Programming Assignment 3

Due Sun. Jun 28, 11:55 pm

When looking at a program output below, the underlined part is what the user should type in the command-line.

Problem 1 – Input Format Conversion

Write a program, InputConversion, that asks the user to enter his name and age as follows: FirstName LastName, Age, and DateOfBirth in the following format: MM/DD/YYYY
For example:

```
jon stewart, 52, 11/28/1962
```

your prgram should then parse the input and produce the following output:

```
Your name is Jon Stewart.
You are 52 years old.
Your date of birth is: November, 28, 1962.
```

A sample program run looks like this:

```
Name: jon stewart
Age: 52
DOB: 11/28/1962
Your name is Jon Stewart.
You are 52 years old.
Your date of birth is: November, 28, 1962.
```

Make your prgram as modular as possible

Problem 2 - The Case of Cases

Whenever we find ourselves writing we identify three cases: Lowercase, Uppercase and Titlecase. Lowercase means that all letters are lowercase letters, uppercase means that all letters are uppercase, and Titlecase means that every word has the first letter only uppercased while the rest are lowercased. Create a file called **Cases.java**

We first make the following assumption: In any character encoding set, the letters are always assigned integers that are close to each other, and the range does not contain any other letters. For instance if “A” is 65, “B” is 66, and “Z” would be 90 (65 + 25) since we have 26 letters.
However you will **not** find a character that is not a letter in between 65 and 90. We also assume that the characters are ordered: A will always be before B.

For this exercise you need to assume that you do not know which number is assigned to which letter, but the fact that they are always next to each other should mean something. You are not allowed to use the Digit class.

Write a first method: **boolean isLetter(char c).** This method should return true if the letter is in the range of a-z and A-Z. Remember that characters are numbers and internally you can do things like ('a' + 1).
Examples: isLetter('a') returns true, but isLetter('~') returns false.

Write a method: **char lowerCase(char c)**. This method should return the lowercase version of the character **c**. If **c** is not a letter, then it should return **c** itself.
Examples: lowerCase('A'), returns 'a', while lowerCase('~') returns '~'.

Write a method: **char upperCase(char c)**. This method should return the uppercase version of the character **c**. If **c** is not a letter, then it should return **c** itself.
Examples: upperCase('b'), returns 'B', while upperCase('~') returns '~'.

Write a method: **String toLower(String s)**. This method returns a lowercase version of the string **s**. If **s** contains characters that aren't letters, it should return them unmodified.
Examples: toLower("Bird <Flies>") returns "bird <flies>"

Write a method: **String toUpper(String s)**. This method returns the uppercase version of the string **s**. If **s** contains characters that aren't letters, it should return them unmodified.
Examples: toUpper("Bird <Flies>") returns "BIRD <FLIES>"

Write a method: **String title(String s)**. This method returns the Titlecase version of the string **s**. If **s** contains characters that aren't letters, it should return them unmodified.
Examples: title("Bird <Flies>") returns "Bird <flies>" (not the "<" is the first letter of the word)
title ("a WeIrD PropHeCy") returns "A Weird Prophecy"
Your program should take as input from the command line a number **n**. And should read from standard input a string **sentence** by prompting: "Enter Sentence: "**.** Your program then should print **sentence** depending on the value of **n:** 1 = lowercase, 2 = uppercase, 3 = titlecase.

Example:

```
> java Cases 1
Enter Sentence: ThE coDe Is 1A134
the code is 1a134

> java Cases 2
Enter Sentence: ThE coDe Is 1A134
THE CODE IS 1A134

> java Cases 3
Enter Sentence: ThE coDe Is 1A134
The Code Is 1a134
```

Make sure that your program is as modular as possible. You might need to create more methods than mentioned.

Problem 4 – Carry Count

Write a java program, `CarryCount`, that takes two numbers between 1 and 1,000,000 from the command line and then finds the number of carry operations that occur when adding the two numbers. For instance when adding 678 and 322, the number of carry operations is 3. The program should define a method, `calcCarry`, with the following signature to find the carry count:

```
int calcCarry(int, int)
```

Following is a sample run of the program:

```
> CarryCount 364 394
The number of carry operations when adding 364 and 394 is 1.
```

Problem 5 – Reversing

Write a method `printReverse` that accepts a string as argument and prints the characters in opposite order. Use the method above to write a program, `Reversing`, that takes an integer command line argument N followed by N strings and prints each of the string in its reverse order.

```
> java Reversing 3 "Test One" Second "This is an output"
enO tseT
dnoceS
tuptuo na si sihT
```

Problem 6 - Mirrors

You are to write a program that reverses a portion of a string. Your program asks the user to input a number **n** and a **sentence**. And will then reverse every **n** characters of the sentence at once. Make your program as modular as possible.

```
> java Mirrors
n: 3
sentence: teMdohm sekail  efyawae eis.r
Methods make life way easier.
```

Problem 7 - Thousand and One Prime

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. What is the 10001st prime number?

Your output should include **ONLY** the number

Submission Instructions and Notes

- Submit your commented source code in a zip file to Moodle by the deadline.
- Your zip submissions should be named `asst3_`*`netid`*, where *`netid`* stands for your AUBnet user name. For example, if your AUBnetid is abc65, you should call your submission asst2_abc65.