Take Home Exam 2

This exam will use the State Pattern to implement a simple interpreted assembler.

Our assembler initially has 5 instructions, given below. A state diagram for these instructions is shown in stateDiagram.pdf.

addi *reg*, *int* // adds *int* to the contents of *reg*
add reg_1, reg_2, reg_3 // $reg_3 = reg_2 + reg_3$
ldz *reg* // *reg* = 0
print reg_i // System.out.println(registers[i]);
exit // print "terminating normally" and then exit the program

reg is a value 0, 1, 2 or 3 and specifies a register. Registers can be represented as an array of integers. *int* is an integer constant, e.g., 2357.

Given the program

```
ld0 1
add 1 ,  1 , 1
addi 1, 5
print 1
add 1, 1, 1
add 1,  1,  1
print 1
exit
```

the output is

```
reg[1] = 0

reg[1] = reg[1] + reg[1]

reg[1] = reg[1] + 5

print reg[1]

reg[1] = reg[1] + reg[1]

reg[1] = reg[1] + reg[1]

print reg[1]
```

Normal termination of compiler

I will not be worried about line spacing or white space in your answers.

PartA:

Given the code in PartA, write the classes AddiState, CVState, ExitState, Ld0State, PrintState, VState. It should run correctly on testpgm.txt and testpgm2.txt. testpgm2.tex contains erroneous statements.

PartB:

Given the program you wrote in PartA, add a new instruction

store reg1, reg2, reg3, reg4

where the statement

store 0, 1, 2, 3

would produce the output

store reg[0], reg[1], reg[2], reg[3]

The file testpgm3.txt can be used as a test program.

What to turn in:

Make a directory called your username. It should have PartA and PartB subdirectories. These in turn should contain all of the code needed to run the program, along with testpgm.txt and testpgm2.txt in PartA, and testpgm3.txt in PartB. Include a README file that says how to build and run your program on a file.