	Carátula para entrega de prácticas
Facultad de Ingeniería	Ingeniería en Computación

Profesor(a):	
Asignatura:	Programación Orientada a Objetos
Grupo:	1
No de Práctica(s):	3
Integrante(s):	11700202-9
No. de Equipo de cómputo empleado:	
Semestre:	2024-1
Fecha de entrega:	22 de septiembre de 2023
Observaciones:	

CALIFICACIÓN: ______

Index.

Practice 3 - Props and general use classes.

Perfect matching problem.

1.- Introduction.

This report has the objective of explaining the third practice of the Object oriented programming subject, in which we must explore some props for enhancing our programs.

We must build an algorithm for solving the “Perfect matches problem” in Java language for Hospitals and Students.

1.1.- Hypothesis

We have to choose a set in which the elements can choose first the better options. As this can be proven, it can also be proven that every element in the two sets will get a match, so, it doesn’t matter which option we take if the first option of S_i is already chosen. So if the element S_i can’t take its first option, it will take any remaining option.

2.- Development

The steps to follow were:

- Read the data.
- Read the preferences, only storing the first option of each set.
- Create a matches array.
- If the option is available, match instantly the elements.
- Create a visited array and check the missing matches.
- Match them randomly until we fill the matches array with 1:1 pairs.

This solution uses only the array data structure and takes $O(n^2)$ complexity. But we have a problem:

This isn’t an optimal solution, because we want to guarantee that at least one of the sets will have their best options (At the moment they make a choice), so we have to pick greedily the best option each time that an element S_i has a turn to choose.

Investigating a little bit of algorithms that could solve this efficiently and with the property mentioned before, We found the Gale-Shapley algorithm, a greedy algorithm for making the perfect matches, that runs in $O(n^3)$ time in its worst case.

But for implementing the Gale-Shapley algorithm in an efficient way, we need to make constant queries on a String to String set. At this time, our friends, the hash function and hash tables, are joining the game.

We could implement this from scratch, but the Java API makes life easier, with the HashMap class, which extends the AbstractMap class, and implements a hash table with all the attributes and methods we need, such as: get, clear, contains, etc.

So the correct steps to follow were:

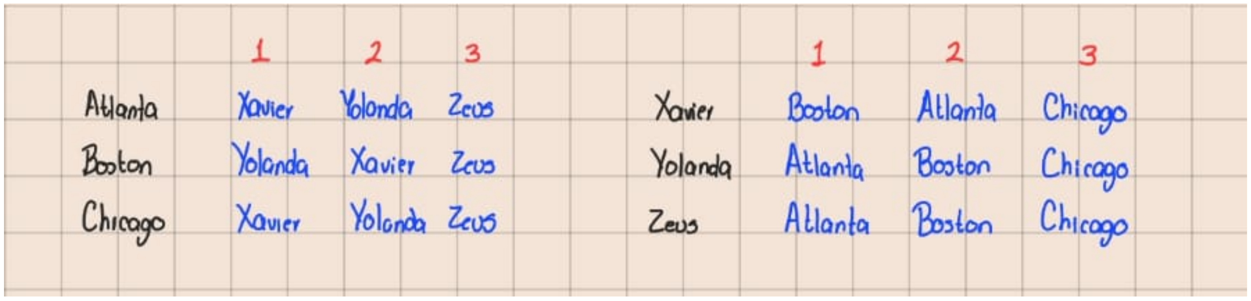
- Read the data.
- Read the preferences, storing them on a HashMap with String as key and array of Strings as the value
- Create a "matches" HashMap.
- If the option is available, match instantly the elements in the HashMap.
- Otherwise, we check the next options one by one until we get the next better option, and then match them.
- Print the matches HashMap.

Gale-Shapley algorithm pseudocode.

```
function GaleShapley(hospitals, students, matches, n):
  for each hospital in hospitals:
    key = hospital.name
    preferences = hospital.preferences
    for each student in preferences:
      student_preferences = students[student]
      for i from 0 to n - 1:
          if student_preferences[i] == key and student is not already matched and
hospital is not already filled:
```

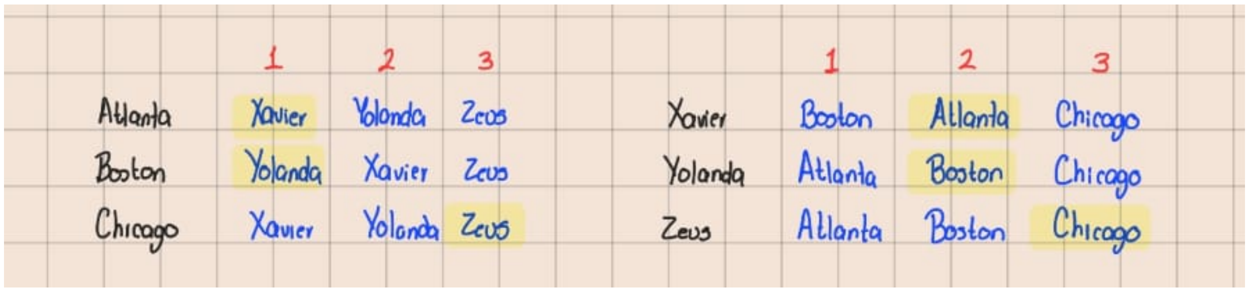
```
            matches[key] = student
            break
        # If the student is not available, check the next student in the preferences
         # until we find a student that the hospital wants and doesn't have a match yet
```

Desktop test with the given input.



	1	2	3
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1	2	3
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago



	1	2	3
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

	1	2	3
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

Conclusion: The hypothesis was incorrect, that drove us into rebuilding the algorithm in a complete way, investigating for a correct algorithm for obtaining the optimal solution.

I've liked this practice a lot because it challenged our algorithmic skills and teached some useful props that we will use in the future like what is a HashMap in java, what is an iterator, investigating the function of these classes and interfaces, and how they work.

References:

- [1] Oracle. (2023, July). Java API, Class HashMap [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

- [2] Oracle. (2023, July). Java API, Interface Iterator [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html
- [3] GeeksForGeeks contributors. (2023, February). Stable Marriage Problem [Online]. Available: https://www.geeksforgeeks.org/stable-marriage-problem/
- [4]Alexander Osipenko. (2019,May). Towards Data Science. Gale Shapley Algorithm simply explained [Online]. Available: https://towardsdatascience.com/gale-shapley-algorithm-simply-explained-caa344e643c2