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

Name: Sergiy Kolodyazhnyy
Assignment: Project #4 Big Addition
Date submitted: 9 / 21 / 2015
Total time: about 8 hours

On time or late? On time
GOOJF? no

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

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

Mostly referenced to java API documentation online.

How'd it go?

What went well?

Desing idea for the stack, data classes, getting to store necessary data from file to data objects. With the test data from input files, it appears my program does addition properly, as outputs and expected outputs match.

What problems did you have?

The originall idea for how stack should perform addition didn't work out quite as I planned,but was quickly corrected with series of step-by-step edits and recompilings.

What did you learn new?

Creating array of objects, creating stacks.

Any remaining questions?

Other comments on the project?

This being preliminary submission, I have focused on functionality. Still missing comma formating.

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

Other discussions as specified with the project.

Driver class

This reads data from an input file (which is passed as one of the command line arguments), stores the information into array of objects created from data class.

```
 1: /**************************************
 2: Author: Sergiy Kolodyazhnyy
 3: Course: CS 2050
 4: Date: Sept 21 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, preliminary
 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[5];
23:
24:         printHeader();
25:         for(int i=0; i<5; i++)
26:             inputData[i] = new data();
27:
28:         for(int i=0; i<5; i++)
29:             {
30:                 if (i == 0)
31:                     planAuthor = readData.nextLine();
32:
33:                 inputData[i].caseDescription = readData.nextLine();
34:                 inputData[i].opA = readData.nextLine();
35:                 inputData[i].opB = readData.nextLine();
36:                 inputData[i].result = readData.nextLine();
37:                 //inputData[i].performAddition();
38:                 //System.out.println("***");
39:
40:             }
41:
42:         /*for (int i = 0; i<5;i++)
43:         {
44:           int size = inputData[i].op1.length();
45:           for (int j = 0; j<size; j++)
46:         }*/
47:
48:
49:         for (int i=0; i<5; i++)
50:             {
51:
52:                 System.out.println("Case:" + inputData[i].caseDescription);
53:                 System.out.println("Operand 1: " + inputData[i].opA);
54:                 System.out.println("Operand 2: " + inputData[i].opB);
```

```
    55:                 System.out.println("Expected Result: " +
inputData[i].result);
    56:                 System.out.print("Result from stack arithmetic: ");
    57:                 inputData[i].performAddition();
    58:                 System.out.printf("\n\n");
    59:             }
    60:
    61:     }
    62:     public static void printHeader()
    63:     {
    64:     System.out.println("Author: Sergiy Kolodyazhnyy\tCS-2050,Fall-2015");
    65:     System.out.println("Project #4 - Big Addition\n\n");
    66:     }
    67:
    68: }
```

Data class

Big chunk of the work is done here. We use this class to create objects that will store information received from each file and then process it.

```
     1: public class data
     2: {
     3:
     4:     //String planAuthor;
     5:     String caseDescription;
     6:     String opA;
     7:     String opB;
     8:     String result;
     9:
    10:
    11:     public void performAddition()
    12:     {
    13:         stack numA = new stack();
    14:         stack numB = new stack();
    15:         stack sumStk = new stack();
    16:         //int carry = 0;
    17:         int result;
    18:
    19:         for (int i=0; i<opA.length(); i++)
    20:             numA.push(Character.getNumericValue(opA.charAt(i)));
    21:
    22:         for (int i=0; i<opB.length(); i++)
    23:             numB.push(Character.getNumericValue(opB.charAt(i)));
    24:
    25:
    26:         int carry = 0;
    27:         while( !(numA.isEmpty()) || !(numB.isEmpty())  )
    28:             {
    29:                 int num1 = numA.pop();
    30:                 int num2 = numB.pop();
    31:
    32:                 if ( num1 == -1 )
    33:                     {
    34:                         sumStk.push(num2 + carry);
    35:                    carry=0;// this ensures  carry is used only once
    36:                          // if there is one
    37:                         continue;
    38:                     }
```

```
39:                 else if (num2 == -1)
40:                     {
41:                         sumStk.push(num1 + carry);
42:                     carry=0; // same as above
43:                         continue;
44:                     }
45:
46:
47:                 int sumNums = num1 + num2 + carry;
48:           // This part considers the carry number
49:                 if (sumNums >= 10 )
50:                     {
51:                         sumNums = sumNums - 10;
52:                         carry = 1;
53:                     // this if statement considers special case where both
54:                     // numbers have carry resulting form adding highest
55:                     // digits, as in case of 512 + 512 = 1024
56:                         if (numA.isEmpty() && numB.isEmpty())
57:                             {
58:                                 sumStk.push(sumNums);
59:                                 sumStk.push(carry);
60:                                 break;
61:                             }
62:                     }
63:                 else
64:                     {
65:                         carry = 0;
66:                     }
67:                 sumStk.push(sumNums);
68:             }
69:
70:
71:
72:
73: //System.out.println();
74:
75:         for(int i = 0; i <sumStk.size; i++ )
76:             {
77:                 int poppedNum = sumStk.pop();
78:                 if ( poppedNum != -1 )
79:                     System.out.print(poppedNum);
80:             }
81:
82:
83:     }
84:
85: }
```

Stack class

Simple stack class using array. Provides a template for each opperand stacks, as well as the summation result stack.

```
 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:
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:
```

Program output

Due to lack of time, I only obtained 3 input files, including my own, on which I could test my program. Nonetheless, it appears to be sufficient , as the program performs as expected. The only requirement the program doesn't meet is printing the output with commas, which wasn't exactly my focus so far, but rather functionality.

Bellow is the command line output as recorded by Unix script command.

```
$ cat typescript
Script started on Mon 21 Sep 2015 09:32:46 PM MDT

CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ javac driver.java

CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ java driver serg-testfile.txt

Author: Sergiy Kolodyazhnyy  CS-2050,Fall-2015
Project #4 - Big Addition


Case:description:same size, single carry
Operand 1: 1024
Operand 2: 4096
Expected Result: 5120
Result from stack arithmetic: 5120

Case:description:different size
Operand 1: 512
Operand 2: 1024
Expected Result: 1536
Result from stack arithmetic: 1536

Case:description:operand>max int
Operand 1: 2147483648
Operand 2: 512
Expected Result: 2147484160
Result from stack arithmetic: 2147484160

Case:description:same size,multiple carry
Operand 1: 4096
Operand 2: 4096
Expected Result: 8192
Result from stack arithmetic: 8192

Case:description:same size, single carry,output>operands in size
Operand 1: 512
Operand 2: 512
Expected Result: 1024
Result from stack arithmetic: 1024


CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ java driver eric-ault-testfile.txt

Author: Sergiy Kolodyazhnyy  CS-2050,Fall-2015
Project #4 - Big Addition
```

```
Case:values have same number of digits
Operand 1: 12345
Operand 2: 67890
Expected Result: 80235
Result from stack arithmetic: 80235

Case:no carries
Operand 1: 22708
Operand 2: 75191
Expected Result: 97899
Result from stack arithmetic: 97899

Case:carry for every digit
Operand 1: 96747968
Operand 2: 86596352
Expected Result: 183344320
Result from stack arithmetic: 183344320

Case:one operand outside int range
Operand 1: 2249847139
Operand 2: 2252
Expected Result: 2249849391
Result from stack arithmetic: 2249849391

Case:each operand within int range but sum over int range
Operand 1: 2147483245
Operand 2: 502
Expected Result: 2147483747
Result from stack arithmetic: 2147483747


CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ java driver jaziel-pauda.txt

Author: Sergiy Kolodyazhnyy   CS-2050,Fall-2015
Project #4 - Big Addition


Case:same size operands
Operand 1: 8975
Operand 2: 2461
Expected Result: 11436
Result from stack arithmetic: 11436

Case:no carry addition
Operand 1: 132546
Operand 2: 54321
Expected Result: 186867
Result from stack arithmetic: 186867

Case:values within max size
Operand 1: 98126750
Operand 2: 2397514
Expected Result: 100524264
Result from stack arithmetic: 100524264

Case:both values greater than max int
Operand 1: 3980753124
```

```
Operand 2: 1324786590
Expected Result: 5305539714
Result from stack arithmetic: 5305539714

Case:one operand greater than max size
Operand 1: 4586138740
Operand 2: 591742866
Expected Result: 5177881606
Result from stack arithmetic: 5177881606


CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ exit

Script done on Mon 21 Sep 2015 09:33:40 PM MDT
```

Original Input files

```
CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ cat jaziel-pauda.txt
Test Plan Author: Jaziel Pauda
same size operands
8975
2461
11436
no carry addition
132546
54321
186867
values within max size
98126750
2397514
100524264
both values greater than max int
3980753124
1324786590
5305539714
one operand greater than max size
4586138740
591742866
5177881606

CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ cat eric-ault-testfile.txt
Eric Ault - Test Plan Author
values have same number of digits
12345
67890
80235
no carries
22708
75191
97899
carry for every digit
96747968
86596352
183344320
one operand outside int range
2249847139
2252
2249849391
each operand within int range but sum over int range
2147483245
502
2147483747
CURRENT DIR:[/home/xieerqi/bin/cs2/hw4]
$ cat serg-testfile.txt
test file author: Sergiy Kolodyazhnyy
description:same size, single carry
1024
4096
5120
description:different size
512
1024
```

```
1536
description:operand>max int
2147483648
512
2147484160
description:same size,multiple carry
4096
4096
8192
description:same size, single carry,output>operands in size
512
512
1024
```