
                index = 3;
                break;
            default: throw new RuntimeException("Invalid Operation - Please try again!");
        }

        operators.get(index).setVals(instr.getValue1(), instr.getValue2());
        return operators.get(index).getOutput();
    }
}

```

```java

/**
 * Brayden Burgess
 */
public class Instruction //Brayden Burgess
{
    private OpCode opCode;
    private double value1;
    private double value2;

    public Instruction()
    {
        value1 = 1;
        value2 = 1;
        opCode = OpCode.NOP;
    }

    public Instruction(double val1, double val2, char op)
    {
        value1 = val1;
        value2 = val2;
        this.setOperation(op);
    }

    public double getValue1() { return value1; }
    public double getValue2() { return value2; }
    public OpCode getOperation() {return opCode; }

    public void setValue1(double newVal1) { value1 = newVal1; }
    public void setValue2(double newVal2) { value2 = newVal2; }

    public void setOperation(char newOp) {
        switch(newOp) {
            case 'A': opCode = OpCode.ADD; break;
            case 'S': opCode = OpCode.SUB; break;
            case 'M': opCode = OpCode.MUL; break;
            case 'D': opCode = OpCode.DIV; break;
            default:  opCode = OpCode.NOP; break;
        }
    }
}

```

```java
import java.util.*;


public class Machine
{
    public screenState state;


    public Machine() {
        state=null;
    }

    public void setState(screenState state){
        this.state=state;

    }
    public void terminateState(){
        state=null;
    }
}
```

```java
public abstract class CalcState
{
    protected CalcMachine machine;

    // current state
    public CalcState(CalcMachine machine){
        this.machine=machine;
    }
}
```

```java
import java.util.Scanner;

public class promptState extends screenState //do prompt state
{
    public promptState(Machine machine, int functType, UserData userData){
        super(machine, functType, userData);


        char opChar;

        double val1;
        double val2;

        int boolTemp;
        boolean isPtr1;
        boolean isPtr2;

        //function picker
        System.out.println("Choose a function (A,S,M,D): ");
        Scanner keyboard = new Scanner(System.in);
        opChar = keyboard.next().charAt(0);

        //is value 1 from memory?
        System.out.println("Is the first value from memory? (true for yes, false for no)");
        isPtr1 = keyboard.nextBoolean();

        //is value 2 from memory?
        System.out.println("Is the second value from memory? (true for yes, false for no)\n");
        isPtr2 = keyboard.nextBoolean();


        //get first value
        System.out.println("Choose the first value: ");
        val1 = keyboard.nextDouble();

        //get second value
        System.out.println("Choose the second value: "); //get second value
        val2 = keyboard.nextDouble();

        machine.setState(new valuesState(machine, functType, userData, val1, val2, isPtr1, isPtr2, opChar));
        }
    }
```

```java
/**
 Kyle
 */
public class PathItem
{
    double val1;
    double val2;


    protected PathItem(double val1, double val2)
    {
        this.val1 = val1;
        this.val2 = val2;
    }

    protected PathItem()
    {
        this.val1 = 0.0;
        this.val2 = 0.0;
    }

    public void setVals(double val1, double val2)
    {
        this.val1 = val1;
        this.val2 = val2;
    }


    protected double getOutput(){return 0.0;} //parent operation
}

```

```java

/**
 * Brayden Burgess
 */

public class Calculator
{
    public static void main() {
        Controller controller = new Controller();
        double answer;
        boolean exitTime = false;
        UserData userData = new UserData();

        CalcMachine calcStateMachine = new CalcMachine();

        String welcome = ("Welcome to the Calculator\nEnter a command (? for list): ");

        Screen.print(welcome);




        while(true) {
            try {
                calcStateMachine.setCalcState(new S0(calcStateMachine, userData, controller));

                //Below obsolete (done by above state machine):

                //main calculator code
                //Screen.prompt(userData);

                //to calculate:
                //controller.setValues(userData.getVal1(), userData.getVal2(), userData.getIsPtr1(),
userData.getIsPtr2());
                //answer = controller.doOperation(userData.getOpChar());
                //end calculation

                //Screen.displayAnswer(answer, userData);

                //ask if the user would like to quit
                //exitTime = Screen.isExit();
            }
            catch(Exception e) {
                Screen.print(e.getMessage()); //RuntimeException thrown for invalid operation
            }
        }
    }
}
```

```

/**
 * Brayden Burgess
 */
public enum OpCode
{
    ADD, SUB, MUL, DIV, NOP
}

```

```
1
2  /**
3   * Brayden Burgess
4   */
5  public class Multiplier extends PathItem
6  {
7      public double getOutput() { return val1 * val2; }
8  }
9
```

```java
public class CalcMachine
{
    public CalcState state;

    public CalcMachine() {
        state=null;
    }

    public void setCalcState(CalcState state){
        this.state=state;
    }
}
```

```java
/**
 Kyle
 */
public class Adder extends PathItem //subclass of PathItem built for addition
{
    public double getOutput(){return val1 + val2;} //return the addition
}

```

```java
import java.util.List;
import java.util.Vector;

/**
 * Brayden Burgess
 */
public class Controller
{
    private double value1, value2;
    private boolean isVal1MemLoc, isVal2MemLoc;
    private char operation; //ASMD
    private ALU alu = new ALU();
    private Instruction instr;
    private List<Register> regFile = new Vector<Register>();

    public Controller()
    {
        isVal1MemLoc = false;
        isVal2MemLoc = false;
        operation = 'N';
        instr = new Instruction(0.0, 0.0, operation);
    }

    public double doOperation(char op) {
        operation = op;
        instr = new Instruction(value1, value2, operation);

        double ans = alu.getResult(instr); //call alu's function

        Register tempReg = new Register(ans);
        //tempReg.setVal(ans);
        regFile.add(tempReg);
        return ans;
    }

    public void setValues(double v1, double v2, boolean isPtr1, boolean isPtr2) {
        isVal1MemLoc = isPtr1;
        isVal2MemLoc = isPtr2;

        int index1 = (int)v1;
        int index2 = (int)v2;

        value1 = (isVal1MemLoc) ? regFile.get(index1).getVal() : v1;
        value2 = (isVal2MemLoc) ? regFile.get(index2).getVal() : v2;
    }

}
```

```java
public class valuesState extends screenState
{
    public valuesState(Machine machine, int functType, UserData userData, double val1, double val2,
    boolean isPtr1, boolean isPtr2, char opChar){
        super(machine, functType, userData);


        userData.setValues(val1, val2, isPtr1, isPtr2, opChar);

        machine.terminateState();
    }



    }
```

```java
public class displayState extends screenState //do display
{
    public displayState(Machine machine, int functType, UserData userData, double answer){
        super(machine, functType, userData);

        char op = userData.getOpChar();
        double val1 = userData.getVal1();
        double val2 = userData.getVal2();


        switch(op)
        {
            case 'A':
            System.out.println("Sum: " + String.format("%.4f", answer)
                + "\n");
                break;

            case 'S':
            System.out.println("Difference: " + String.format("%.4f", answer)
                + "\n");
                break;

            case 'M':
                System.out.println("Product: " + String.format("%.4f", answer)
                + "\n");
                break;

            case 'D':
                System.out.println("Quotient: " + String.format("%.4f", answer)
                + "\n");
                break;

        }

        machine.terminateState();
    }
    }
```

```java
/**
 Kyle
 */
public class Register
{
    private double memVal; //value stored in the register

    public Register(double memVal) {this.memVal = memVal;} //constructor that instantiates a value

    public Register(){this.memVal = 0.0;} //default constructor

    public void setVal(double newVal) {this.memVal = newVal;}//setter to change register's value

    public double getVal(){return memVal;} //getter to return register's value


}

```

```java
import java.util.Scanner;

public class commandState extends screenState
{
    public commandState(Machine machine, int functType, UserData userData){
        super(machine, functType, userData);

        //variables needed for setup
        boolean commandExit = false;
        String userCommand;
        Scanner keyboard = new Scanner(System.in);

        while (!commandExit){
        userCommand = keyboard.nextLine();

        switch(userCommand)
            {
            case "?":
                String commandList = "---------------\n" +
                                     "List of Commands: \n" +
                                     "MathTime (Enters the calculator)\n" +
                                     "Credits (Display credits)\n" +
                                     "Exit (Bye bye)\n" +
                                     "---------------\n";

                Screen.print(commandList);
                break;
            case "MathTime":
                Screen.print("Good Luck Soldier\n");

                commandExit = true;
                break;

            case "Credits":
                String credits = "-----------------------------\n" +
                                 "Master programmers: Brayden and Kyle\n" +
                                 "Made his diagrams pretty: Brayden\n" +
                                 "Made the user interface pretty: Kyle\n" +
                                 "Wasted most time: Kyle\n" +
                                 "Needs more sleep: Brayden\n" +
                                 "-----------------------------\n";
                Screen.print(credits);
                break;

            case "Exit":
                System.exit(0);
                break;

            default:
                Screen.print("Not a command\n");
                break;

            }
        }

        machine.terminateState();
        }
    }
```

```

/**
 * Brayden Burgess
 */
public class Divider extends PathItem
{
    public double getOutput() {
        return (double)val1 / (double)val2;
    }
}

```

```java
public class S0 extends CalcState
{
    public S0(CalcMachine machine, UserData userData, Controller controller){
        super(machine);

        boolean initialize = true;
        if (initialize)
        {
            Screen.commands(userData);
            initialize = false;
        }


        Screen.prompt(userData);

        machine.setCalcState(new S1(machine, userData, controller));
    }
}
```

```java

/**
Kyle
 */
public class Subtractor extends PathItem //subclass of pathItem built for subtraction
{
   public double getOutput(){return val1 - val2;} //do the subtraction
}

```

```java
public class S1 extends CalcState
{
    public S1(CalcMachine machine, UserData userData, Controller controller){
        super(machine);

        controller.setValues(userData.getVal1(), userData.getVal2(), userData.getIsPtr1(), userData.getIsPtr2());

        machine.setCalcState(new S2(machine, userData, controller));
    }
}
```

```java
public class S2 extends CalcState
{
    public S2(CalcMachine machine, UserData userData, Controller controller){
        super(machine);

        double answer = controller.doOperation(userData.getOpChar());

        machine.setCalcState(new S3(machine, userData, controller, answer));
    }
}
```