JC2002 Java Programming - Practical 9 (Day 11)

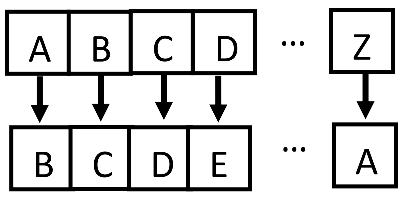
This practical walks you through a number exercises having to do with string manipulation in Java. Most of these can be simple, short programmes. For the later ones you might want to use Maven, so that you can integrate testing too. If you follow the maven path, then use instructions from previous practicals. For all of exercises today, you should work with the methods already in the Java **String** class.

1. Reversing words is a place to start. Write a small program that takes in a string from the user, and then prints out the words in their reverse order.
2. Use the **regionMatches()** method to compare parts of strings. Write a small program that takes two input strings from the user and then reports back on whether there is a match with part of them have a match. You can ignore case (uppercase or lowercase) in this task.
3. Write a program that takes in a line of text from the keyboard, and then prints a table indicating letter frequency, as shown in the example output below. It should not print anything if there are zero instances of a given letter. You should count uppercase and lowercase letters together.

   Example output:

   ```
   a: 5
   b: 2
   c: 1
   d: 4
   …
   z: 1
   ```

4. Let us assume that a personal ID code has format **DDMMYY-XXXX**, where DD are the digits of the day of birth (01-31), MM are the digits of the month of birth (01-12), YY are the last two digits of the year of birth (00-99), and XXXX is a personal identifier of four characters that can contain digits and capital letters. Use **matches()** method and regular expressions (*Regex*) to write a small program to test if given input is a valid personal ID code.
5. Write a program that reads a word from the user and encrypts it using shift cipher, where the letters are replaced by the following letter in the alphabet, as shown in the figure below.



   <u>Hint:</u> you can convert characters **a-z** to integers **0-25** using:

   int i = Character.digit(c, 36) – 10;

Respectively, you can convert integers **0-25** to characters **a-z** using:

```
char c = Character.forDigit( i + 10, 36 );
```

You can try also different shifts, e.g., shifting the alphabets by two positions, three positions etc.