# **Project 6:** Algorithms as Future Makers

*A modified version* [*of the assignment created*](https://ethicalcs.github.io/#future-makers) *by Evan Peck at Bucknell University*

1. Due Date & Time: **2nd December, 2022**
2. What to submit: Submit 1 zip file containing the 2 Java files (Applicants.java and TestApplicants.java) as described below by the deadline. You also need to include a Word/PDF File. Unlike the earlier projects, I want you to use this Word file and respond to the prompts at different places. The assignment aims to support you to make incremental progress (called “scaffold” in learning sciences).



*In this lab, we will create algorithms that decide who will get an interview*

We have seen that algorithms have the potential to assign value to people and make decisions based on those values. Here, we will experiment with that idea in another *very real* context that will likely impact your job prospects in the future.

### In this project, you’ll practice...

* Modeling a class to manage job applications [5 points]
* Making decisions on who to shortlist for an interview [7 points]
* Evaluating the fairness of your algorithm [3 points]

## Moogle’s Hiring Algorithm: Who gets an interview?

Imagine you are working for *Moogle*, a well-known tech company that receives tens of thousands of job applications from graduating seniors every year. Since the company receives too many job applications for the Human Resources Department (HR) to individually assess in a reasonable amount of time, you are asked to create a program that algorithmically analyzes applications and selects the ones most worth passing onto HR. **Your job is to build an algorithm that helps determine the order which job applicants will be called in for an interview.** [This is a real scenario playing out every day and will likely impact you as an applicant someday.](https://hiring.monster.com/employer-resources/recruiting-strategies/workforce-planning/hiring-algorithms/)

### Creating the Applicants Class and TestApplicants Class [5 points]

### Data fields [2 points]:

Your first task is to create a class called “Applicants” that will manage the job applications. Recall how we short-listed a set of data fields (states) of the House class earlier. You must have a similar list of data fields (along with the corresponding data types). The first one is an example for you to refer, ***fill in the others***. You don’t have to fill all, only those you think are important.

* String name; // to hold the name
* String email; // to hold the email
* int phoneNumber; // to hold the phone
* String major; // to hold the major
* int experienceYear; // to hold the experience year
* int [] grades; // to hold the grades
* int passYear; // to hold the pass year

**Applicant’s academic data:**

One of the data fields has to be the applicant’s academic data. To make it easier for their algorithms to process, *Moogle* designs their application forms to get some numerical data about their applicants’ education. Applicants must enter the grades they receive from their 6 core CS courses. So, effectively, **this will be an array containing integers of length 6**.

***Your task: Add the academic data as a data field in your class.***

To make things simpler, let’s think about the 6 courses that you will study at SFSU if you choose to take a CS major:

1. CSC 210: Introduction to Computer Programming
2. CSC 220: Data Structures
3. CSC 340: Programming Methodology
4. CSC 413: Software Development
5. CSC 415: Operating Systems
6. CSC 510: Analysis of Algorithms

For your convenience, assume that the grades will be automatically converted to an array that you can store and access in your array. For example, a student who received the following score (after converting from their grades) …

* **Intro to Computer Programming**: 90
* **Data Structures:** 95
* **Programming Methodology:** 80
* **Software Development:** 100
* **Operating Systems:** 75
* **Analysis of Algorithms:** 95

… would result in the following array:

|  |
| --- |
| [90, 95, 80, 100, 75, 95] |

You can assume the same order (i.e., index 0 is *always* Intro to Computer Programming and so on).

### Thinking About Constructors [1 point]

Now that you have written the data fields, think about the different constructors that you may want in your class. What are the values that you think are typically known when we want to create a new Applicant object? Write down the constructors. Make sure that you include a default constructor (i.e., a constructor without any parameter) and at least 2 different constructors with different parameters.

***Your task: Add at least 3 constructors, including a default constructor.***

*I think that in order to create an applicant object we need to have at least one of information from the data fields.*

*The 3 constructors are a default constructor and 2 others.*

*Of the other two, first one has name, email and phone number as parameters, and the second one has major, experience year and pass year as parameters.*

### Defining the Basic Behavior of the Class [2 points]

Remember, our discussion on data encapsulation and visibility modifiers (private, public, default). Now, let’s incorporate that in our Applicant class.

***Your task: Add visibility modifiers to the class data fields.***

*The visibility modifier I added are private as I did not want anyone to alter it.*

Along with the visibility modifier, you need to create methods that will allow other classes to access and modify the variables. As we have discussed, these are achieved through getters and setters. Remember, not all data fields may need getters and setters. You need to make a decision on what fields need it and what don’t.

***Your task: Add appropriate getters and setters for the data fields that you think other classes will need to access and/or modify.***

*Added getters and setters to all the data fields which I needed to access it in the test applicant class.*

### Adding Methods to Analyze the Application [7 points]

In your code, you will be writing a series of predefined methods which apply different criteria to applicants.

We are first going to write basic methods that Moogle has pre-established. These methods will filter candidates based on some rules that **Moogle thinks are important** for their purposes.

**Your goal: Create the following methods in your class** [4 points]

* averageAbove85 accepts applicants that have an average above 85
* noGradeBelow65 accepts applicants that have no grade below 65.   
   This includes all grades.

* atLeastFour85 accepts all applicants that have at least 4 grades above 85

(their non-CS GPA counts as a grade in this case)

* upperLevelAbove85 accepts applicants that have an average above 85 *in their upper-level CS courses*.

Upper-level courses are those with numbers beginning 3 or more. These are the last 4 elements in the array.

**Testing your Applicant Class:**

***Your goal: Create a test class called Test Applicants******to test your methods*** [1 point]

* Create the class in the same folder as the Applicants class (for most of us, it is the src folder)
* Write a main method in the test class
* Inside the main method, create a new object of the Applicant class type (remember, the constructors you had created)
* Populate the object with appropriate values (remember, the getters and setters you have created above)
* Individually try out each method and see if it works as expected.
  + Iterate until it works as expected
* Create different objects of the Applicants class to test different values and conditions
  + Note: testing only one condition is not enough

### 

### **Create your *own* filter [2 points]**

You’ve now written and seen 4 different selection filters, *it’s time to create your own.*

* Decide what you think would be a fairer way of filtering applications.   
  **You shouldn’t just use what we’ve created so far. Come up with something *better*.** Provide you decision and rationale for filtering applications (in English, not code).
  + What criteria(s) do you want to use to filter applications? (Complete this.)

*The criteria I wanted to use to filter applications is how many years of experience the applicants had.*

* + Why did you select your criteria? (Complete this.)

*I selected these criteria because I noticed that many companies nowadays prefer candidates who have at least 3 years of experience so that they can give good outcome to the company, as a result, enhancing the growth of company.*

* + So far, we have looked at “easy” ways to measure applicants’ abilities – just their grades. You may have identified other criteria above. It is also important to think how users will showcase these abilities so that they can be used in the algorithm. Not all the criteria that you have identified above can be filtered by a computing algorithm. Remember, **not everything that you can measure matters and not everything that matters can be measured**. How will an applicant showcase their abilities to meet all criteria that you have listed above? (Complete this.)

*The applicant would need to fulfill all the criteria to get selected. Thus, applicants not only need to maintain good grades but also have few years of experience in order to get shortlisted.*

## Before you Code, Assess the Needs of Your Users

While the list above was *your* take of important criteria to consider. There are **many** more potential criteria to consider if you want to create a **fair** algorithm that takes into account the diverse applicants. Tech companies need to ensure more diverse workforce.

**Group Work:** Reach out to 3—4 other students in the class. Discuss the various criteria that you all have come. Create a bullet-point list of all the criteria that came up in conversation. Discuss together to rank the criteria and select all that you think is important to consider. [2 points]

Note: you all may not agree in the criteria to include. Listen to the differing perspectives and evaluate the difference in thoughts. Note them down below.

* Factor 1:
* Factor 2:
* Factor 3:
* Factor 4:
* Factor 5:
* Factor 6:
* …

Now, you will write your filtering method.

**Your task:** But before writing your code, provide **at least 3** test cases for your method customFilter [2 points]

customFilter:

* An example: Our data will be an array that has the courses listed above along with three other factors: non-CS GPA, has previous volunteering experience, and number of years working in a related job. We will select a candidate if they have average of CS courses above 70, have non-CS GPA above 3.5, and have previously volunteered somewhere.

Thus to test, we will call customFilter([100, 90, 95, 80, 100, 75]) should return true.

* Test Case 1:
* Test Case 2:
* Test Case 3:

During your creation, keep a couple of things in mind:

* Use block comments to explain what data that you are collecting to filter the applicants.
* Use both block and inline comments to describe what was happening in the program.
* Choose variable names that clearly describe that data that they hold.
* Use spacing to group similar code.

After you write your code, test it with your test cases. Did it work? If not, analyze why your code is not working.

## Your Code Works… but is it fair?

**You should never deploy real code without checking your assumptions.**

Your test cases tested your *technical* assumptions, but not *your social* assumptions.

1. Find classmates either inside or outside of the class.
2. Run your code with them.
3. Get feedback on what worked and what didn’t?

**Your task:** In particular, you should reflect on...

1. **Which students who apply for jobs are most likely to benefit from your algorithm?**
2. **Which students who apply for jobs are most likely to be penalized by your algorithm?**

## Your Reflection: Think about the limitations of the algorithm

**(REPLACE THE FOLLOWING REFLECTION WITH YOUR OWN REFLECTIONS.)**

No doubt that I provided various filters to short list the candidates, but it is not necessary that the companies take only these criteria into consideration. There are various other factors that companies think are significant. For instance, an applicant could have significant contribution in co-curricular activities as well as might have certifications related to the job they are applying, as a result, companies would consider it and thereby would increase their probability of hiring. Also, I would like to add few filtering criteria in the future, to enhance my eligibility criteria thus providing a better result and filtering.

*If you’re interested…*

* [Can an algorithm hire better than a human?](https://www.nytimes.com/2015/06/26/upshot/can-an-algorithm-hire-better-than-a-human.html)

By Claire Cain Miller (The New York Times; you have free subscription via SFSU)

* [Amazon Scraps Secret AI recruiting tool that showed bias towards women](https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G)

By Jeffrey Dastin (Reuters)