CS 2, Fall 2015 **CS 2 Cover Letter**
Dr. Gurka

Name: Sergiy Kolodyazhnyy
Assignment: Project 4, big addition, final submission
Date submitted: Sept 29, 2015
Total time: 30 – 35 hours

On time or late? On time
GOOJF?

Did you collaborate with any classmates on this project? No
If yes, who and what did you work together on?

Did you get any tutoring or similar help on this project? Explain.

<u>How'd it go?</u>

What went well?

Preliminary program went pretty smooth hence the final submission also turned out quite easy

What problems did you have?

None on the final submission

What did you learn new?

Use of stacks, interaction between objects, scope of variables from class to class

Any remaining questions?

No

Other comments on the project?

If this project is late, what is not included or not working correctly?

Other discussions as specified with the project.

Driver class

```
 1:/*****************************************
 2:Author: Sergiy Kolodyazhnyy
 3:Course: CS 2050
 4:Date: Sept 29 2015
 5:Instructor: Prof Gurka
 6:Java version: OpenJDK, 1.7.0
 7:IDE: nano text-editor and java compiler
 8:Project: #4, Big Addition, final
 9:*****************************************/
10:
11:import java.io.File;
12:import java.util.Scanner;
13:import java.io.IOException;
14:
15:public class driver
16:{
17:    public static void main(String[] args) throws IOException
18:    {
19:        String planAuthor;
20:        File inpFile = new File(args[0]);
21:        Scanner readData = new Scanner(inpFile);
22:        data inputData = new data();
23:        planAuthor = readData.nextLine();
24:
25:        printHeader(planAuthor);
26:
27:        while (readData.hasNext())
28:            {
29:
30:                inputData.caseDescription = readData.nextLine();
31:                inputData.opA = readData.nextLine();
32:                inputData.opB = readData.nextLine();
33:                inputData.result = readData.nextLine();
34:
35:                System.out.println("Case:" + inputData.caseDescription);
36:                System.out.println("Operand 1: " + formatNumber(inputData.opA) );
37:                System.out.println("Operand 2: " + formatNumber(inputData.opB) );
38:                System.out.println("Expected Result: " + formatNumber(inputData.result) );
39:                System.out.print("Result from stack arithmetic: ");
40:                System.out.print(formatNumber( inputData.performAddition() ) );
41:                System.out.printf("\n\n");
42:
43:            }
44:
45:
46:    }
47:
48:    public static void printHeader(String testAuthor)
49:    {
50:        System.out.println("Author: Sergiy Kolodyazhnyy\tCS-2050,Fall-2015");
51:        System.out.println("Project #4 - Big Addition\n\n");
```

```
52:       System.out.println("Plan author:" + testAuthor);
53:    }
54:
55:    public static String formatNumber(String num)
56:    {
57:       String temp = "";
58:       String formated = "";
59:       int digitCount = 0;
60:
61:       for (int i = num.length() - 1 ; i >=0 ; i-- )
62:          {
63:
64:             digitCount++;
65:             if (digitCount%3==0)
66:                {
67:                   temp = temp + num.charAt(i) + "," ;
68:                }
69:             else
70:                temp = temp + num.charAt(i) ;
71:
72:          }
73:
74:       for (int j = temp.length()-1; j >= 0; j-- )
75:          {
76:             if (j == temp.length() - 1 && temp.charAt(j) == ',')
77:                continue;
78:             formated = formated + temp.charAt(j);
79:          }
80:
81:       return formated;
82:    }
83:
84:
85:}
```

Data class (performs calculations)

```
 1:public class data
 2:{
 3:
 4:    //String planAuthor;
 5:    String caseDescription;
 6:    String opA;
 7:    String opB;
 8:    String result;
 9:    int sumStkSize;
10:
11:    public String performAddition()
12:    {
13:       String output = "" ;
14:       stack numA = new stack(opA.length());
15:       stack numB = new stack(opB.length());
16:
17:       if ( numA.size > numB.size )
18:          {
```

```
19:             sumStkSize =  numA.size + 1;
20:          }
21:       else
22:          {
23:             sumStkSize = numB.size + 1;
24:          }
25:
26:       stack sumStk = new stack(sumStkSize);
27:       int result;
28:
29:       // break strings into individual numbers, push onto stacks
30:       for (int i=0; i<opA.length(); i++)
31:          numA.push(Character.getNumericValue(opA.charAt(i)));
32:
33:       for (int i=0; i<opB.length(); i++)
34:          numB.push(Character.getNumericValue(opB.charAt(i)));
35:
36:       int carry = 0;
37:       while( !(numA.isEmpty()) || !(numB.isEmpty())  )
38:          {
39:             int num1 = numA.pop();
40:             int num2 = numB.pop();
41:
42:             if ( num1 == -1 )
43:                {
44:                   sumStk.push(num2 + carry);
45:                   carry=0;// this ensures  carry is used only once
46:                   // if there is one
47:                   continue;
48:                }
49:             else if (num2 == -1)
50:                {
51:                   sumStk.push(num1 + carry);
52:                   carry=0; // same as above
53:                   continue;
54:                }
55:
56:
57:             int sumNums = num1 + num2 + carry;
58:             // This part considers the carry number in  normal addition
59:             if (sumNums >= 10 )
60:                {
61:                   sumNums = sumNums - 10;
62:                   carry = 1;
63:                   // the if statement bellow considers special case where both
64:                   // numbers have carry resulting form adding highest
65:                   // digits, as in case of 512 + 512 = 1024
66:                   if (numA.isEmpty() && numB.isEmpty())
67:                      {
68:                         sumStk.push(sumNums);
69:                         sumStk.push(carry);
70:                         break;
71:                      }
72:                }
```

```
73:                 else
74:                     {
75:                         carry = 0;
76:                     }
77:                 sumStk.push(sumNums);
78:             }
79:
80:         // now that we have answer stack ready
81:         // we build the string out of the digits
82:         // on the stack for later processing in
83:         // the driver class
84:         for(int i = 0; i <sumStk.size; i++ )
85:             {
86:                 int poppedNum = sumStk.pop();
87:                 if ( poppedNum != -1 )
88:                     output+=poppedNum;
89:             }
90:
91:
92:         return output;
93:     }// end of performAddition()
94:
95:} //end of class
```

Stack class

```
 1:public class stack
 2:{
 3:    private int top = -1;
 4:    int size;
 5:    int[] stack ;
 6:
 7:    // default constructor with size 10
 8:    public stack()
 9:    {
10:        size=10;
11:        stack = new int[size];
12:    }
13:    // constructor with custom size
14:    public stack(int arraySize)
15:    {
16:        size=arraySize;
17:        stack= new int[size];
18:    }
19:
20:    public void push(int value)
21:    {
22:        if(!(top==size-1))
23:            {
24:                top=top+1;
25:                stack[top]=value;
26:            }
27:        else
28:            {
29:                System.err.println("Stack is full, can't push a value");
```

```
30:             System.exit(-1);
31:
32:         }
33:   }
34:
35:   public int pop()
36:   {
37:      if(!isEmpty())
38:         {
39:
40:            int num = stack[top];
41:            top--;
42:            return num;
43:
44:         }
45:      else
46:         {
47:            return -1;
48:         }
49:   }
50:
51:   public boolean isEmpty()
52:   {
53:      return (top==-1);
54:   }
55:
56:   public void display()
57:   {
58:
59:      for(int i=0; i<=top; i++)
60:         {
61:            System.out.print(stack[i]+ " ");
62:         }
63:      System.out.println();
64:   }
65:}
```

Command line output

Script started on Tue 29 Sep 2015 07:24:35 PM MDT

hw4:$ javac driver.java

hw4:$ java driver prof-gurka-test-data.txt
Author: Sergiy Kolodyazhnyy CS-2050,Fall-2015
Project #4 - Big Addition

Plan author:Programmer: ??, Test plan author: J. Gurka
Case:minimum operands
Operand 1: 0
Operand 2: 0
Expected Result: 0
Result from stack arithmetic: 0

Case:minimum carry
Operand 1: 1
Operand 2: 9
Expected Result: 10
Result from stack arithmetic: 10

Case:minimum operand size, no carry
Operand 1: 2
Operand 2: 7
Expected Result: 9
Result from stack arithmetic: 9

Case:carry across all digits
Operand 1: 999,999
Operand 2: 1
Expected Result: 1,000,000
Result from stack arithmetic: 9,999,100

Case:first operand zero, different lengths
Operand 1: 0
Operand 2: 12,345
Expected Result: 12,345
Result from stack arithmetic: 12,345

Case:second operand zero, different lengths
Operand 1: 6,789
Operand 2: 0
Expected Result: 6,789
Result from stack arithmetic: 6,789

Case:no commas, no carry, same lengths
Operand 1: 333
Operand 2: 444
Expected Result: 777
Result from stack arithmetic: 777

Case:no commas in operands, comma in answer, same lengths
Operand 1: 606
Operand 2: 404
Expected Result: 1,010
Result from stack arithmetic: 1,010

Case:answer exactly maximum integer
Operand 1: 1,073,741,824
Operand 2: 1,073,741,823
Expected Result: 2,147,483,647
Result from stack arithmetic: 2,147,483,647

Case:operands > maximum integer, no carry, same lengths
Operand 1: 11,111,111,111,111
Operand 2: 22,222,222,222,222
Expected Result: 33,333,333,333,333
Result from stack arithmetic: 33,333,333,333,333

Case:operands > maximum integer, some carries, different lengths
Operand 1: 123,456,123,456,123,456
Operand 2: 5,678,567,856,785,678
Expected Result: 129,134,691,312,909,134
Result from stack arithmetic: 129,134,691,312,909,134

Case:both operands = maximum integer
Operand 1: 2,147,483,647
Operand 2: 2,147,483,647
Expected Result: 4,294,967,294
Result from stack arithmetic: 4,294,967,294

Case:maximum stack size, both operands and answer
Operand 1: 1,222,333,444,555,666,777,888,999
Operand 2: 3,210,003,210,003,210,003,210,000
Expected Result: 4,432,336,654,558,876,781,098,999
Result from stack arithmetic: 4,432,336,654,558,876,781,098,999

hw4:$ exit
Script done on Tue 29 Sep 2015 07:25:01 PM MDT

Original input file

hw4:$ cat prof-gurka-test-data.txt
Programmer: ??, Test plan author: J. Gurka
minimum operands
0
0
0
minimum carry
1
9
10
minimum operand size, no carry
2
7
9
carry across all digits
999999
1
1000000
first operand zero, different lengths
0
12345
12345
second operand zero, different lengths
6789
0
6789
no commas, no carry, same lengths
333
444

777
no commas in operands, comma in answer, same lengths
606
404
1010
answer exactly maximum integer
1073741824
1073741823
2147483647
operands > maximum integer, no carry, same lengths
11111111111111
22222222222222
33333333333333
operands > maximum integer, some carries, different lengths
123456123456123456
5678567856785678
129134691312909134
both operands = maximum integer
2147483647
2147483647
4294967294
maximum stack size, both operands and answer
1222333444555666777888999
3210003210003210003210000
4432336654558876781098999