ASSIGNMENT 1

COMP-202A, Fall 2013, All Sections

Due: September 23rd, 2013 (23:30)

Please read the entire pdf before starting.

You must do this assignment individually and, unless otherwise specified, you must follow all the general instructions and regulations for assignments. Graders have the discretion to deduct up to 10% of the value of this assignment for deviations from the general instructions and regulations. These regulations are posted on the course website. Be sure to read them before starting.

Part 1:	0 points
Part 2, Question 1:	30 points
Part 2, Question 2:	30 points
Part 2, Question 3:	40 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment through automated tests. While these tests will not determine your entire grade, it will speed up the process significantly, which will allow the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks :)

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions.*

Warm-up Question 1 (0 points)

Create a file called `HelloWorld.java`, and in this file, declare a class called `HelloWorld`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display "`Hello world!`". You can find such a class in the lecture slides; make sure you can compile and run it properly.

Warm-up Question 2 (0 points)

Create a file called `A.java`, and in this file, declare a class called `A`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display the following pattern:

```
    A
   A A
  AAAAA
 A     A
A       A
```

Warm-up Question 3 (0 points)

Consider the following 2-d matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Write a Java program that first reads 4 doubles, representing a,b,c, and d from the keyboard. It then outputs to the screen the determinant of the matrix.

For a 2x2 matrix, the determinant is always equal to $a * d - b * c$

Warm-up Question 4 (0 points)

Now consider the same question except on a 3x3 matrix:

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Write a Java program that first reads 9 doubles, representing the 9 letters above from the keyboard. It then outputs to the screen the determinant of the matrix.

For a 3x3 matrix, the determinant is always equal to

$$a * (i * e - f * h) - b * (d * i - f * g) + c * (h * d - e * g)$$

Warm-up Question 5 (0 points)

Write a Java program which asks the user to enter a 4 digit number and then prints the sum of the 4 digits. For example if the user enters 1235 your program should print 11. See Part 2, Question 2 for hints on how to do this.

Warm-up Question 6 (0 points)

Without using any if statements write a Java program which asks the user to enter a 3 digit number and prints `true` if all the digits of the number are the same and `false` otherwise. See Part 2, Question 3 for hints on how to do this.

Part 2

The questions in this part of the assignment will be graded.

Question 1: Cell phone woes (30 points)

The following code should be placed in a class `CellComparison` *and thus a file* `CellComparison.java`

Marvin is a robot who has just landed on the planet Earth using his spaceship *Heart of Gold*. He has analyzed human life and understands everything about it except for one thing: cell phone plans.

He went to a wireless provider *Bellgers* who told him that there were two plans he could choose between:

1. Plan A: Marvin is given a free IPhone, but he needs to pays 50 dollars a month to use it plus an additional 10 cents per minute of talking.
2. Plan B: Marvin must pay 500 dollars up front for a new IPhone. However, he can pay only 25 dollars per month to use it and must pay 5 cents per minute of talking.

Marvin can not figure out which plan is better because he doesn't know ahead of time how long he'll stay on Earth or how many minutes he'll use each month! He can't figure out why they make it so complicated! He wants to figure out which plan is best for him in several different situations based on how long he stays on Earth and how many minutes he uses each month.

Being a friendly Earthling, you decide to help Marvin out by writing a computer program `CellComparison` in which you do the following.

First, ask Marvin to enter how many months he will use the phone for (only integers are allowed) and how many minutes he plans to talk *per month* (also integers only). The program should then calculate how much money he will spend if he bought either of those plans and then display the result to the screen.

A sample run of the program is below. Where it says "User typed this" it means that the line does NOT have to be produced by your program itself.

```
dans-MacBook-Pro:solutions dan$ java CellComparison
How many months will you use the phone?
20 <---- User typed this
How many minutes will you talk per month?
500 <---- User typed this
Under the first plan, you will pay $2000.0.
Under the second plan, you will pay $1500.0.
```

It is very important that your output look as similar to the above example as possible. You must read the 2 numbers (months and minutes) in the order above. This will make it faster for the TAs to grade. However, it is not important if your results have extra decimal points or unusual rounding issues. For example, your results may show up as $1999.999999 instead, which is fine.

Feel free to be creative with your messages as long as the layout is similar.

Hint: Remember that if you write `System.out.print()` instead of `System.out.println()` that the next print statement's text will appear on the same line. Alternatively you can use the + operator to put both text and a number on one line.

Also note that if the user enters illegal values (such as non-integer values), then it is OK if your program crashes or outputs the wrong value.

Question 2: A mathematical magic trick (30 points)

For this question, to be fair to all students who haven't seen more advanced topics, you are not allowed to use if statements, Strings (except for print statements), loops or any library objects other than Scanner

The following code should be in a class called `MathTrick` *and thus a file called* `MathTrick.java`

In this problem, you will demonstrate a magic trick using math.

1. Think of a 3 digit number in which the digits are decreasing (e.g. 952 or 521 but not 887 or 231).
2. Now reverse that number (so 852 becomes 258)
3. Subtract the number you got in step 2 from the number you wrote in step 1. (852 - 258 = 594)
4. Reverse the number you obtained in step 3. (594 becomes 495)
5. Add the number from step 3 to the number in step 4 (594 + 495 = 1089)

Let me guess. The number you got was 1089.

It turns out this "magic" trick is due to some properties of numbers which makes it so that *any decreasing 3 digit number* will have this property. A few examples can be found at `http://www.basic-mathematics.com/number-trick-with-1089.html`. An explanation (not required for the assignment) can be found at `http://www.mathsisfun.com/1089algebra.html`

In this question, you will write a computer program `MathTrick` in which you demonstrate this trick on a number the user types in.

1. First, print to the screen some magic words (such as "Abracadabra!"), and then ask the user to enter a 3 digit number with decreasing digits. (You may assume the user follows your instructions. If they do not, it does not matter what your program does)
2. Next, go through the above, step by step to print the results at various steps and show that indeed the result is 1089.

A sample run of the program is below:

```
 dan:~$ java MathTrick
Abracadabra hocus pocus, I predict that the last step will result in the number 1089.
Enter a 3 digit number in which all digits are decreasing.
852 <------The user typed this
Your number reversed is 258
The difference is 594
The difference reversed is 495
The total sum is 1089
```

Hint: To reverse a number, it is useful to know a couple of tricks.

1. Any time you take an integer modulo (%) 10, you will get the ones digit of a number.
2. Any time you divide an integer by 10, you will shift all the digits of a number so that the 10s become the ones, the 100s become the 10s, *etc.*
3. To extract the 100s digit of a number, for example, you can combine the above two tricks: First divide the number by 100 to "slide" the hundreds digit into the correct place. Then take the result % 10 to get the ones digit of the number.
4. Once you have extracted the individual ones, tens, and hundreds digits, you can reverse a number by keeping in mind that in the new number, what had previously been in the ones column (and worth 1) is now worth 100, what had previously been worth 10 is still worth 10 and what had previously been worth 100 is now worth 1. So if the original number looked like ABC where A, B, and C are all one digit numbers, then the new number is C*100 + B*10 + A

Question 3: Calculating consecutive numbers (40 points)

For this question, to be fair to all students who haven't seen more advanced topics, you are not allowed to use if statements or Strings (except for print statements) or loops or any library objects other than Scanner

The following code should be put into a file `ConsecutiveChecker.java`

The numbers 1234, 321, and 456 are all examples of numbers in which all of the digits are consecutive digits.

Write a Java program that asks the user to enter a number and then tells him or her whether it is a number with all consecutive digits or not. You may assume the user enters a positive integer that is at most 4 digits long. Note that 0 does NOT count as a positive number so you don't need to worry about this case. That is, your program does not need to worry about what happens if the user enters a value that is longer than 4 digits, negative, or zero.

Note the following hints:

1. If you have a boolean variable `x`, you can print it as you would an integer variable by writing `System.out.println(x)`. The word `true` or `false` will print to the screen depending on the value of x.
2. Remember the tricks from the previous question regarding how to extract individual digits using % and /.
3. Remember that `a == b` is true whenever a is equal to b and false otherwise.

4. Remember that `a != b` is true whenever a is NOT equal to b and false otherwise.
5. Remember that `a && b` is true only if BOTH a and b are true.
6. Remember that `a || b` is true as long as at LEAST one of a and b are true.
7. If the user types a leading zero for example `0345` this will be stored into the computer as the number 345. What this means is that your program will automatically treat 345 and 0345 as the same and so it is NOT necessary to do anything special to take care of this. That is to say, regardless of whether the user types 0345 or 345 your program should (and will automatically) say the same thing.

A general outline for solving this problem is given:

1. Store into 4 different `int` variables each of the 4 digits of the number.
2. Create a boolean variable `isOneDigitConsecutive` and come up with an expression to check if the number is a one digit consecutive number. There may be several correct expressions here: For example, a number is a one digit consecutive number if and only if the 10s, 100s, and 1000s digits all are 0. An alternate check would be whether the number is less than 10.
3. Repeat the previous step for 2, 3, and 4 digits. For example, for 2 digits, an expression could be that the 100s and 1000s digits are equal to 0 and the tens and ones digits are exactly one apart.
4. A number is a consecutive number if it is a one digit consecutive number OR a two digit consecutive number OR a three digit consecutive number OR a four digit consecutive.

A sample run of the program looks as follows:

```
dan$ java ConsecutiveChecker
Enter a 4 digit or less positive number.
3456 <---- User typed this
Is the number consecutive? true
```

What To Submit

You should submit your assignment on MyCourses. In order to do this, you will need to make a `zip` of the file. You can do this on windows by following the instructions at this link: `http://condor.depaul.edu/slytinen/instructions/zip.html`. On a mac or linux, you can find instructions at `http://osxdaily.com/2012/01/10/how-to-zip-files-in-mac-os-x/`

You should submit a zip file called `Assignment1.zip` with the following files inside of it.

`CellComparison.java`
`MathTrick.java`
`ConsecutiveChecker.java`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise he or she would not.