# CSCI 1100 — Critical Computer Science 1

Homework 8

Classes & Algorithms

Reading: “What We Get Wrong About Closing the Racial Wealth Gap,” by William Darity Jr., Darrick Hamilton, Mark Paul, Alan Aja, Anne Price, Antonio Moore, and Caterina Chiopris

---

## OVERVIEW

This homework is worth 100 points total toward your overall homework grade and is due Thursday, April 25th, 2019 at 11:59:59 pm. The file should be submitted as:

```
hw8Part1.py
hw8Part2.py
README.txt
```

Part 1 and 2 of the homework are worth 40 points each, while the README is worth 20 points.

In this assignment, we’ll be focusing on simulating a company’s hiring, firing, and raising processes; we’ve been talking about data ontologies and how the decisions we make about *how* we decide to order and sort the world in our datasets can lead to biased decisions, even independently of any bad intentions on the part of the programmer. For your final assignment, you will be creating your own data ontology--determining what kinds of variables are important to consider when hiring, firing, and promoting employees. We might, for example, think about how database systems can reproduce marginalization against women and persons of color without collecting any data about race, gender, body type, or place of birth. While the assignment will give you some basic requirements for variables to be measured, you will also have to come up with some of your own.

The following parts must be done using classes, which will contain the data and algorithms needed for completion of the assignment.

## **PART 1 :** *Creating the Worker and Job classes*

For part one, you’ll be focusing on creating a Worker class and a Job class to generate 25 workers, assign them jobs, and output their characteristics in whatever way you deem appropriate.

The Worker class is what you should create first and foremost. This class should contain these 5 characteristics:

1. Education
2. Level of Prestige and Connections
3. Number Previous Jobs
4. Number of Current Jobs
5. Importance of previous position (were they management? Labor? maintenance?)

In addition to these 5 characteristics, you need to create 5 more of your own that you feel is important for employers to measure about their potential employees. If you were running your own company and were asked to design this software system, what other quantifiable variables would you want to measure? For ease, all of these characteristics should be easily measurable and quantifiable, echoing practices of job searches generally. Think about how to quantify these things and what happens when you're quantifying particular things like education, stress, and some of the characteristics you add. All of these variables will influence your algorithms and operations in the second part.

Part of the work here will be "justifying" the quantitative translations of each of your categories. For example, how can you quantify level of education? Do you use years in school? Do you add a "point" for every degree an applicant has earned?

In your README file, list all 10 characteristics that you have created, and justify them. For the 5 predetermined variables, how did you choose to quantify and measure them, and why did you choose to measure them that way? For the 5 custom variables, why did you choose them, how did you choose to quantify them, and why did you choose to measure them in that way?

To start, you should first create your basic class structure and files. After you can successfully create a Worker, you should randomly create 25 more. To randomize these statistics, you should import the **random** module and generate the needed numbers. Here's how it works:

---

```
import random

print("Printing random float number with two decimal places is ",
round(random.random(), 2))
```

```
print("Random float number between 2.5 and 22.5 with two decimal places is ",
round(random.uniform(2.5,22.5), 2))
```

---

This would output:

---

```
Printing random float number with two decimal places is  0.8

Random float number between 2.5 and 22.5 with two decimal places is  9.5
```

---

Once you've created your 25 Workers, you can store these in whatever structure makes sense (a dictionary, a matrix, etc.), but you must be able to recall the Workers and retrieve information on them.

Next is the creation of the Job class. This class should be seen as a partner to the Worker class; it informs the Worker, and the Worker informs it. This is crucial in considering the creation of your class.

Here's what the Job class should consist of:

1. Position
2. Salary
3. Productivity
4. Requirements

Each Worker will have a Job. There are three different types of Jobs (you should not create three Job classes; your class should change based on what position the Job is): Manager, Store Worker, Office Worker. The salary for each, respectively, is $60,000, $42,000, and $40,000. Title and Salary are straightforward variables, whereas Productivity and Requirements are more complicated.

Requirements is there to help you when assigning a Worker a Job or choosing who to hire. In order to do this, Requirements should include an algorithm that runs through specific checks that you decide is important for each job. You should weigh different characteristics more (i.e., perhaps a Worker having the Education to be a manager is more important than the Previous Number of Jobs). Based on what the Worker scores best in is what Job they should be assigned.

Productivity refers to how effective a worker is on the job. Like Requirements, Productivity relies on an algorithm that talks to the Worker class. How this is ranked and decided is totally up to you, but think about what you're deciding to rank and weigh more.

To properly complete part one, after creating 25 Workers, the Job class, and assigning them Jobs, you should print out each Worker, their information, and their Job information in a legible, formatted way.

For full points on this part, you will be graded on:

- A Worker Class
    - Inclusion of required characteristics
    - Creation of extra characteristics
    - Class Structure
- A Job Class
    - Inclusion of required characteristics
    - Algorithms for Productivity and Requirements
    - Class Structure
- Creation of 25 Workers
- Formatted output of each Worker and Job
- Commenting, structure, and code style

## PART 2 : *HiringManager, Hiring, and Firing*

For Part Two of this homework, you'll be creating a Hiring Manager class that serves as a translation class. There are two algorithms you'll be focusing on making: a Hiring algorithm, and a Firing algorithm. Like the first part of the homework, you'll have a freedom in deciding who to fire and who to hire.

Your program should build upon the previous part and start with the 25 randomly generated workers. From there, you should ask the user if they need to hire more workers, or fire more workers.

For the Hiring algorithm, your program should ask what number of Jobs per type (Manager, Shop Worker, Office Worker) needed. Then, you should create at least double the total workers needed. For example, if I want to hire 1 Manager, 5 Shop Workers, and no Offices Workers, your program should randomly create 12 Workers. From these Workers, your Hiring Manager class

should talk to your Job class (in particular, Requirements) to figure out which workers should be hired. The code should print out who was hired, and who was not.
In order to do this, you'll need to write some decision making behaviors. For example, are workers hired based on a combination of education and experience? Or one or the other? If a combination, how are they weighted--50% each? 40%-60%? Why?

For the Firing algorithm, your program should ask what number of Workers need to be fired. Your code should then go through the current Workers and figure out from their Worker and Job characteristics and how many of that position you have of who to fire. Productivity should be used in this algorithm to inform the decision. The code should output who was fired, and why.

In your README file, describe in words how your hiring and firing algorithm works, and why you chose to make it work that way. What types of workers does it privilege, and what types of workers might get left behind?

For full points on this part, you will be graded on:

- Reading input from the user and asking to Hire or Fire
- A HiringManager class
    - A functional Hiring algorithm
    - A functional Firing algorithm
    - Talking to other classes
    - Using Productivity and Requirements for Hiring and Firing
    - Class Structure
- Formatted and legible output of who is hired
- Formatted and legible output of who is fired
- Commenting, structure, and code style