DM 502 Programming A

Fall 2010 Project (Part 2)

Department of Mathematics and Computer Science
University of Southern Denmark

September 29, 2010

Introduction
The purpose of the project for DM502 is to try in practice the use of programming techniques and knowledge about the programming language Java on small but interesting examples. The project consists of two parts.

Please make sure to read this entire note before starting your work on this part of the project. Pay close attention to the sections on deadlines, deliverables, and exam rules.

Exam Rules
This second part of the project is a part of the final exam. Both parts of the project have to be passed to pass the overall project.

Thus, the project must be done individually, and no cooperation is allowed beyond what is explicitly stated in this document.

Deliverables

- A short project report in PDF format (at least 4 pages without front page and appendix) has to be delivered. This report has to contain the following 7 sections (for details see the slides for September 14):
  - front page
  - specification
  - design
  - implementation
  - testing
  - conclusion
  - appendix
- All source code.

The deliverables have to be delivered using Blackboard's Assignment Hand-In functionality. Delivering by e-mail or to the teacher is only considered acceptable in case Blackboard is down before the deadline.

Deadline

November 1, 12:00

The Problem

The Caesar cipher used in the first part of the project is not very secure. There are two reasons for this. First, there are only 26 possible shifts (and one does not change the plaintext). So it is possible to go through the 25 interesting shifts and look for the plain text.

Second, and more importantly, the letter frequency distribution of a natural language text is not random. In English, for example, the letter "e" is by far the most common letter. Thus, when you are sure the plaintext is in English, analyzing the letter frequency distribution of the encrypted text will easily reveal the key used.

Your task in this part of the project is to write a Java program that reads an encrypted English text from an input file given as a first argument and a letter frequency distribution for English as the second argument. Then, the English plaintext corresponding to the encrypted English text should be printed to the screen. To this end, you will have to determine the key that was used to encrypt the text. The key should be determined *without user interaction*. If an output file is given as a third argument, the plaintext should additionally be written to that output file.

The Input

The first input is an encrypted message containing arbitrary text, that is, any kind of characters are allowed. You may assume, though, that only the letters $A, B, \ldots, Z$ and $a, b, \ldots, z$ have been encrypted. Thus, our input might look like this:

> Puzaybjavyz jhu zluk lthps av hss vy zlsljalk pukpcpkbhs Bzlyz, Zabkluaz, Nyvbwz, Alhjopun Hzzpzahuaz, Puzaybjavyz vy Vizlyclyz pu h Jvbyzl. Myvt h Ishjrivhyk Slhyu jvbyzl, lthpsz jhuuva il zlua av hufvul dov pz uva h tltily vm aol jvbyzl.

The second input is a letter frequency distribution for English available as `engelsk.dat` from the home page.

The Output

The output of your solver should be the English plain text corresponding to the encrypted message:

> Instructors can send email to all or selected individual Users, Students, Groups, Teaching Assistants, Instructors or Observers in a Course. From a Blackboard Learn course, emails cannot be sent to anyone who is not a member of the course.

The Task

Implement a main method that handles the inputs from files and the output to the screen (and possibly to a file).

Write a class `Histogram` that uses a `HashMap` to keep frequency data. There should be constructors or static methods to build histograms from the given letter frequency data as well as from a given encrypted text.

Write a method for comparing the histogram obtained from the encrypted text to the histogram from the given letter frequency data. This method should return as a result the best guess for the key needed to decrypt the given encrypted text. For the above example, this method should return 7.

Write a method that returns a list of letters (and possibly frequencies) ordered by the frequency such that the most frequent letter comes first. This method should be used by the above method to determine the key. Use the interface Comparable or the interface Comparator for sorting the list.

You decide which method to use for determining a potential key. For inspiration, see the Wikipedia page on the Caesar cipher:

`http://en.wikipedia.org/wiki/Caesar_cipher`

Your implementation should at least be able to correctly decrypt the three texts `engelsk1.crypt, engelsk2.crypt, engelsk3.crypt`, which you can download from the course home page:

`http://www.imada.sdu.dk/~petersk/DM502/#project`