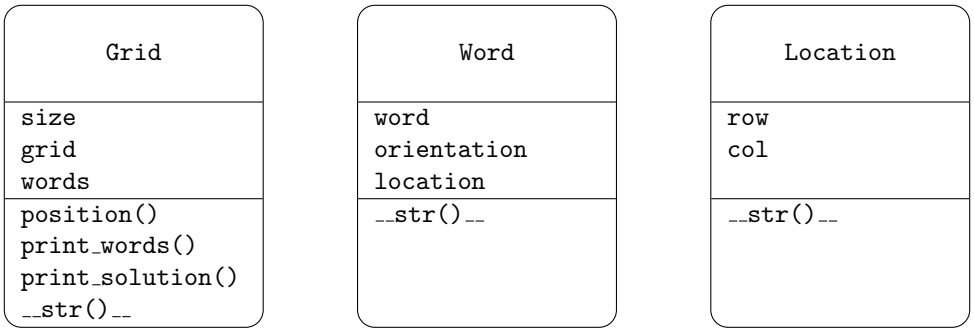
A word search is a puzzle which conssits of words that are hidden within a grid of letters. The words can be hidden in any orientation: horizontally (right-to-left OR left-to-right), vertically (top-to-bottom OR bottom-to-top), and diagonally (all four) as shown below for the words - `PANTHER ZEBRA CROCODILE SNAKE ALLIGATOR`

```
U C Y Z G A D D P E
W M E K A N S Z N L
K Q Q J Y L Y P W I
L Z S A A X A E O D
I A E M E N Q W F O
P D F B T A U K S C
G Q U H R K S B K O
U C E I H A S O S R
C R D I G J N P X C
H A L L I G A T O R
```

Specification

In this programming assignment (which is part I), you will only focus on the `Word` and `Location` classes. The Grid class will be the focus of the next programming assignment. Create the **Word and Location** classes based on the class diagrams:



For this assignment, the following are provided to you:

- animals.txt
- words.txt
- WordSearch.py: the main part of the program that makes use of the Location and Word classes.

Location Class

- The default values of row and col in the constructor should be 0.
- Define getters and setters for row and col. If negative values are specified through the setters, the values should be reset to 0.
- A __str__() method that returns a string representation of the Location in the format `(row,col)`.
- Test your Location class by creating different location objects.

Word Class

- Default value for orientation and location should be `None`.
- Getters and setters for the instance variables word, orientation, and location. Note that location should be an instance of the Location class.
- A __str__() method that returns a string representation of the Word in the format - word/orientation@location. For example: `ZEBRA/HR@(3,5)`.
- Define a list of strings named "`ORIENTATIONS`" as a class variable[1]. The list should contain -
  - HR: horizontally to the right (i.e., from left-to-right)
  - HL: horizontally to the left (i.e., from right-to-left)
  - VD: vertically down (i.e., from top-to-bottom)
  - VU: vertically up (i.e., from bottom-to-top)
  - DRD: diagonally to the right and down (i.e., from top-left to bottom-right)
  - DRU: diagonally to the right and up (i.e., from bottom-left to top-right)
  - DLD: diagonally to the left and down (i.e., from top-right to bottom-left)
  - DLU: diagonally to the left and up (i.e., from bottom-right to top-left)
- Test your Word class by creating different word objects.

Sample Outputs

Make sure all the related files/classes are in the same folder. Note that the output varies on every run, as the words are randomly chosen. For the number of words as 10 and grid size of 25, running WordSearch.py has the folllowing output:

```
CHEETAH/VU@(6,16)
PANTHER/HL@(7,7)
IGUANA/DRD@(17,13)
HYENA/HR@(15,14)
KANGAROO/DLD@(13,12)
FROG/DLD@(10,11)
ARMADILLO/DLD@(24,5)
EAGLE/HR@(21,2)
TURTLE/VU@(11,3)
CHAMAELEON/VU@(15,4)
```

[1] `https://www.tutorialspoint.com/python/python_classes_objects.htm`

Deliverable

- Submit both the Word.py and Location.py.

Rubric

Item	Points
Good coding style	2
Appropriate Comments & Header	4
Location class	3
Location class constructor	3
Location class getters	4
Location class setters	4
Word class	4
Word class constructor	4
Word class getters	4
Word class setters	4
Orientations properly defined	4
Total	40

FIN