# COSC 1336 Lab: 12

Relevant reading: Chapter 11

**Due: Dec. 11, 2:30 pm**

(Late date: Dec. 13, 2:30 pm)

40 Points

**Note that the late date for this lab is just two days after the due date, because of the end of the semester.**

**Problem 1. [20 points]** In a file called `employee.py`, write a class named `Employee` that holds the following data about an employee in attributes: name, ID number, department, and job title. The class should have a constructor and a `__str__` function. The class's attributes should be private.

Once you have written the class, write a `main` function that creates a list of `Employee` objects to hold the following data:

| Name | ID Number | Department | Job Title |
|---|---|---|---|
| Mark Jones | 39119 | IT | Programmer |
| Arielle Lopez | 21345 | IT | System Administrator |
| Susan Meyers | 47899 | Accounting | Vice President |
| Joy Rogers | 81774 | Manufacturing | Engineer |
| Sirin Safar | 52341 | Accounting | CPA |

The `main` function, after creating the list of employees, should allow the user to input a department and display all the employees that work in that department. A blank department indicates that the user is done making queries.

**Problem 2. [20 points]** For this problem you will create a simple spell check program. Download the `dictionary.txt` file and in the same directory where you save it, create a file called `spell_checker.py` and then do the following.

a. Write a class named `SpellChecker`, which has the following functions.

- A constructor, which receives file name which is the name of a dictionary file. The constructor should open the file, assume that the file contains a single word on each line and read the words into a list attribute called `__words`. Be sure to strip the newline character from the end of each word before adding it to the list. When you call the constructor, pass in the name of the dictionary file you downloaded. It has 80370 words in it, so if your constructor is correct, you should end up with the `__words` list having length 80370.
- A `spellcheck_word` function, which receives a word and returns a Boolean indicating whether the word matches one of the words in the `__words` attribute. Since the word may have uppercase letters in it, but the words in the dictionary file are all lowercase, you should convert the word to lowercase before looking for it in `__words`. You should definitely stop and test this function thoroughly before you continue. Depending on how you write it, this function can be very short.
- A `spellcheck_sentence` function which receives a sentence, removes any punctuation from the sentence, splits the sentence into words, and spellchecks each. For each word that is not found in the dictionary, the function should display a message such as `"The word "sntence" does not appear to be spelled correctly."` If all the words are found in the dictionary, the function should display a single message at the end, such as `"All words are spelled correctly."` Again, test this function before you continue.

b. Now write a `main` function which creates a `SpellChecker` object, and then allows the user to type in sentences and spellchecks each one. A blank sentence indicates that the user is finished entering sentences.

For this lab, you should submit the following files:

- `employee.py`
- `spell_checker.py`