Department of Computer Science & Applied Physics

H.Dip. in Science (Software Development) - Object-Oriented Software Development (2025)

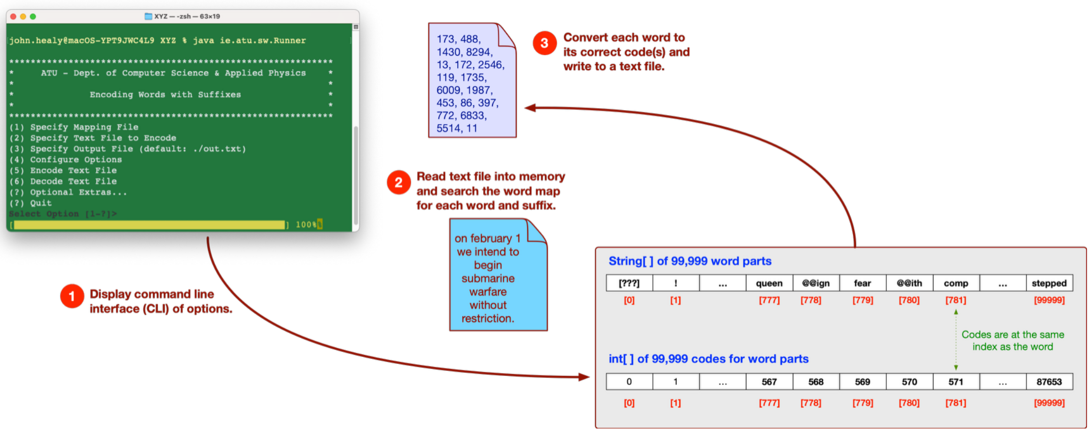
Encoding Words with Suffixes (50%)

ASSIGNMENT DESCRIPTION & SCHEDULE

YOU MAY NOT USE ANY GENERATIVE AI IN THIS ASSIGNMENT

Overview

Text encoding is the process of converting characters (letters, numbers and symbols) into a specific format for storage or transmission in computers. There are lots of different types of text encoding, e.g. ASCII, Unicode and ISO-8859-1 (Latin-1), but they all map characters to a numeric representation to that they can be stored, transmitted and processed by a computer. In this assignment, you are required to develop a Java application with a command line interface (CLI) that uses arrays of words and word suffixes to encode text files as numbers and decode them back to the original text. An overview of the process is shown below:



```
john.healy@macOS-YPT9JWC4L9 XYZ % java ie.atu.sw.Runner

*************************************************************
*     ATU - Dept. of Computer Science & Applied Physics     *
*                                                           *
*                Encoding Words with Suffixes               *
*                                                           *
*************************************************************
(1) Specify Mapping File
(2) Specify Text File to Encode
(3) Specify Output File (default: ./out.txt)
(4) Configure Options
(5) Encode Text File
(6) Decode Text File
(?) Optional Extras...
(?) Quit
Select Option [1-?]>
[                                                      ] 100%
```

XYZ — -zsh — 63×19
1 Display command line interface (CLI) of options.
2 Read text file into memory and search the word map for each word and suffix.
on february 1 we intend to begin submarine warfare without restriction.
3 Convert each word to its correct code(s) and write to a text file.
173, 488, 1430, 8294, 13, 172, 2546, 119, 1735, 6009, 1987, 453, 86, 397, 772, 6833, 5514, 11
String[ ] of 99,999 word parts
[???] ! ... queen @@ign fear @@ith comp ... stepped
[0] [1] [777] [778] [779] [780] [781] [99999]
Codes are at the same index as the word
int[ ] of 99,999 codes for word parts
0 1 ... 567 568 569 570 571 ... 87653
[0] [1] [777] [778] [779] [780] [781] [99999]

All file paths, variables and options should be entered into a command-line interface (CLI) by the user. A set of stubs, including a menu, is available on Moodle to help you get started.

A mapping of word and word suffixes (denoted with a prefix "*@@*") are provided in the file ***encodings-10000.csv***. Each of the 9,999 lines in the CSV file contains a word / suffix and its numeric value. The CSV file should be read into a set of arrays that act as a lookup table for words / suffixes and their identifiers. You can assume that the file ***encodings-10000.csv*** is in the current directory and is accessible as "***./encodings-10000.csv***". ***Do not alter the CSV file in any way and do not include the CSV file with your submission.***

The encoding process involves searching the arrays for words or parts of words and writing out their numeric values to a file. As shown below, decoding reverses the process.

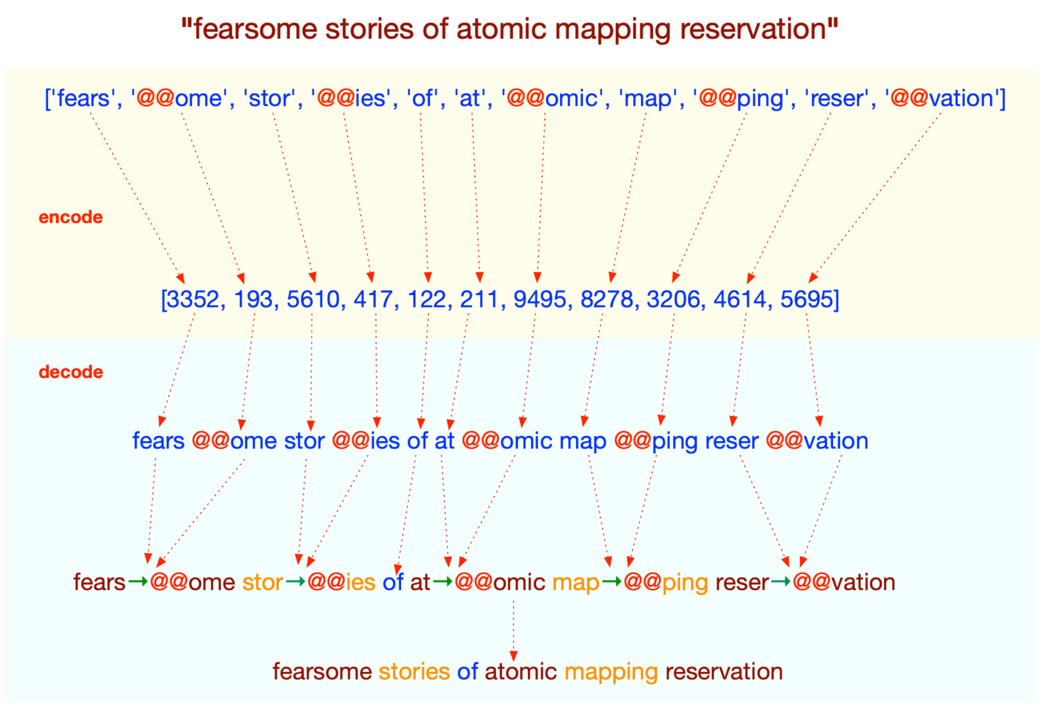

"fearsome stories of atomic mapping reservation"
['fears', '@@ome', 'stor', '@@ies', 'of', 'at', '@@omic', 'map', '@@ping', 'reser', '@@vation']
encode
[3352, 193, 5610, 417, 122, 211, 9495, 8278, 3206, 4614, 5695]
decode
fears @@ome stor @@ies of at @@omic map @@ping reser @@vation
fears→@@ome stor→@@ies of at→@@omic map→@@ping reser→@@vation
fearsome stories of atomic mapping reservation

Fig. 2

Use the entry **[???]→0** for any words that do not exist in the arrays as whole words or word parts. You will have to work out an algorithm yourself to encode and decode text as shown in Fig. 2. You are free to implement this part of the assignment in any way that you like, i.e. use any algorithmic approach.

Please note that ***you must use arrays for this assignment*** and not *ArrayLists* or any other classes from the *Java Collections Framework*. Understanding how to use arrays is a key learning outcome on this module and will greatly assist later in understanding how more complex data structures work.

Minimum Requirements

- **Use the package name *ie.atu.sw*** and place the *main()* method in a class called *Runner*.

- **Provide a simple command-line user interface** that enables a user to specify the following:
  1. A path and name for the text file to encode or decode.
  2. Switch between encoding and decoding modes.
  3. The path and name for the file to output.

  You can include as many other menu items as you wish and will be given marks for relevant functionality added.

- **Build a mapping** of the 9,999 words to their word suffixes in the file ***encodings-10000.csv***. This can be achieved by creating a set of arrays or a single 2D Object array.

- **Search** the arrays and encode each word as a set of one or more codes from the arrays. Reverse the process for decoding.

- **Output** the encoded or decoded text to a file.

- **Include comments** for each method in your application and indent / structure your code correctly.

- Provide a **README** file in **PDF** format detailing the main features of your application in 300-400 words. Marks will only be given for features that are described in the README. The README must be written in a reflective style and clearly state the rationale for each of the design choices that you have made. The README must reference the topics covered in lectures and labs and explain how they have influenced your implementation.

Deployment and Submission

Please **read the following carefully** and pay particular attention to the files that you are required to submit and those that you should not include with your assignment:

- ***The project must be submitted by 5pm on Sunday August 31st 2025***. Before submitting the assignment, you should review and test it from a command prompt on ***a different computer*** to the one that you used to program the project.
- The project must be submitted as a Zip archive ***(not a 7z, rar or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the *"Encoding Words with Suffixes - (50%) Assignment Upload"* link on Moodle. Only the contents of the submitted Zip will be considered. ***Do not add comments to the Moodle assignment upload form.***
- The name of the Zip archive should be <*id*>.zip where <*id*> is your ATU student number.

Do NOT submit the assignment as an Eclipse project or submit any text files or other resources. If you do, you will lose marks. You will also lose marks if you hard-code any environmental variables in your project like system paths and file names.

- The Zip archive should have the structure shown below.

Item	**Description**
src	A directory that contains the packaged **source code** for your application.
README.pdf	A **PDF file** detailing the main features of your application in **no more than 300 words**. Marks will only be given for features that are described in the README.

Scoring Rubric

Component	**Marks**	**Description**
Robustness & Design	50	The application deploys and executes correctly.
Documentation	50	Comments and supporting documentation.