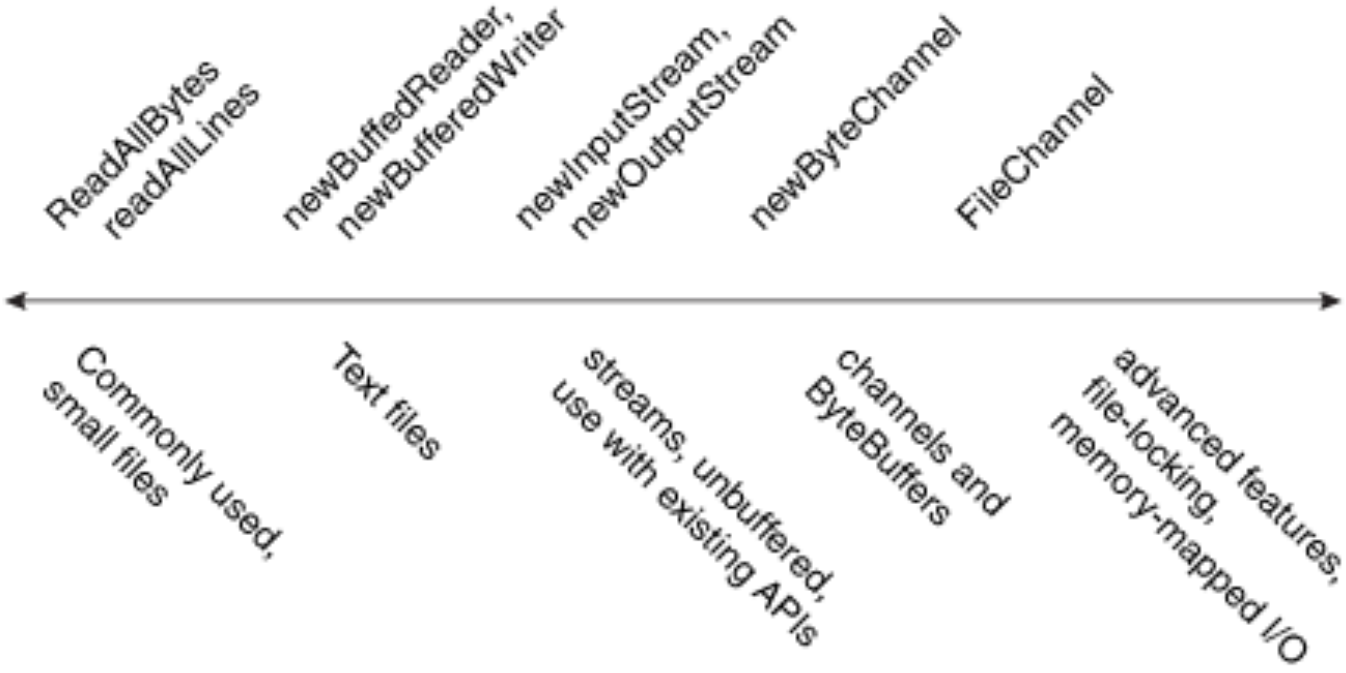
PRACTICAL OVERVIEW

FILE IO

- In first year you used files to read and save data on a disk
- You can review the basics of file IO here: https://docs.oracle.com/javase/tutorial/essential/io/file.html

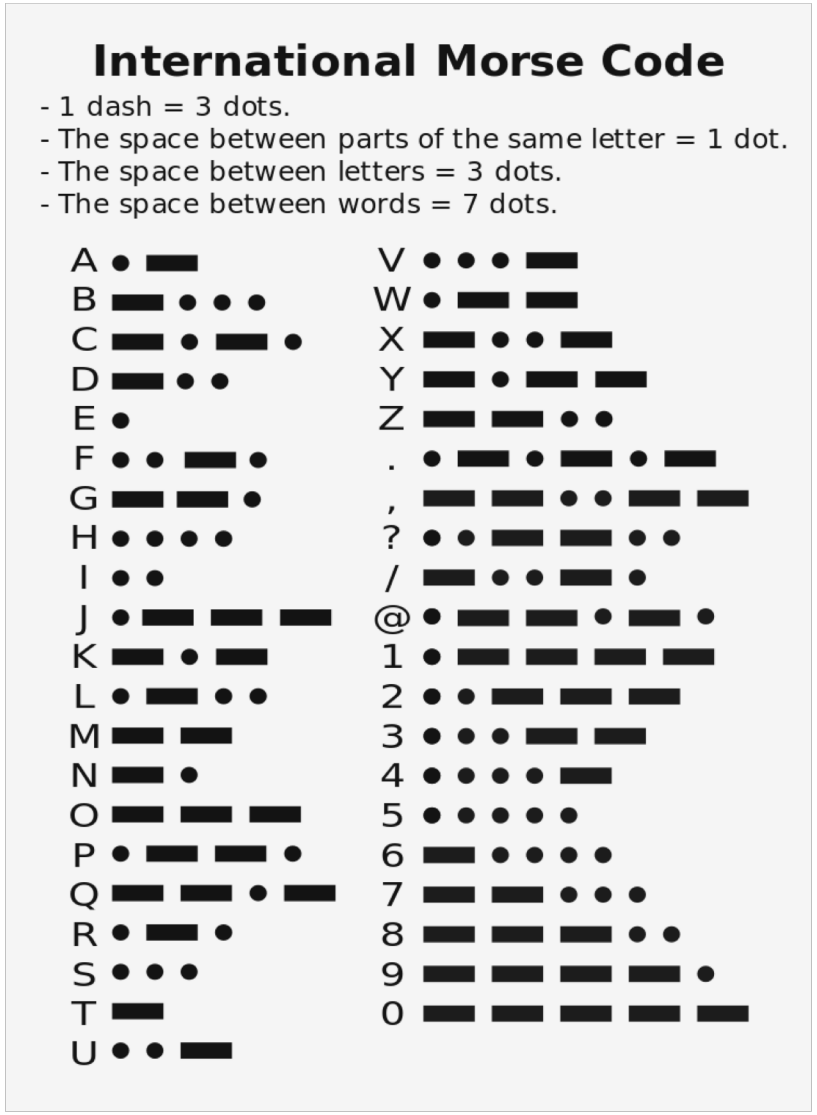
FILE IO

- There are many file I/O methods to choose from. Here are the file I/O methods arranged by complexity (left is simpler)
- In the lab we emphasize the use of exception handling to facilitate reading and writing from files in Java

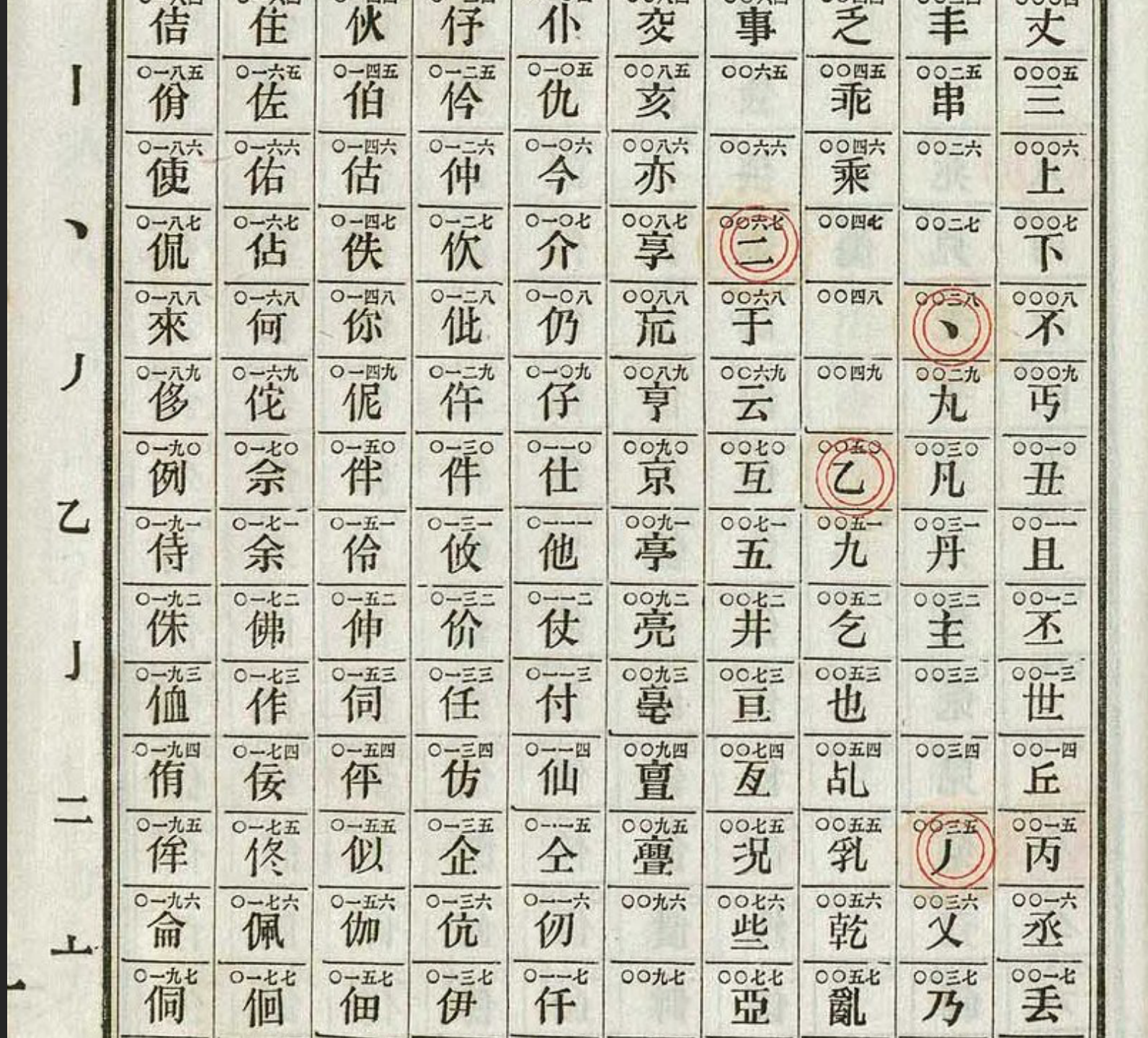


CHARACTER ENCODING

- The International Morse Code encodes the 26 English letters A through Z, some non-English letters, the Arabic numerals and a small set of punctuation and procedural signals (prosigns). There is no distinction between upper and lower case letters.

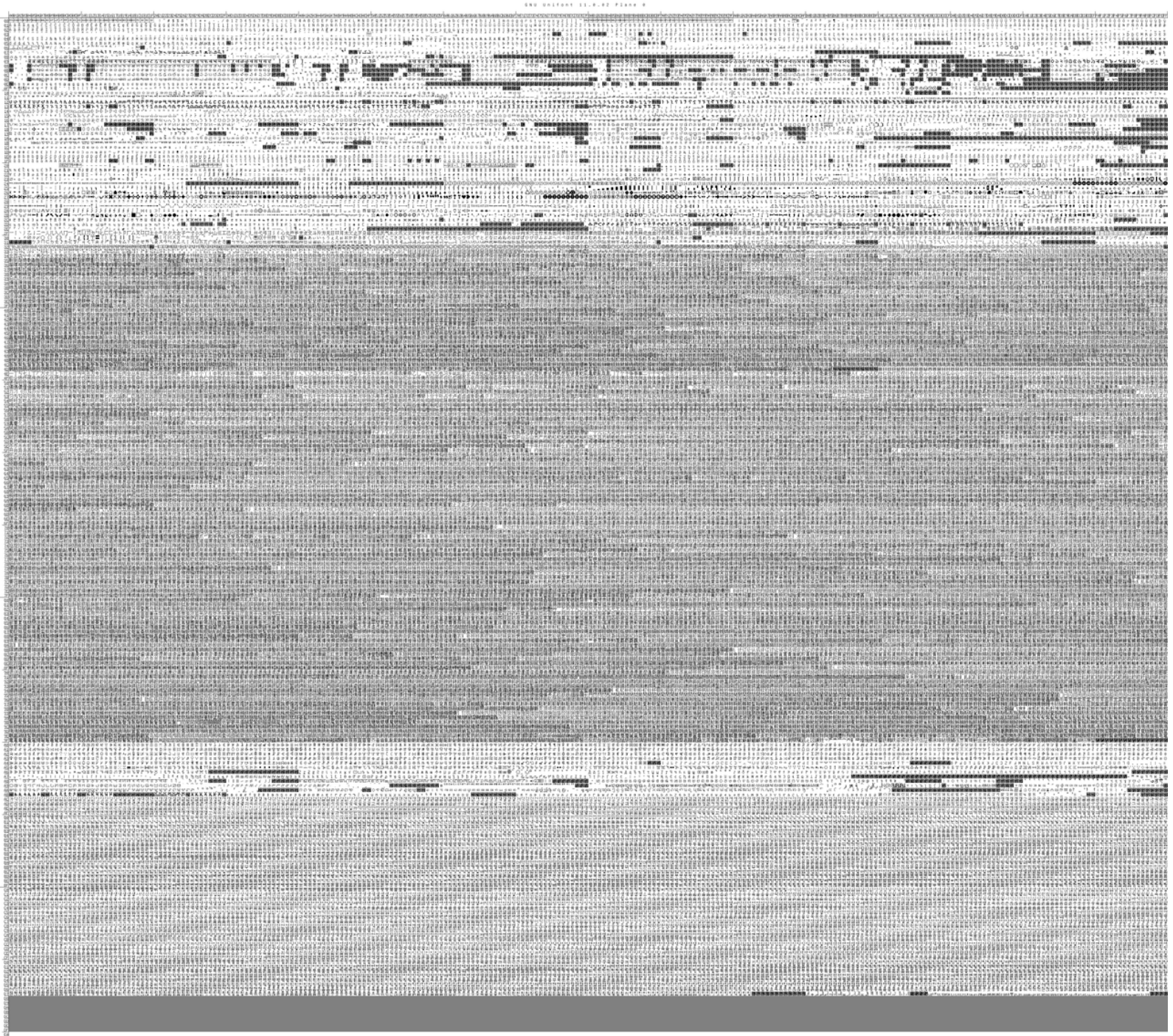


CHINESE TELEGRAPH CODE



INTERNATIONALIZATION

- For further examples see https://docs.oracle.com/javase/tutorial/i18n/intro/quick.html
- Internationalization refers to making programs that can take input and output that is tailored to different locations and languages. Historically encodings were used to map between telegraph signals and letters in a minimal way (often not considering case), here encoding refers to a mapping from characters used in language to 0s and 1s used in binary representation. As different languages contain a wide variety of letters and characters, for example Chinese, Arabic, English, etc, character sets are an important element of in the process of making software systems international or language/location independent.
- A character encoding can take various forms depending upon the number of characters it encodes. The number of characters encoded has a direct relationship to the length of each representation which typically is measured as the number of bytes. **Having more characters to encode essentially means needing lengthier binary representations.**



UNICODE

- It is not difficult to understand that while encoding is important, decoding is equally vital to make sense of the representations. **This is only possible in practice if a consistent or compatible encoding scheme is used widely.**
- Different encoding schemes developed in isolation and practiced in local geographies started to become challenging.
- This challenge gave rise to **a singular encoding standard called Unicode which has the capacity for every possible character in the world.** This includes the characters which are in use and even those which are defunct!
- We use hexadecimal as the base for code points in Unicode as there are 1,114,112 points, which is a pretty large number to communicate conveniently in decimal!
- **Therefore, how these code points are encoded into bits is left to specific encoding schemes within Unicode.** We will cover some of these encoding schemes in the sub-sections below.

UNICODE ENCODINGS

- UTF-32 is **an encoding scheme for Unicode that employs four bytes to represent every code point** defined by Unicode. Obviously, it is space inefficient to use four bytes for every character.
  - For example: "T" in "UTF-32" is "00000000 00000000 00000000 01010100" (the first 3 bytes are unnecessary)
- Variable length encoding: UTF-8 is **another encoding scheme for Unicode which employs a variable length of bytes to encode**. While it uses a single byte to encode characters generally, it can use a higher number of bytes if needed, thus saving space.
  - For example: "T" in "UTF-8" is "01010100"
  - But "語" in "UTF-8" is "11101000 10101010 10011110"

SUMMARY

- FileIO, Internationalization, and Charset encoding are important practical concepts
- In this weeks practical we look at some examples and demonstrate how topics from the lectures including Object Orientation and Exception Handling are able to be used in these problems