**COMP 202 - Fall 2021 - Homework #1**
**Due date - 23:59 10/11/2021**

## HW Description

In this homework, you will implement a “ContactList”. ContactList will be a list of “Contact” elements. You will use **linked lists** for this implementation. Contact class and main class are already provided for you. You do not need to implement anything for those classes. They are straightforward classes, but before you start writing your code, make sure that you understand them correctly. You will implement several methods in the ContactList class. Then we will test your methods one by one, as you can see from the main class. If all of your methods are working fine you should see an output match from Github’s Autograding.

Methods that you will implement are addToEnd(), insertAfter(), insertBefore(), delete(), addToFav() and removeFromFav() . They are all explained below. Node class and displayList method are already provided for you. You can define additional helper methods in the ContactList class if you want.

There will be a favourite feature of our ContactList. Once contact is on our list, we will be able to add it to favourites. For adding a contact to favourites, you will basically change its position to the very beginning of the list. If the given contact is already in favourites you will not change its position. We will use addToEnd(), insertAfter() and insertBefore() methods to add new contacts to our list. No contact will be initialized as a favourite before adding it to the list. Therefore, you do not need to worry about the favourites situation before adding them to your lists. Similarly, for removing a contact from favourites, you will change its position to the very end of the list. If the given contact is not in favourites, you will not change its position.

You need to throw a NullPointerException when we try to add a null contact to the linked list. In other words, handle null inputs for addToEnd(), insertAfter() and insertBefore().

See below for more details about each method.

**addToEnd(Contact new_contact)**: Add a new contact to the end of the linked list. Notice that if the list is empty, this new contact will be your first contact.If **new_contact** is null, throw a NullPointerException with the message “Null contact entry denied!”.

**insertAfter(Contact prev_contact, Contact new_contact)** : Add a new contact after a given previous contact. If there is no such previous contact, do not do anything.If **new_contact** is null, throw a NullPointerException with the message “Null contact entry denied!”.

**insertBefore(Contact next_contact, Contact new_contact)** : Add a new contact before a given “next contact**"** .If there is no such next contact, do not do anything.If **new_contact** is null, throw a NullPointerException with the message “Null contact entry denied!”.

**delete(Contact del_contact)**: Delete a given contact from the ContactList. If there is no such contact, do not do anything.

**addToFav(Contact contact_to_fav)**: Change position of the given contact to the very beginning of the linked list. If the given contact is already in favourites do not change its position. Change the fav boolean of the contact to true. Do not add a new contact to the list with this method.

**removeFromFav(Contact contact_to_fav)**: Change position of the given contact to the very end of the linked list. If the given contact is not in favourites do not change its position. Change the fav boolean of the contact to false. Do not add a new contact to the list with this method.

**displayList()**: DO NOT change this method. It basically prints the list. It is important to have the same output format.

## Submission Details

Usage of github classroom and submission details:
1) Accept the invitation using this link (https://classroom.github.com/a/MOh5NvoK)
    a) If you have no github account, create one.
2) Choose your student ID from the list, it will be linked to your github account from now on.
3) A personal repository with the starter code will be created. You can directly edit the code on github's page, clone it to your local storage and edit it there, or use the online IDE VS Code. Ensure that your repositories are private.
4) Make sure your changes are committed and pushed to the repository.
5) Check if you have successfully passed the tests in the "Actions" tab of your repository. Note that these automatic tests do help you understand whether or not your solution works up to some level, but passing them does not guarantee that you will receive a full Grade.
6) To make sure your code is received and avoid any potential problems on github's side, submit a copy of java files on Blackboard as well. Submit all files as a single .zip file, named as: "ID_NAME_SURNAME.zip"

## Grading Criteria

You are going to receive 15%, for each 6 methods that you will implement . You are going to receive the remaining 10% from handling null inputs.

For your further questions about the homework send an email to: bbiner21@ku.edu.tr