Regular expressions

- To replace certain parts of a string
- .replaceAll(".", "Y") this will replace all characters with a Y. '.' means all characters.
- '^' followed by the first few letters, will replace those letters if found only at the beginning. If that set of letters repeat in the middle, they will not be replaced.
- Set of letters followed by '$' will replace only hat set if found at the end.
- Replace any occurrence of a letter, put in square brackets "[abc]","Y") any occurrence of a or b or c with y
- [abc] [fj] a or b or c followed by either a f or j will be replaced
- .matches, the whole string needs to match.

System.*out*.println("harry".replaceAll("[Hh]arry", "Harry"));
The above will return Harry.

- To replace every letter besides a few
  - .replaceAll("[^ej]", "x")
    - In this case the carrot is inside square brackets, therefore it does not relate to the start. This will replace all letters with x besides e and j.
- Regex is case sensitive.

- A range of letters can be specified in the following manner [a-f3-8]
  - A to f and 3 to 8
  - To ignore case sensitivity, (?I)[a-f] can be used.
- Replace all numbers
  - [0-9] or "\\d"
- Replace all non-digits
  - "\\D"
- To remove white space
  - "\\s"
- To remove all non whitespace spaces
  - \\S
- To replace everything
  - \\w
- To surround each word with certain characters, tags etc
  - \\b

Quantifiers

- If you have a few letters together which are the same e.g. 5 e's in a row. Instead of writing 'eeeee'
  - E{5} can be written.
  - Or if you don't care how many e's you want to get rid off
    - e+
  - The above 2 methods are used when e is present.
  - If e is not present, you can use the * method.
    - abcDe*
      - If e is not present, it will still change the letters before it. So it checks if e is present tor not and still changes.
  - By adding {2,5} this represents same character between 2-5 times
    - abcDeee will get changed
    - abcDe will not as it's not in between.
  -