German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Wael Aboul Saadat

April 7, 2016

CSEN 202: Introduction to Computer Programming
Spring term 2015-2016

Midterm Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Please tick your major

	Major
	Civil
	BI
	Engineering

2) Duration of the exam: 2 hours (120 minutes).

3) No books or other aids are permitted for this test.

4) This exam booklet contains 11 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.

5) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.

6) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below ;-)

Exercise	1	2	3	$\sum$
Possible Marks	40	13	10	63
Final Marks				

Exercise 1 UPC Code (8+10+6+6+10=40 Marks)

Note: You need to read the whole question thoroughly then start to solve.

You will write a Java program that checks UPC strings to see if they are valid. Your program should first prompt the user to enter a string of numbers as a UPC code, or enter `Q` to quit the program. If the user does not quit, your program should ensure that this string is exactly 12 characters in length. Assume that the string consists of integers only.

Your program should use the algorithm below to compute what the check digit should be (i.e expected check digit) and then compare it to the actual value of the check digit in the provided string (where the actual check digit is the rightmost digit in the string) and report whether the UPC is valid or wrong. If it is wrong, your program should report what the correct check digit should be for the input UPC and print that it is invalid. Your program should keep asking for new UPC until the user enters a `Q` to quit the program.

The algorithm for checking for a valid UPC is:

1. From left to right, add the digits in the odd-numbered positions (starting the count from 1) and then multiply the result by 3.
2. From left to right, add the digits in the even-numbered positions. You should not include the last digit in your calculation (i.e. rightmost digit in the string should be not included). Then, add the total of the even-numbered positions to the total computed in step 1.
3. Take the result from step 2 (i.e. sum) and compute the remainder when divided by 10 (result modulus 10). If the remainder is not zero, subtract this remainder from 10 to get the expected check digit. If the remainder is zero, then the expected check digit should be 0.

This is a sample example of what your program should do.

```
Enter a UPC (or enter Q to quit): 036000291453
Check digit should be: 2
Check digit is: 3
UPC is not valid

Enter a UPC (or enter Q to quit): 036000291452
Check digit should be: 2
Check digit is: 2
UPC is valid

Enter a UPC (or enter Q to quit): 014633149077
Check digit should be: 4
Check digit is: 7
UPC is not valid

Enter a UPC (or enter Q to quit): 014633149074
Check digit should be: 4
Check digit is: 4
UPC is valid

Enter a UPC (or enter Q to quit): 0853911765722
Enter a UPC (or enter Q to quit):

Enter a UPC (or enter Q to quit): 085391176572
Check digit should be: 2
Check digit is: 2
UPC is valid

Enter a UPC (enter a blank line to quit): Q
Goodbye!
```

For this assignment, you should implement and then use the following methods. Read all parts then start to solve:

a) Write a method that uses Scanner to get the input string from the user. If the user does not enter `Q` (i.e. does not quit), your program should ensure that this string is exactly 12 characters in length. Assume that the String consists of integers only. If the user enters a string that is not 12 characters in length, your program should ask again till the user enters a valid string. The method should return the input string.

For example:

```
getString() --> returns: "036000291453"
getString() --> returns: "014633149074"
getString() --> returns: "Q"
```

```
public static String getString() {
   Scanner sc =  new Scanner(System.in);
```

Solution:

```
public static String getString() {
     Scanner sc = new Scanner(System.in);
     String upc;

     do
     {
         System.out.println("Enter a UPC(or enter Q to quit):");
         upc = sc.nextLine();

    }while(upc.length()!=12&& !upc.equals("Q"));

    return upc;

}
```

b) Write a method that given the inputString, returns the sum. The sum is calculated as follows:

1. From left to right, add the digits in the odd-numbered positions (starting the count from 1) and then multiply the result by 3.
2. From left to right, add the digits in the even-numbered positions. You should not include the last digit in your calculation (i.e. rightmost digit in the string should be not included). Then, add the total of the even-numbered positions to the total computed in step 1.

NOTE: For this method you will need to use the `Character.getNumericValue()` method. This method takes a character value such as `'2'` and converts it to the appropriate integer value (in this case the number `2`). For example:

```
char input='2';
int x = Character.getNumericValue(input); // x now has a value 2
```

For example:

```
calculateSum("036000291453") --> returns: 58

calculateSum("014633149074") --> returns: 86
```

```
public static int calculateSum(String inputString) {
```

Solution:

```
  public static int calculateSum(String inputString) {

     int odd = 0;
     int even = 0;
     int i;
     for(i = 0;i < inputString.length()-2;i+=2)
     {
          odd += Character.getNumericValue(inputString.charAt(i));
          even += Character.getNumericValue(inputString.charAt(i+1));
     }

     odd += Character.getNumericValue(inputString.charAt(i));
     odd *=3;
     int sum = odd+even;

     return sum;

    }
```

c) Write a method that given the sum, **prints** then returns the expected checkdigit. The expected checkdigit is calculated as follows: Take the sum and compute the remainder when divided by 10 (result modulus 10). If the remainder is not zero, subtract this remainder from 10 to get the expected check digit. If the remainder is zero, then the expected check digit should be 0. This method returns int.

For example:

```
 getExpectedCheckDigit(58) --> prints:"Check digit should be: 2" and
                               returns: 2

 getExpectedCheckDigit(86) --> prints:"Check digit should be: 4" and
                               returns: 4
```

```
public static int getExpectedCheckDigit(int sum) {
```

Solution:

```
public static int getExpectedCheckDigit(int sum) {

     int expected = 0;

     if(sum%10 !=0)
          expected = 10 - sum%10;

     System.out.println("Check digit should be: "+expected);
     return expected;

}
```

d) Write a method that given the **expected** checkdigit and the inputString, prints the check digit in the inputString and then checks whether the `UPC` is valid or not. The check digit in the inputString is the rightmost digit in the string. The `UPC` is valid if the rightmost digit in the inputString (i.e. check digit in the string) is equal to the **expected** checkdigit, otherwise it is invalid. The method returns a string.

Note: You should use conditional operator for the check.

For example:

```
checkValid(2,"036000291453") --> prints: "Check digit is: 3" and
                                 returns: "UPC is invalid"

checkValid(4,"014633149074") --> prints: "Check digit is: 4" and
                                 returns: "UPC is valid"
```

```
public static String checkValid(int checkDigit, String inputString) {
```

Solution:

```
public static String checkValid(int checkDigit, String inputString) {

     System.out.println("Check digit is: "+ inputString.charAt(11));
     return (checkDigit==Character.getNumericValue(inputString.charAt(11)))?
     "UPC is valid":"UPC is not valid";

}
```

e) Write a `main` method to combine all the methods implemented and to run like the sample example mentioned above. You are requested to use all methods you implemented before.

Solution:

```
public static void main(String[] args) {

     String upc;
     int sum;
     int ex;
     String print;

    do
    {
          upc = getString();
          if(upc.charAt(0)!= 'Q')
          {
               sum = calculateSum(upc);
               ex = getExpectedCheckDigit(sum);
                print = checkValid(ex, upc);
               System.out.println(print);
        }

    }while(upc.charAt(0)!='Q');
```

```
        System.out.println("Goodbye!");
    }
```

Exercise 2 Tracing (1.5+2.5+2+2+2+3=13 Marks)

a) Compute the value of the following Java expression. Show your workout.

```
((byte)(127+1) > 0) && (true == false) || (21.0/4.0 > 5)
```

Solution:

```
true
```

b) What is the output of the following lines of Java code?

```
int a = 9;
double b = a / 2.0;
int c = a / 2;
double d = a / 2;
String e = c + "c";
a = a % 6;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);
```

Solution:

```
a = 3
b = 4.5
c = 4
d = 4.0
e = 4c
```

c) If in the following statements the casting to `float` is removed would this raise an exception? Explain what would happen with and without the casting and what would be printed by the third statement in the two cases:

```
int total = 45;
float result = (float) total / 7;
System.out.println (result);
```

Solution:

No exception, result would be assigned to 6.0 (without cast) and 6.428 (with cast).

d) Write the output of these statements:

```
int base = 1;
int count = 3;
if (base++ < count || count < 4 )
System.out.println(count);
System.out.println(base);
```

Solution:

```
3
2
```

e) Write the output of these statements:

```
int a = 2;
System.out.println(a++ + Math.pow(a, 2));
a = 2;
System.out.println(Math.pow(a, 2) + a++);
```

Solution:

```
11.0
6.0
```

f) How many times will the `System.out.println("*");` statement execute inside of the following nested for-loops? What will be printed?

```
for (int j=0; j<10; j++)
{
   for (int k=10;k>j*2;k--)
      System.out.print("*");
   System.out.println("");
}
```

Solution:

Answer: 30 Explanation: The first iteration of the outer loop has j = 0, so the inner loop iterates from 10 down to 1, or 10 times. The next iteration of the outer loop has j = 1, so the inner loop iterates from 10 down to 3, or 8 times. This continues until the outer loop has j = 5, in which case the inner loop iterates 0 time. Thus, the System.out.println statement executes a total of 10 + 8 + 6 + 4 + 2 = 30 times.

```
**********
********
******
****
**
```

The loop stops here.

Exercise 3 (4+6=10 Marks)

We would like you to write two methods that will allow us to count up the number of occurrences of all of the letters in a String.

a) Write a method `numOccurs(letter,source)` which returns the number of times that a character named letter occurs in source.

```
public static int numOccurs(char letter, String source)
```

Solution:

```
public static int numOccurs(char letter, String source)
{
    int count = 0;

    for (int i = 0; i < source.length();i++)
    {
        if (source.charAt(i) == letter)
            count ++;
    }

    return count;
}
```

b) Write a method `countLetters(source)` which returns a String with 26 entries where every entry represents the number of occurrences of each letter `'a'` through `'z'` in source.

For example, for the following source `aabbbdaaaazaaadddaaabbbbbzd` the method should return

```
12 8 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
```

Use the method `numOccurs` above to simplify your code

```
public static String countLetters(String source)
```

Solution:

```
public static String countLetters(String source)
{
    String out = "";

    for (int i = 'a'; i <= 'z'; i++)
    {
        out += numOccurs((char)i,source);
        out += " ";
    }

    return out;
}
```

Scratch paper

Scratch paper

Scratch paper