LSUS | Department of Computer Science

318.795.4266 (Fax) 318.795.2419
One University Place
Shreveport, LA 71115-2399

CSC 135, Spring 2020
Dr. Urska Cvek
Final Project
Assigned: 4-14-20
Due: 4-30-20 by 12pm

This is an individual project, all work should be your own work!

The United States Post Office handles millions of pieces of mail each day. After mail is picked up it is taken to a post office and is sorted by zip code. The mail destined for zip codes in the same region of the country are loaded on trucks and shipped to a "hub" post office in the destination region. All of the mail arriving at a "hub" post office is again sorted by zip code. Then based on the zip code this sorted mail is shipped to individual post offices in the region and finally delivered to the recipient. If looking at all of those zip codes sounds like a lot of work, you are correct! To help with this process the post office developed a method for translating zip codes into barcodes. These barcodes can then be read by computers which then automatically sort the mail. Today they call this "Intelligent Mail Barcode for Mail Pieces" and the description of this process is located in a file located after this final project assignment (if you wanted to know the details).

When your outgoing mail arrives at your local post office it is sent through a machine that reads the recipient's address from the envelope. The address is then translated into a barcode representing the address, and the barcode is printed on the envelope. The barcode is easier for machines to read and therefore enables these machines to sort the mail more quickly. Thus, when your letter arrives at a "hub" post office it can be sorted very quickly by just scanning the barcodes printed on the letter. In implementing this system, a critical piece of software is needed that can convert zip codes into barcodes, which is going to be accomplished by your project.

You will be decoding a nine-digit ZIP+4 code. Every barcode is framed by a tall frame bar at the beginning and a tall frame bar at the end (left frame bar and right frame bar). In between there are the 9 digits of the ZIP code plus the correction "check-bar", each encoded by a 5-bar frame bar item. A frame bar item consists of tall (length 1, full size) and short (length 0, half size) lines. The figure below shows an example how ZIP+4 code 12345-6789 would be encoded:

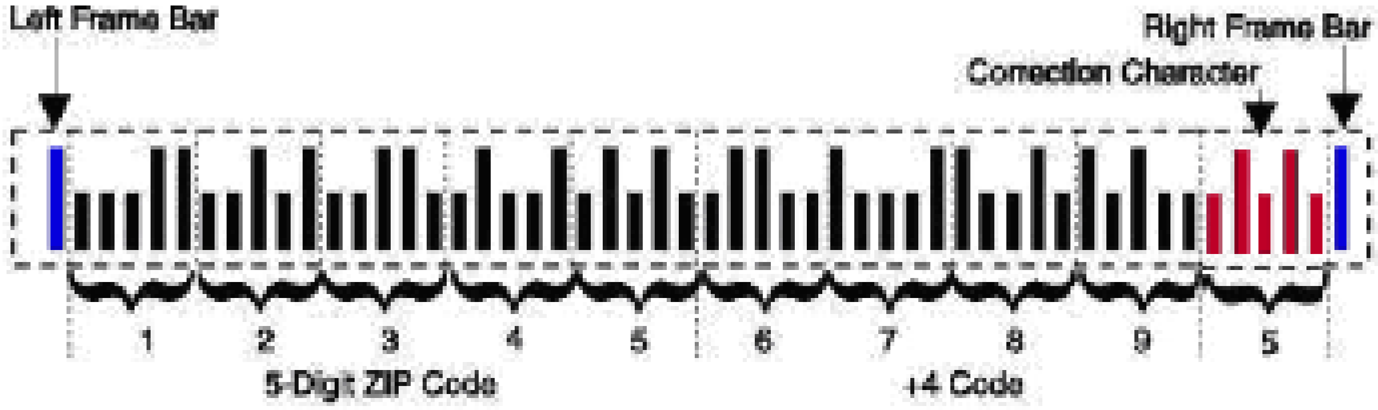
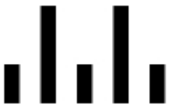


Figure: encoding of the delivery point barcode 12345-6789

Frame bars
Every barcode begins and ends with a tall frame bar (length 1, full size). They serve no purpose other than as end markers for the barcode itself. They are displayed as **blue** bars.

Digits of the ZIP code
Each digit of the zip code (9 digits total) is encoded by a 5-bar frame bar item, or five positional bars. The five bars of the item represented by weights or values 7, 4, 2, 1, and 0, respectively. Each digit is represented ALWAYS as a 2 of 5 code using positional weights, which means that 2 of the bars are tall (length 1, full size) and 3 other bars are short (length 0, half size) to represent a digit. For example, to represent a digit 5 we would represent the second and fourth bar as tall, and the other three bars as short:



$7\ \ \underline{4}\ \ 2\ \ \underline{1}\ \ 0 \qquad \underline{4} + \underline{1} = 5$

The following table represents each of the 10 individual digits that you need to be able to represent, where a 1 represents a full (length 1, full size) bar and a 0 represents a short (length 0, half size) bar:

	First bar: 7	Second bar: 4	Third bar: 2	Fourth bar: 1	Fifth bar: 0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Table: Digit representation using five positional bars
Please, note that 0 digit does not add up to 0 but rather to an 11 (which cannot be a digit, but two digits) which is the standard annotation.

Correction Character
The last number encoded in every barcode is a correction character. This is in addition to the ZIP+4 code information, and is appended at the end, just before the right frame bar. The correction character serves as an on-the-fly check digit for the barcode sorters and is used to confirm the accuracy of the barcode information as read by the machinery. The correction character is derived by adding the sum of all the numbers encoded in the barcode and subtracting that number from the next whole integer of 10. Example:

ZIP+4 Code = 11011-9000
1 + 1 + 0 + 1 + 1 + 9 + 0 + 0 + 0 = $\underline{13}$

Since the sum is 13, then the next whole integer of 10 is 20. (If the sum had been 23, then the next whole integer of 10 would have been 30. And so on....)
20 - $\underline{13}$ = 7

Therefore, for this example, the proper correction character for this ZIP+4 barcode would be 7. The number would be encoded in the last set of five tall and short bars, just before the final tall frame bar.

In the above example barcode in the Figure ZIP 12345-6789 would sum to 45 then the next whole integer of 10 is 50. Therefore, the proper correction character for this ZIP+4 barcode would be 5, as depicted in the figure.

Task

Write a program that asks the user for a ZIP code and prints the bar code. Use symbol : for a short bar and | for a full bar, just as indicated by the table above. **Do your best job to take advantage of object-oriented programming and divide your code into multiple classes as we have learned in class. I expect to have more than just one .java class submitted for the solution.**

Name your project *FinalProject_ LastName_FirstName*, and include at least the following:

The user is prompted to enter a nine-digit ZIP code using the Scanner class. Loop the process so that the user can answer YES or NO whether they want to enter another ZIP+4 code.

Every valid zip code is displayed using characters/strings in the message section (using System.out.print and println methods).

Prompt the user if they want to enter another zip code.

When done, copy the project files on Moodle to submit in the Final Project.

Have fun and let me know if you have questions! ☺