```java
public class Computer {

    /** This constant represents the size of the memory unit of this Computer
     *   (number of memory registers). */
    public final static int MEM_SIZE = 100;

    /** This constant represents the memory address at which the constant 0 is
stored. */
    public final static int LOCATION_OF_ZERO = MEM_SIZE - 2;

    /** This constant represents the memory address at which the number 1 is
stored. */
    public final static int LOCATION_OF_ONE = MEM_SIZE - 1;

    // Op-code definitions:
    private final static int WRITE = 9;
    private final static int READ = 8;
    private final static int GOTOP = 7;
    private final static int GOTOZ = 6;
    private final static int GOTO = 5;
    private final static int STORE = 4;
    private final static int LOAD = 3;
    private final static int SUB = 2;
    private final static int ADD = 1;
    private final static int STOP = 0;

    // Put the rest of the op-code definitions here.

    /** The Computer consists of a Memory unit, and two registers, as follows: */
    private Memory m;
    private Register dReg;
    private Register pc;

    public Computer() {
        m = new Memory(MEM_SIZE);
        dReg = new Register();
        pc = new Register();
        reset();
    }

    public void reset() {
        m.reset();
        m.setValue(LOCATION_OF_ZERO, 0);
        m.setValue(LOCATION_OF_ONE, 1);
        pc.setValue(0);
        dReg.setValue(0);
    }

    public void run() {
        int stop = 0;
        while (stop != -1) {
            int value = m.getValue(pc.getValue());

            if (value / 100 == ADD) {
                execAdd(value - 100);
            } else if (value / 100 == SUB) {
                execSub(value - 200);
            } else if (value / 100 == LOAD) {
                execLoad(value - 300);
            } else if (value / 100 == STORE) {
                execStore(value - 400);
```

```java
 60                } else if(value / 100 == GOTO) {
 61                    execGoto(value - 500);
 62                } else if(value / 100 == GOTOZ) {
 63                    execGotoz(value - 600);
 64                } else if(value / 100 == GOTOP) {
 65                    execGotop(value - 700);
 66                } else if(value / 100 == READ) {
 67                    execRead();
 68                } else if(value / 100 == WRITE) {
 69                    execWrite(dReg.getValue());
 70                } else {
 71                    stop = execStop();
 72                }
 73            }
 74        }
 75
 76        // Private execution routines, one for each Vic command
 77        private void execLoad (int addr) {
 78            dReg.setValue(m.getValue(addr));
 79            pc.addOne();
 80        }
 81
 82        private void execRead () {
 83            int value = StdIn.readInt();
 84            dReg.setValue(value);
 85            pc.addOne();
 86        }
 87
 88        private void execWrite (int value) {
 89            System.out.println("" + value);
 90            pc.addOne();
 91        }
 92
 93        private void execStore (int addr) {
 94            m.setValue(addr, dReg.getValue());
 95            pc.addOne();
 96        }
 97
 98        private void execAdd (int addr) {
 99            dReg.setValue(m.getValue(addr) + dReg.getValue());
100            pc.addOne();
101        }
102
103
104        private void execGoto (int addr) {
105            pc.setValue(addr);
106        }
107
108
109        private void execGotop (int addr) {
110            if (dReg.getValue() > 0) {
111                execGoto(addr);
112            } else {
113                pc.addOne();
114            }
115        }
116
117
118        private void execGotoz (int addr) {
119            if (dReg.getValue() == 0) {
120                execGoto(addr);
```

```java
121            } else {
122                pc.addOne();
123            }
124        }
125
126        private void execSub (int addr) {
127            dReg.setValue(dReg.getValue() - m.getValue(addr));
128            pc.addOne();
129        }
130
131        private int execStop () {
132            System.out.println("Program terminated normally");
133            pc.addOne();
134            return -1;
135        }
136
137        public void loadProgram(String fileName) {
138            StdIn.setInput(fileName);
139            int counter = 0;
140            while(StdIn.hasNextLine()) {
141                int value = StdIn.readInt();
142                m.setValue(counter, value);
143                counter++;
144            }
145        }
146
147        public void loadInput(String fileName) {
148            StdIn.setInput(fileName);
149        }
150
151        public String toString () {
152            // Put your code here
153            return "D register  = " + dReg.getValue() + "\n" + "PC register = " +
     pc.getValue() + "\n" + "Memory state:" + "\n" + m.toString();
154        }
155 }
```