

Employee Management System Final project

Thomas Hillstrand / Logan Foster



March 18, 2021

Everett Community College

# Overview

This project Is an employee management system where you can add employees to the system, edit or modify employees, remove employees, view all employees in the system, view individual employees or display organization specific Employees.

# Functionality

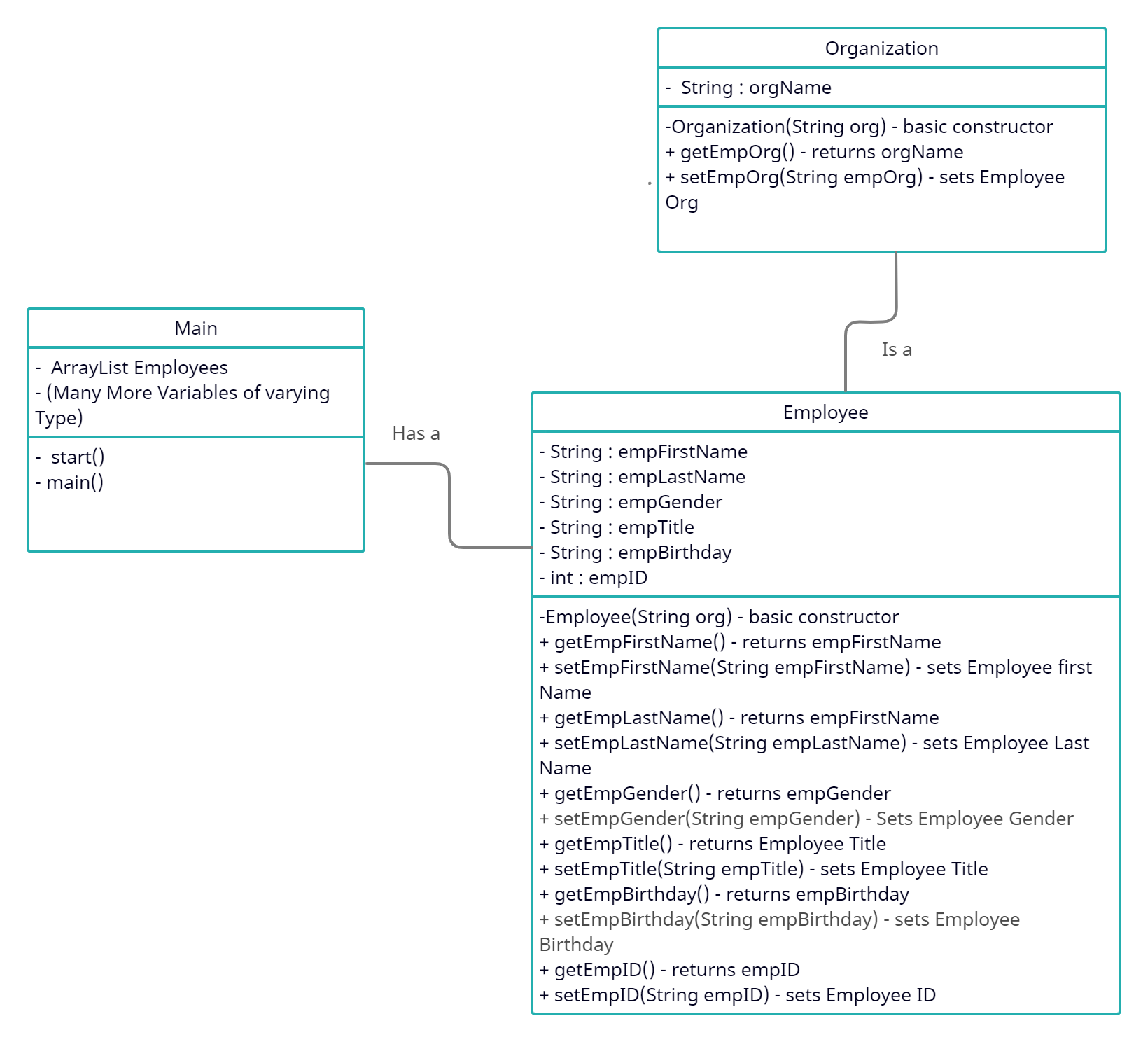
This project allows the user to Add a list or single Employees to a system where they can input specific pieces of information about each employee including, first name, last name, gender, job title, birthday, and organization. Each new employee is also assigned a random ID number. With the ability to add, remove or edit employees it makes an easy job of storing and displaying various employee’s information.

Being able to input and edit any employees makes it easy to fix any sort of mistakes that may occur during data entry.

# Implementation

The process that our group went about implementing our code was iteratively. We began from, quite literally the top of our list and worked our way down. We implemented the Add employee’s functionality and then used the built into String in the employees’ class to test them. Then the logical next step was to implement the view all employee’s functionality, once again using Print statements as well as the console to find our issues. A large bug we encountered along the way was that Java FX compiled initially and did not like to adjust its data displayed when we went to view it. (Ex: When Adding an employee and then attempting to view it, the program would say there are no employees because we would start with none). The workaround we found for this was to implement functionality with the buttons to allow them to update the respective scenes and text fields whilst using the program. The outcome is a program that is always attempting to update itself utilizing the users button presses, eliminating unnecessary loops that always run in the background. The process that our code uses to function is also rather simple. We declare an array list to use throughout the class and all our functions, pretty simply, operate around that. Each of our functions utilizes the array list to either pull the data from the employees or store it to be showcased. For our last section where we showcase all the employees within a specific organization, we utilize a secondary array list and store all the employees with a matching organization name that we take from the user. We then parse through the array List and add the employees that match to a new array List. If none are found, then we tell the user that. If some are found, exactly like how we display all the employees, we display the employees we found to the user in an easy to navigate way.

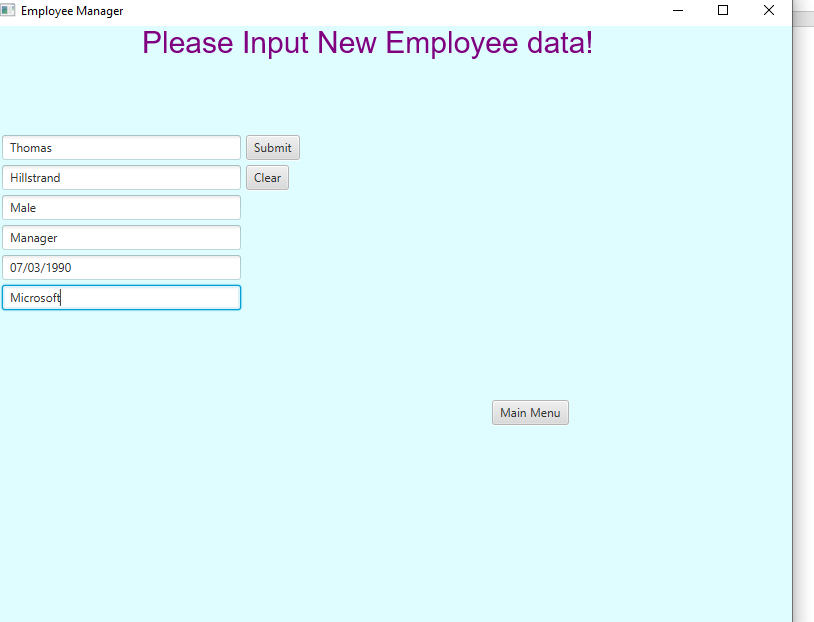
Flow Diagram

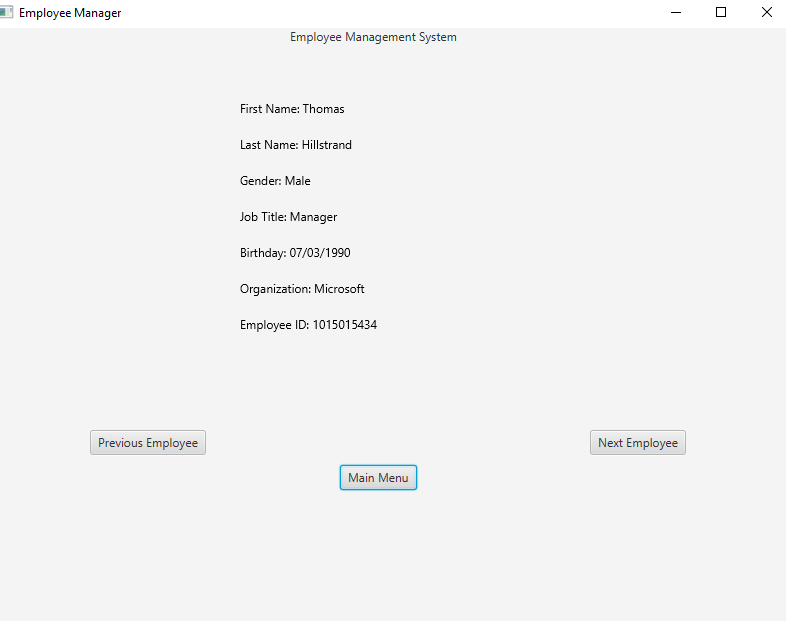
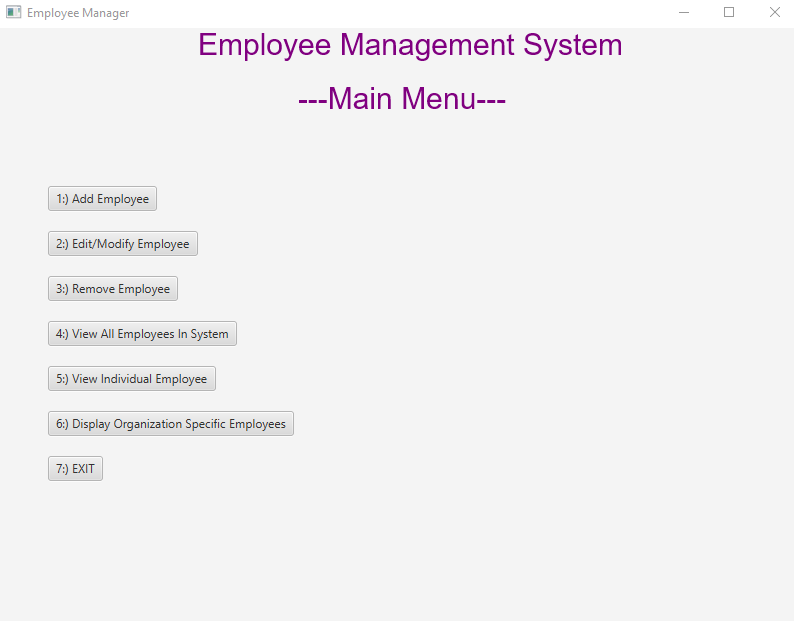


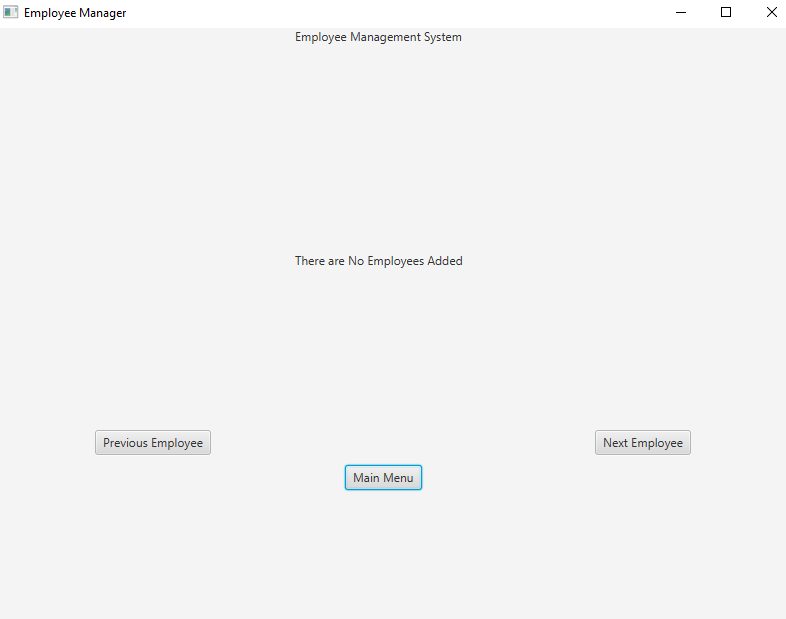
Results

The process that our group went about implementing our code was iteratively. We began from, quite literally the top of our list and worked our way down. We implemented the Add employee’s functionality and then used the built into String in the employees’ class to test them. Then the logical next step was to implement the view all employee’s functionality, once again using Print statements as well as the console to find our issues. A large bug we encountered along the way was that Java FX compiled initially and did not like to adjust its data displayed when we went to view it. (Ex: When Adding an employee and then attempting to view it, the program would say there are no employees because we would start with none). The workaround we found for this was to implement functionality with the buttons to allow them to update the respective scenes and text fields whilst using the program. The outcome is a program that is always attempting to update itself utilizing the users button presses, eliminating unnecessary loops that always run in the background. The process that our code uses to function is also rather simple. We declare an array list to use throughout the class and all of our functions, pretty simply, operate around that. Each of our functions utilizes the array list to either pull the data from the employees or store it to be showcased. For our last section where we showcase all the employees within a specific organization, we utilize a secondary array list and store all the employees with a matching organization name that we take from the user. We then parse through the array List and add the employees that match to a new array List. If none are found, then we tell the user that. If some are found, exactly like how we display all the employees, we display the employees we found to the user in an easy to navigate way.

During the process of making and debugging our code, we went through thorough testing of each of our functionalities making sure they worked. We did this by utilizing the process of Console Print statements, the print method in our employee class, and our code also does built-in checks. When our arrays or temporary employee variables are not in use, they are declared as null. Our code checks this to ensure that the null variables are not used improperly for a null pointer exception. You can directly see this if you search for an employee and our code cannot find one, we have a built-in function to state that we have not found your employee.







Read Me

For this program we created and easy to run Jar file that executes our program.

The first step to using this program is selecting:

option 1:) Add Employee. Once selected a new slide is shown where it is clearly labeled where you enter all the employee’s information. Once all the data is entered, you hit the submit button to save the employee to the system. You can add as many employees to the system as you would like you just need to hit option 1:) and enter in another employee.

Once one, or more, employees have been entered into the system; there is a variety of things you can do with the employee’s information to view or edit it.

Option 2:) Edit/Modify Employee allows the user to select an already added employee in the system using their first and last name. Make sure to type the name exactly as it was entered, or you will not be able to locate the employee in the system to edit them. Once the employee is found, you can edit any one of the previous data areas with new information and submit it.

Option 3:) Remove employee simply allows you to select an employee in the system via their first and last name and completely remove them from the system.

Option 4:) View all Employees in the system allows you to view the data of all employees that were entered in the system and using the previous and next buttons to navigate through them. Option 5:)View individual Employee allows the user to Search up a single employee via first and last name so that they will be able to view their information.

Option 6:) Display organization specific employees allows the user to look up all employees in a specific corporation by entering in the name of the corporation and submitting.

Option 7:) allows the user to quickly quit the entire program once the button is pushed.

Every one of the slides in our program, except for the main menu, has a button leading directly back to the main menu. We also made it so if a user does not submit the information, when they go back to add employee, the unsubmitted employee information will still be there unless deleted. For all the text boxes that require text entry we put a clear button beneath the submit button to quickly be able to clear all text fields.

Something to remember when using this program is that the submit button must be pushed when entering in new data or the data will not be saved.