```
/**
 * StackMachine.java
 * A simple stack-based arithmetic processor.
 *
 * @author Hawk Weisman
 * @see StackMachineInstruction
 * @see Stack
 *
 * PLEDGE:
 */
import java.util.EmptyStackException;

public class StackMachine {

    public static void main (String[] argv) {

        Stack<StackMachineInstruction> ExecutionStack;
        ConsoleReader console = new ConsoleReader(System.in);
        boolean keepLooping = true;

        // determine what kind of stack we are working with
        if (argv.length > 0) {
            switch (argv[0]) {
                case "-node":
                    ExecutionStack = new NodeStack<StackMachineInstruction>();
                    break;
                case "-array":
                    // FIXME: write ArrayStack
                    ExecutionStack = new ArrayStack<StackMachineInstruction>();
                    break;
                case "-help":
                    System.out.println ("> Welcome to StackMachine." +
                                        "\n> Command-line Arguments:" +
                                        "\n> -node: runs with NodeStack " +
                                        "\n> -array: runs with ArrayStack \n> -
                                            help: displays this help file \n>
                                            Syntax: " +
                                        "\n> +, -, *, /, %: puts an add,
                                            subtract, multiply, divide, or modulo
                                            command on the stack" +
                                        "\n> d: displays the contents of the
                                            stack " +
                                        "\n> s: puts a swap on the stack" +
                                        "\n> e: evaluates the top instruction" +
                                        "\n> x: exits StackMachine");
                    ExecutionStack = new NodeStack<StackMachineInstruction>();
                    break;
                default:
                    ExecutionStack = new NodeStack<StackMachineInstruction>();
                    break;
                }
        } else {
            ExecutionStack = new NodeStack<StackMachineInstruction>();
            System.out.println ("> Welcome to StackMachine." +
                                "\n> Command-line Arguments:" +
                                "\n> -node: runs with NodeStack " +
```

```
                        "\n> -array: runs with ArrayStack \n> -help:
                            displays this help file \n> Syntax: " +
                        "\n> +, -, *, /, %: puts an add, subtract,
                            multiply, divide, or modulo command on the
                            stack" +
                        "\n> d: displays the contents of the stack " +
                        "\n> s: puts a swap on the stack" +
                        "\n> e: evaluates the top instruction" +
                        "\n> x: exits StackMachine");
        }
        try {
            do {
                System.out.print ("> ");

                String currentLine = console.readLine();
                StackMachineInstruction currentInstruction;
                // parse input into StackMachineInstructions
                try {
                    switch (currentLine) {
                        case "+":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.ADD);
                            ExecutionStack.push(currentInstruction);
                            break;
                        case "-":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.SUBTRACT
                                );
                            ExecutionStack.push(currentInstruction);
                            break;
                        case "/":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.DIVIDE);
                            ExecutionStack.push(currentInstruction);
                            break;
                        case "*":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.MULTIPLY
                                );
                            ExecutionStack.push(currentInstruction);
                            break;
                        case "%":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.MODULO);
                            ExecutionStack.push(currentInstruction);

                            break;
                        case "s":
                            currentInstruction = new StackMachineInstruction
                                (StackMachineInstruction.InstructionType.SWAP);
                            ExecutionStack.push(currentInstruction);

                            break;
                        case "e":
                            currentInstruction = ExecutionStack.pop();
                            currentInstruction.eval(ExecutionStack);
```

```
                            break;
                        case "d":
                            System.out.println (ExecutionStack.toString());
                            break;
                        case "x":
                            keepLooping = false;
                            break;
                        default:
                            try {
                                currentInstruction = new StackMachineInstruction
                                    (Integer.parseInt(currentLine));
                                ExecutionStack.push(currentInstruction);
                                } catch (NumberFormatException e) {
                                    System.err.println ("> Please only enter
                                        numbers and acceptable instructions.");
                                }
                            break;
                        }

                    } catch (EmptyStackException e) {
                        System.err.println ("> Cannot pop, the stack is empty.");

                    } catch (FullStackException e) {
                        System.err.println ("> Cannot push, the stack is full.");

                    }
                } while (keepLooping);

            } catch ( Exception e) {
                e.printStackTrace(System.err);
            }
        }
    }
```