CRITERION A

Description of the Scenario

The client is Felix Acosta, my father and regional director of Telcel. Due to the current Covid-19 Pandemic, He has been forced to start working from home, and just occasionally go to the office. It was evident that this has caused him to be unable to effectively communicate with his coworkers and assign them tasks. Due to my relationship with him, and the fact that this issue is really common due to the circumstances, I decided I wanted to develop a product for him. Therefore, I approached him to offer a program that can solve this issue, and he seemed interested. We programmed an interview so we could discuss the details of the program being developed. After the interview, The client was really interested in having this software since it would allow him to have a better management of the tasks he assigns as well as sending mails automatically. Now, The details were clear. It needs to be able to assign tasks to coworkers, but also be able to assign individual deadlines. The software needs to also check the deadlines for those tasks and send mails to inform both the user and the coworker that the due date has arrived.

Rationale for the Proposed Product

The product searches to improve the tedious labor of sending emails to every coworker, and having to organize if every one of them has finished their assigned work. As an online platform is a bit out of my knowledge, this product would instead use an email the user wants to send the emails from, and would automatically send emails to the coworker's mail to remind them of project deadlines, and asking them for advances. In case the project is not delivered or advancements are not received, an alarm in the form of an email will be sent to both emails. This way, all of the client's objectives will be accomplished with a program that, by being mainly local, will protect the data and fulfill his expectations, as shown in questions 2, 3, and 4 of the Interview in appendix A. I'll be using Java in IntelliJ to achieve this goal. I chose Java because it is a programming language that I know well and that due to its nature, can be compiled and run on any computer regardless of the operating system or the model of the computer. I'll use IntelliJ because it is a programming environment I feel comfortable with. This solution best fits my client since by the program being mainly local, the data is not compromised, as opposed to other online solutions. The program will not require personal info rather than your email and password. The system also suits him since it will be able to run on any computer system. As for me, due to its local nature, this solution is easier than an online platform and more cost-effective. The only other thing to consider is that the client must have an email address in order to use the program as well as the coworker.

Success Criteria:

- The software is able to assign multiple tasks to coworkers
- The software is able to assign individual deadlines for each of the tasks
- The software sends an email to notify the coworker of the task assigned

- The user is able to ask for advancements on the project, which the program will send an email to ask the coworker for advancements
- In case a deadline is up or there is no answer at a certain time after asking for the advancement, an alarm in the form of an email can be sent to both the user's and the coworker's email
- All the data, such as the content of emails, emails themselves, or name of the coworkers or the user are strictly private and secured
- The software is user-friendly

(The client agreed to this success criterion as shown in question 5 in the interview on Appendix A)

Word Count: 499 words

CRITERION B

Plan:

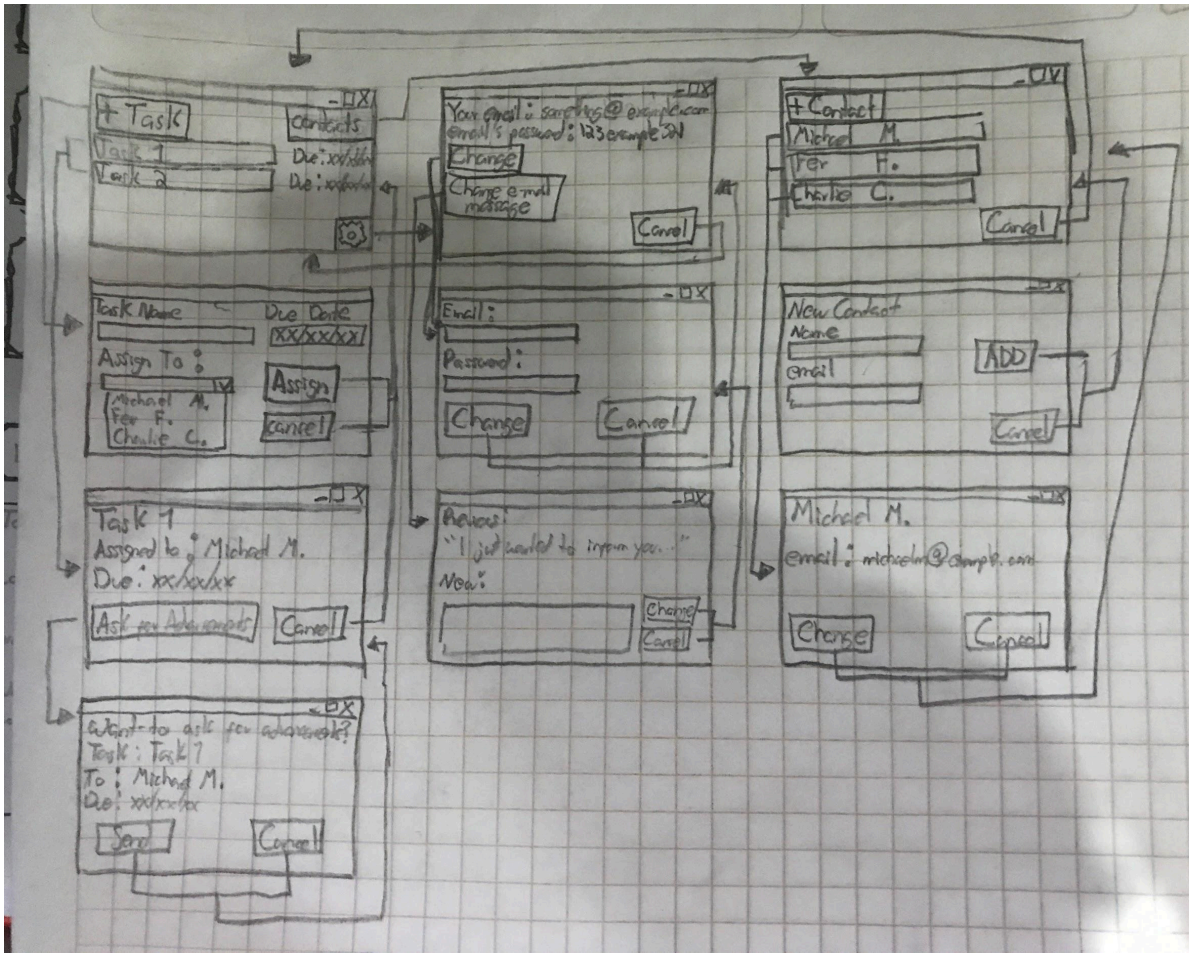


Figure 1: Planned Visual Design

Design:

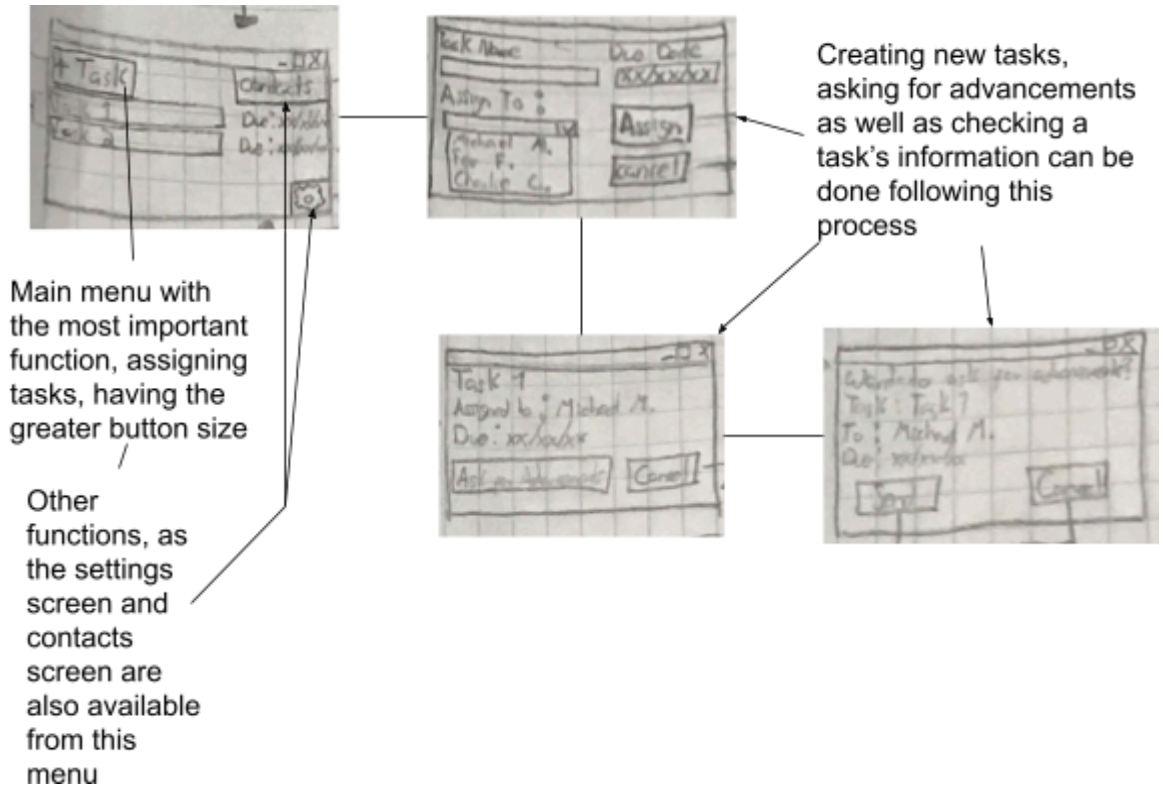


Figure 2: Task-related screens and process

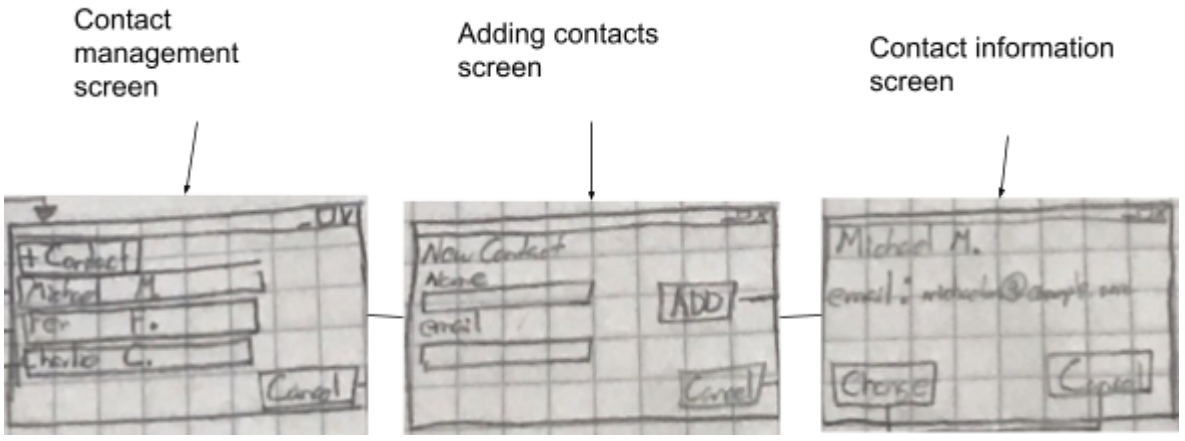


Figure 3: Contact Management, Adding, and Information screen designs

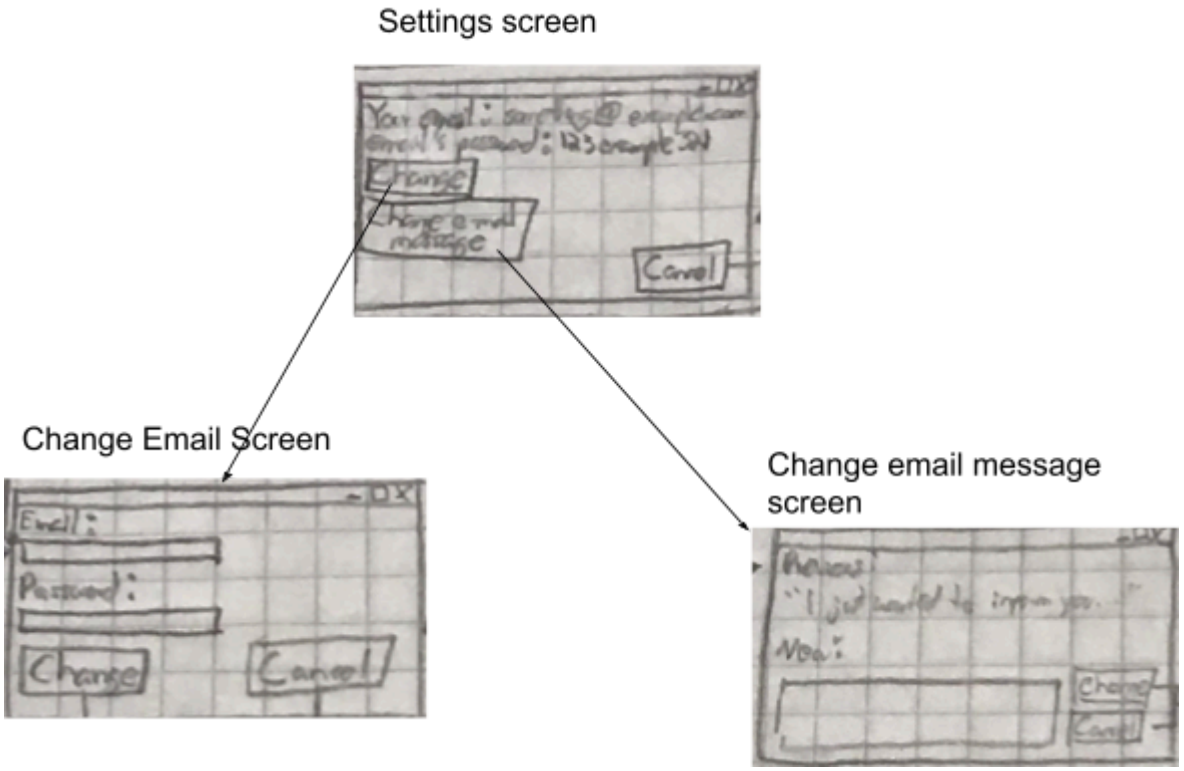


Figure 4: Settings, Change Email, and Change Email Message screen designs

Develop:

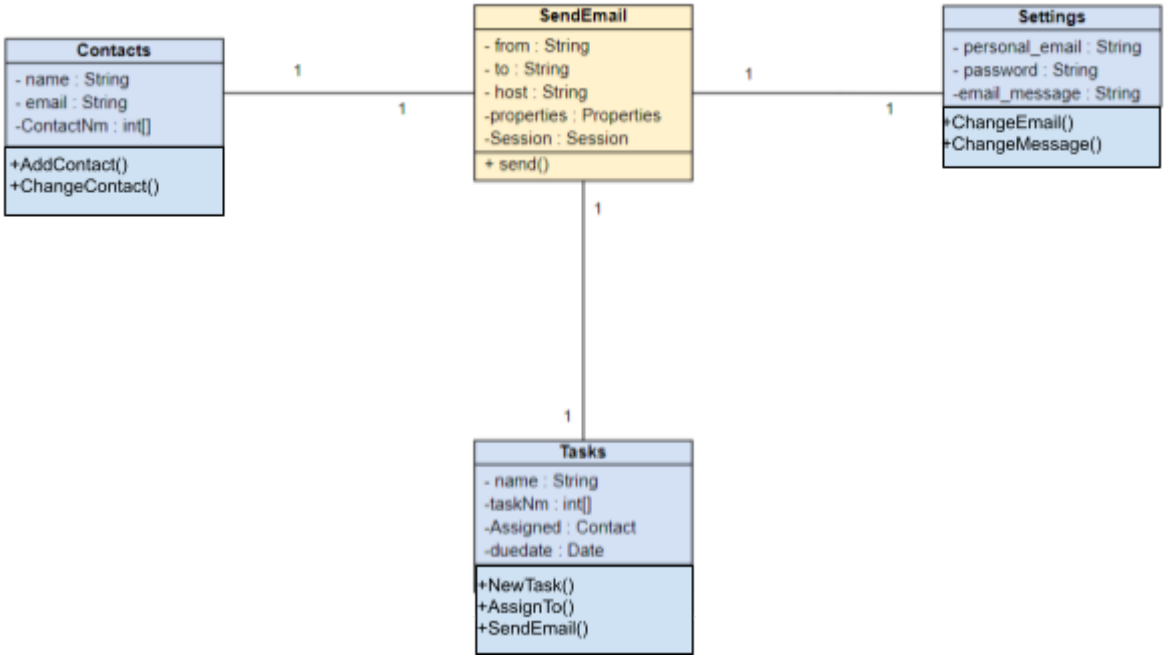


Figure 5: UML Diagram of the class SendEmail. All data managed here will be gathered from user input and stored locally

It was proposed that the program would be divided in three main parts, the task-related screens, the contact-related screens and the settings-related screens, the client approved. Tasks are the main focus of this program, as shown in Criterion A, the main menu is where you can see your tasks, with big buttons in the corners to either add a task or to manage your contacts, as well as a simple cancel button in the bottom-right corner of almost every screen except for the main one, where instead the setting button is located there. In this way, the design itself divides the program into these three main areas from the very beginning, while still allowing the user to access their tasks from the main menu. The design is simple to accommodate the client's desires and small to avoid occupying too much space of the user's monitor. Since the client is normally very busy with multiple programs open, this design helps him access either of the main parts of the program or his tasks really fast while allowing space for the client to see other programs he might have open.

Flowcharts of program procedures:

Below, one program procedure for each of the main elements of this program:

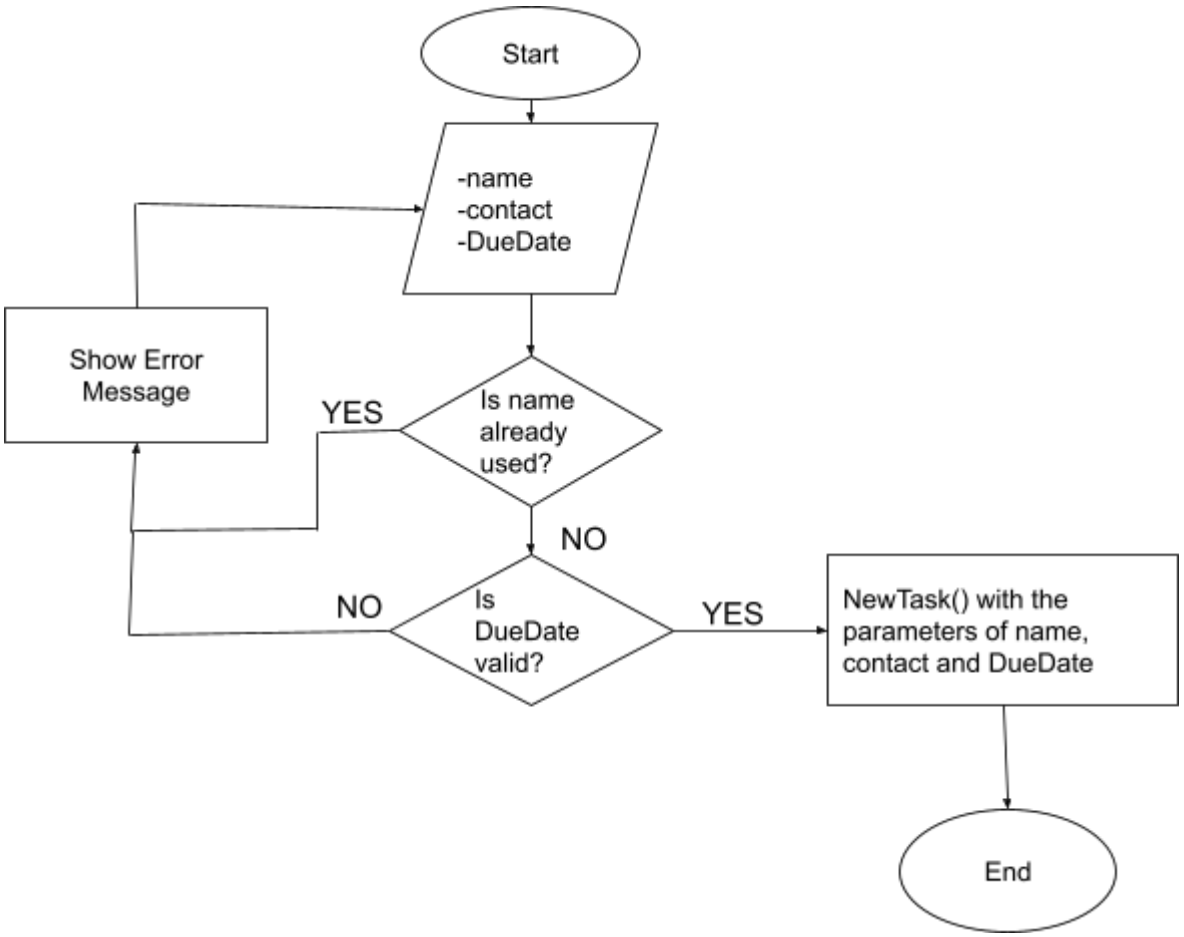


Figure 6: Create a New Task Procedure

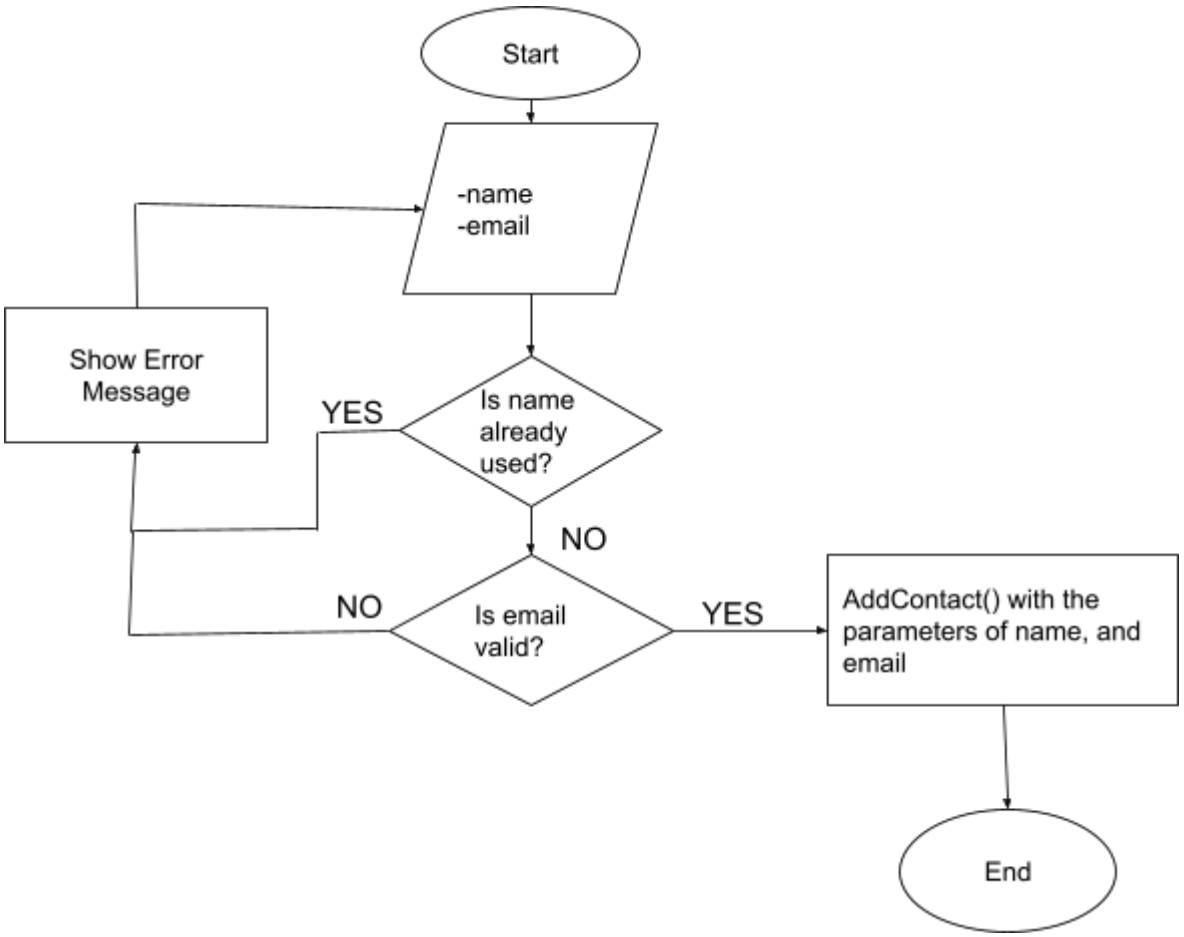


Figure 7: Add a Contact Procedure

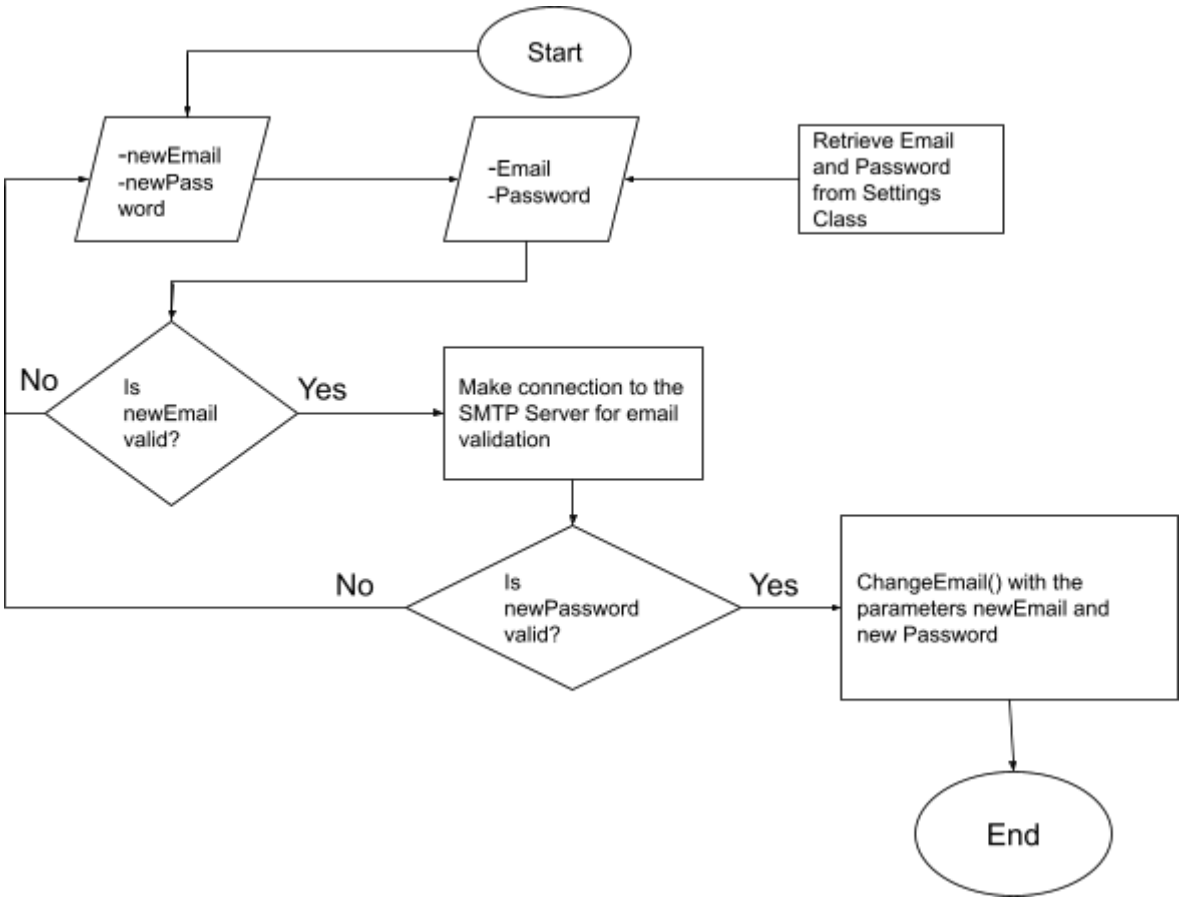


Figure 8: Change Email Procedure

Test Plan

Description	Success Criterion Tested	Input	Expected Outcome	Actual Outcome*	Pass/Fail *
Creating a New Task (Correct)	Create Tasks	name: Test Task Contact: Michael M. DueDate: 25/12/20	A Task will be created, sending an email notifying Michael M. with a due date set to December 25th	First 4 attempts: the email wont send unless it has an actual email address	After 4 retries: Pass
Creating a New Task (Already used name)	Create Tasks	name: Test Task Contact: Michael M. DueDate:	Error message that tells the user that the name has already been used in another task	The loop overwrites itself and creates an empty task	After 7 retries: Fail

		25/12/20		with the repeated name. For further development this could be fixed.	
Creating a New Task (Invalid Due Date)	Create Tasks	name: Test Task Contact: Michael M. DueDate:47/14/11	Error message asks the user for a valid date	An error message appears indicating the user to enter a valid date	Pass
Adding a Contact (Correct)	Assign Tasks	name: Michael M. email: michaelm@gmail.com	A contact inthe Contact class will be created with the parameters name and email	A Contact object is created and appears in the menu.	Pass
Adding a Contact (Invalid email)	Assign Tasks	name: Michael M. email: faketest	An error message asks the user to give a valid email	The system does not check if the mail is valid, but if its not either the task itself is not assigned to a contact or the user receives an email indicating that that email does not exist.	Pass (although future develop ment could enhance this feature)
Change Password (Correct)	Send Verification Emails	newEmail:test@cs.com newPassword: 123	The email and password are now equal to newEmail and newPassword respectively	The email and password change in both variables and the settings save file.	Pass
Change Password (Invalid Email)	Send Verification Emails	newEmail: notanemail newPassword: 123	Error message asks the user to give a valid email address	The system does not validate the mail and when sending the emails an	Fail (future develop ment could esure

				error appears	email validation and inform the user if the mail is sent successfully.)
Change Password (Invalid Password)	Send verification Emails	newEmail:test@cs.com newPassword: 321	Error message asks the user to give a valid password	The system does not validate the password and when sending the emails an error appears	Fail (future development could esure password validation and inform the user if the mail is sent successfully)

*Tests done after the development of the product.

Record of tasks:

Task Number	Planned action	Planned outcome	Time estimate	Target completion date	Criterion
1	Have interview with my client	Have an idea on what the client wants for a product and what criteria could be used to consider this program successful	1hr	15/10/20	A
2	Plan on the tools needed for the development of the product	Think and analyze the tools I might need to	30min	15/10/20	A

		develop the product as well as to fulfill the client's expectations.			
3	Define Criteria for success	Have a clear and definite list of success criteria my product must achieve	15min	16/10/20	A
4	Design the layout of my product	Have a plan and diagram on the design of the program	2h	23/10/20	B
5	Create an UML Diagram of the SendEmail class	Have an UML Diagram explaining the main components of my program as well as the relationship with the SendEmail class	1h	24/10/20	B
6	Create flow charts of the main procedures of the program	Have flow charts that explain the procedures of Changing email, creating a task and creating a contact	3h	30/10/20	B
7	Create a test table	Create a test table where I show the different tests that need to be taken on the main procedures of the program	2h	2/11/20	B
8	Investigate on how to make a GUI in Java	Learn how to make a basic GUI from scratch in	3h	10/11/20	C

		Java, that uses buttons, text fields and labels.			
9	Make the main GUI of the product	Have a working and basic GUI with buttons and labels that serves as the basis of my product	2h	15/11/20	C
10	Make the Settings GUI	Have a working Settings page that can be accessed through the main GUI	1h	20/11/20	C
11	Make the Contacts GUI	Have a working GUI that can be accessed through the main GUI where you can manage your contacts	1h 30m	25/11/20	C
12	Make the new task menu and GUI	Have the new task menu (accessible through the main menu) where you can create your tasks.	2h	30/11/20	C
13	Make the new contact menu and both task and contact info GUI	Have three menus, two where the user can see info both of their tasks and contacts, as well as a menu for creating new contacts	2h	15/12/20	C
14	Make the change email, and	Have a menu where the user	3h	30/12/20	C

	change email message menus and GUI	can change both their email and their password, as well as a menu for changing the content of the email.			
15	Make the send email menu (ask for advancements feature)	Have a menu where the user can ask for advancements on any task assigned to someone. Accessed from the task info GUI	1h	10/1/21	C
16	Program the Contact feature	Have the contact feature developed and working with the adequate menus, where the user can manage, create and edit contacts.	3h	15/1/21	C
17	Program the task feature	Have the task feature ready and running in the appropriate menus. The user is able to create, manage and assign tasks to contacts.	2h	20/1/21	C
18	Research a way to save data in a java program.	Research how to save data of variables and be able to load it every time the program is opened	2h	23/1/21	C
19	Program the save and load methods.	Have the code be able to save all the contact, task, and	3h	25/1/21	C

		settings data in local files through the use of RandomAccessFiles.			
20	Research how to send automated emails	Research a way to through java code, be able to send an email.	2h	27/1/21	C
21	Download the JavaMail API and the JAF	Download and install both the JavaMail API and the JAF to be able to use the code i researched	1h 30m	28/1/21	C
22	Program the sendMessage method	Have the code be able to send emails through my program	2h	28/1/21	C
23	Program the ask for advancements feature	The user is able to ask for advancements in any task by sending an automated email.	3h	28/1/21	C
24	Program the alarm emails	The code sends an email when a task is due or when it is newly assigned.	2h	29/1/21	C
25	Check the program for any bugs or errors	Test the program to make sure it doesn't have any bugs or errors in the code	1h	30/1/21	C
26	Record and show the product	Record the product functioning and explaining all	1h 30m	30/1/21	D

		of its features. As well as showing this completed version to the client			
27	Evaluate the product	Evaluate the points that the program got from the success criteria, and the reasons for not achieving certain criteria. As well as certain	2h	5/2/21	E
28	Get Client's feedback	Get and analyze the client's feedback that I received on the final version of the program.	2h	6/2/21	E
29	Evaluate future minor and major improvements.	Come up with several improvements that in future development could make the program fulfill better the client's needs.	1h	6/221	E

Word count: 205 words

CRITERION C

Classes Used:

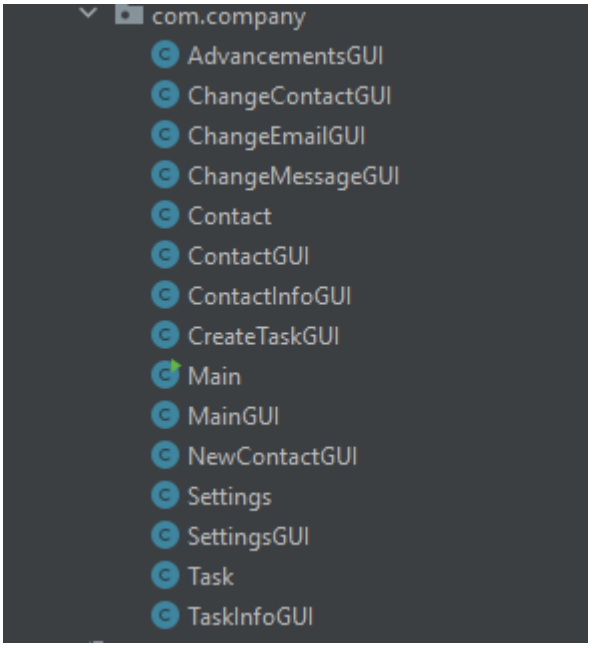


Figure 9: All Classes Used

Techniques Used:

- Method Overloading
- Method Overriding
- Nested If
- Implementing ActionListener to each GUI class
- Use of Array Lists for tasks and contacts
- use of access modifiers and static modifiers
- Use of the gmail SMTP server
- RandomAccessFiles: settings.txt, task.txt, contact.txt

Main menu:

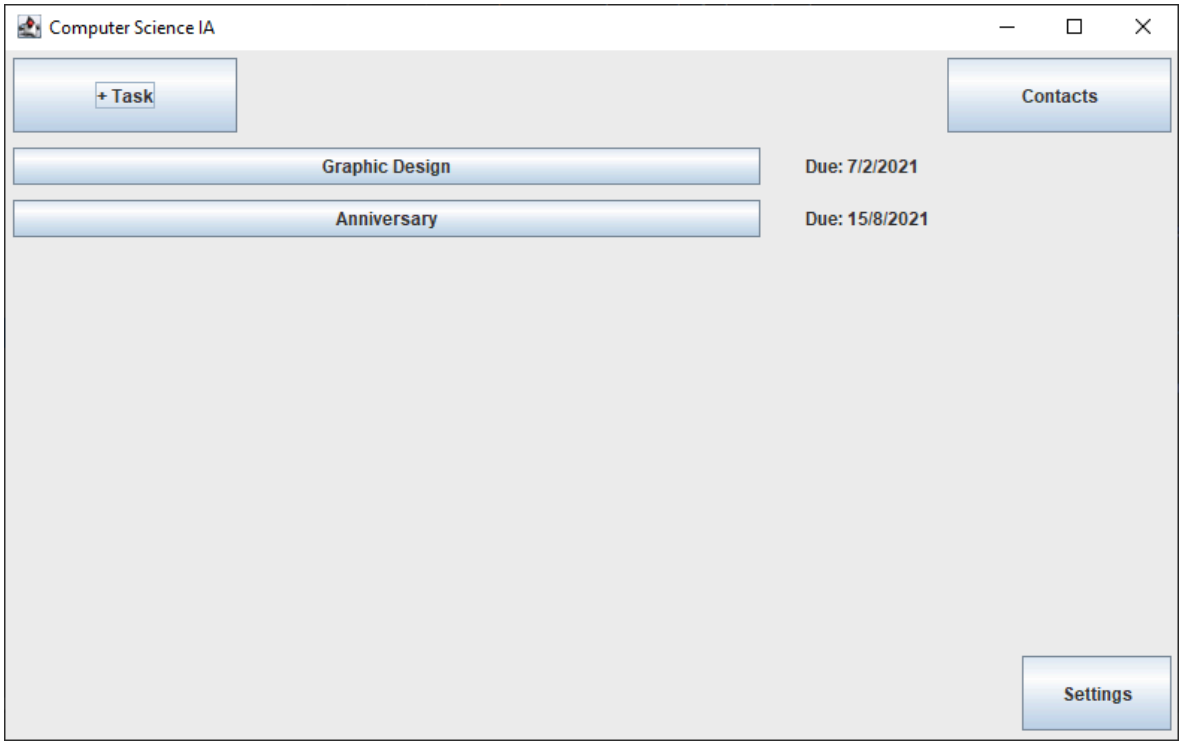


Figure 10: Main Menu screen. MainGUI class object

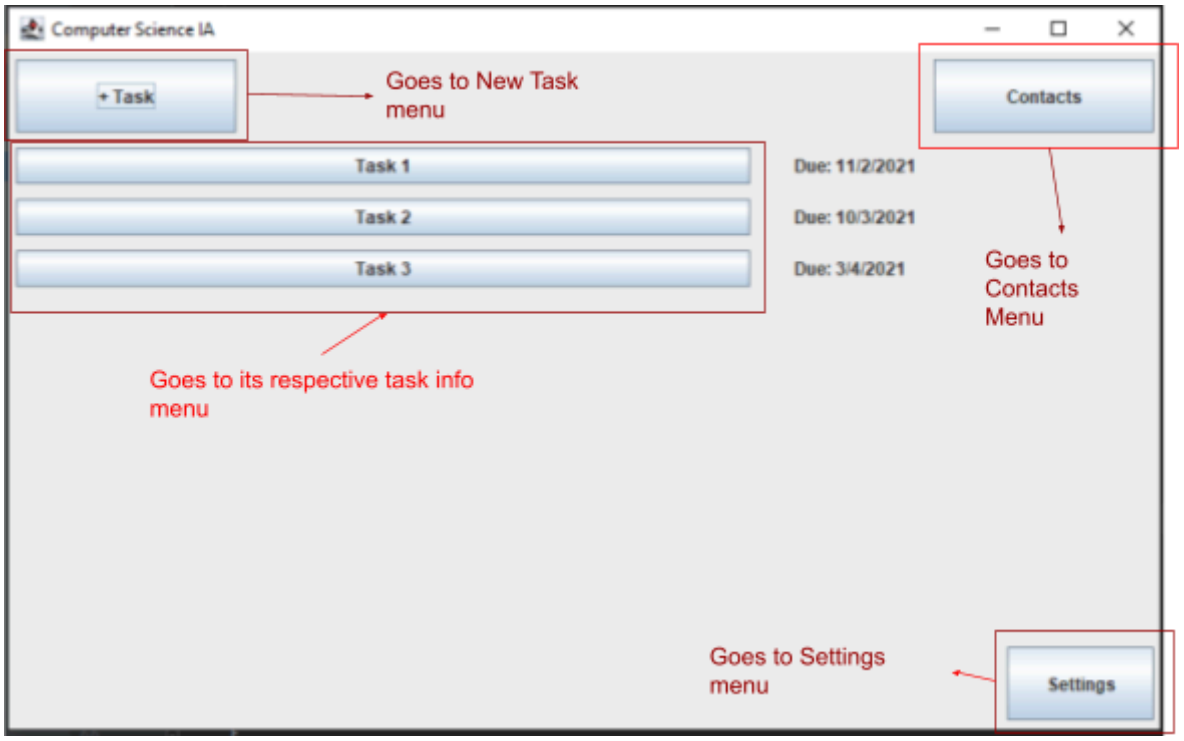


Figure 11: Main menu explained

For the interface of this program, I used the java library “swing”. This was used mainly to create the labels and buttons as well as the text fields. I also used the abstract window toolkit from java in order to make the buttons function properly.

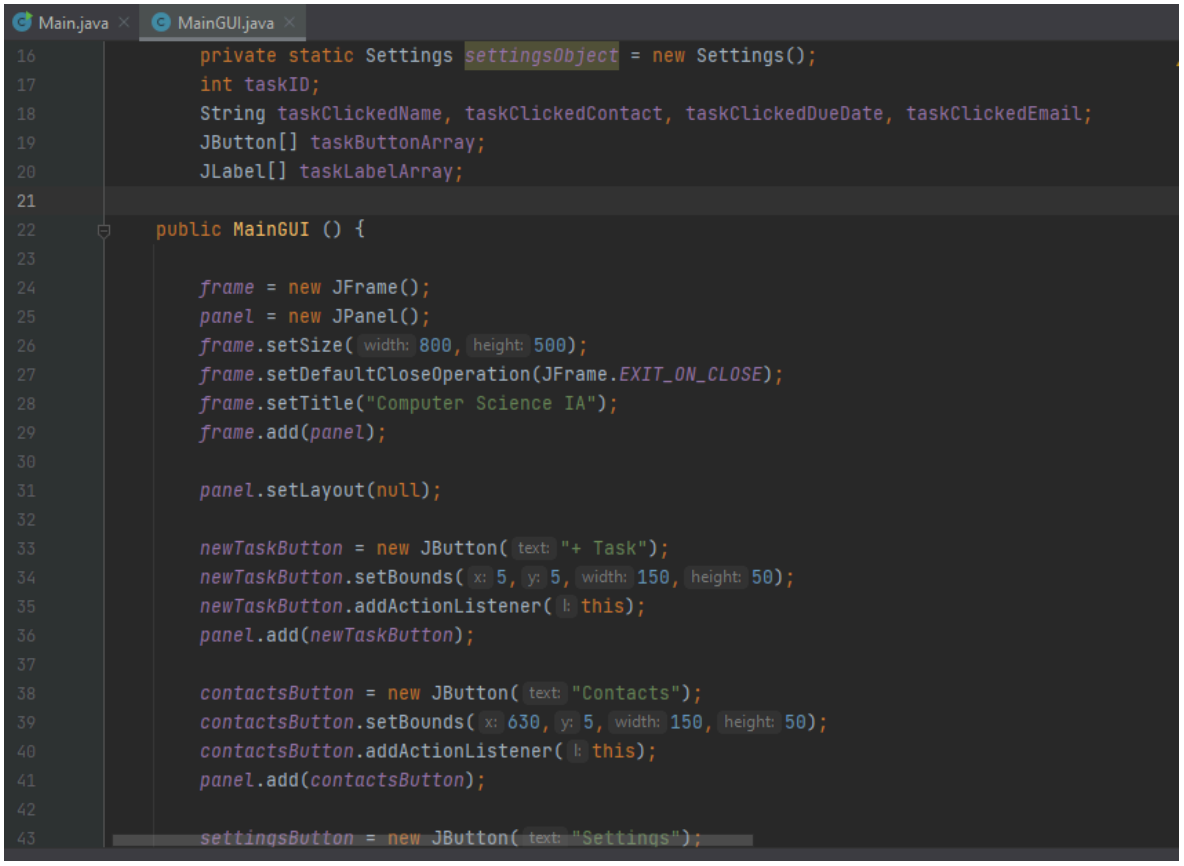


Figure 12: MainGUI code, demonstrating the use of the *swing* library

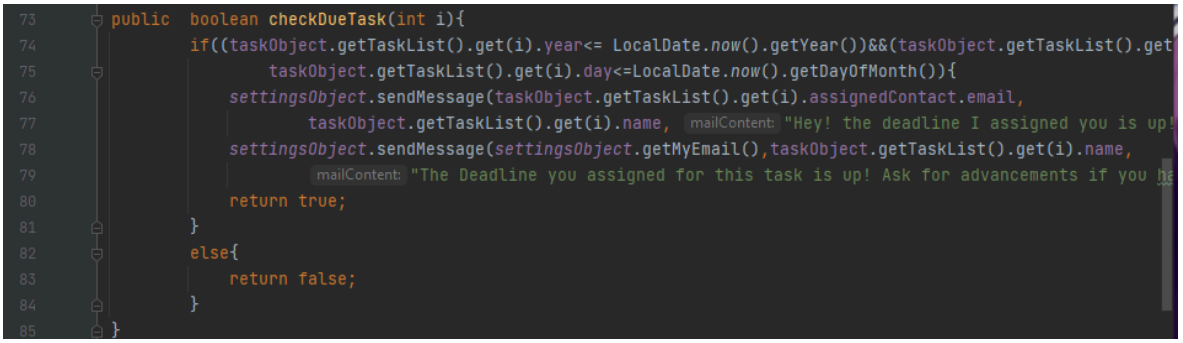
As shown on Figure 12, each element on the window corresponds to a button, label or textField object given by the *swing* library. First, the frame and panel are instantiated and given its values, as well as the default action for when closing the window, in this case, exiting the program. Then, each object is instantiated, given its location and size, as well as changing font, adding an ActionListener and then added to the panel to fit the design shown on Criterion B. this method is always repeated for each GUI object.

Each menu is a separate object and class. This decision was made due to the fact that each menu window, while still retaining similar factors, has different elements and different distributions of these elements.

```
48
49          taskButtonArray = new JButton[10];
50          taskLabelArray = new JLabel[10];
51
52          for(int i = 0; i < taskObject.getTaskList().size(); i++){
53              taskButtonArray[i] = new JButton(taskObject.getTaskList().get(i).name);
54              taskButtonArray[i].setBounds( x: 5, y: 65+(35*i), width: 500, height: 25);
55              taskButtonArray[i].addActionListener( l: this);
56
57              taskLabelArray[i] = new JLabel( text: "Due: " + taskObject.getDueDate(taskObject.getTaskList().get(
58              taskLabelArray[i].setBounds( x: 535, y: 65+(35*i), width: 100, height: 25);
59
60              if(checkDueTask(i)){
61                  taskLabelArray[i].setForeground(Color.RED);
62              }
63
64              panel.add(taskButtonArray[i]);
65              panel.add(taskLabelArray[i]);
66
67          }
68
69          frame.setVisible(true);
70
71      }
```

Figure 13: Example of a for loop to create a button for each task

For loop uses the ArrayList in the Task class to assign to each element of the button array a JButton object corresponding to a task in the ArrayList. The same is done for the “Due:" labels and uses the method getDueDate to get the date in a String variable used by the label. it also checks if any of the tasks in the ArrayList are due using the method checkDueTask. there can only be up to 10 tasks at the same time, due to simplicity. This was consulted with the client. He agreed, claiming he might not need more than 10 tasks at the same time.



```
73      public  boolean checkDueTask(int i){
74              if((taskObject.getTaskList().get(i).year<= LocalDate.now().getYear())&&(taskObject.getTaskList().get
75                      taskObject.getTaskList().get(i).day<=LocalDate.now().getDayOfMonth()){
76                  settingsObject.sendMessage(taskObject.getTaskList().get(i).assignedContact.email,
77                          taskObject.getTaskList().get(i).name,  mailContent: "Hey! the deadline I assigned you is up!
78                  settingsObject.sendMessage(settingsObject.getMyEmail(),taskObject.getTaskList().get(i).name,
79                          mailContent: "The Deadline you assigned for this task is up! Ask for advancements if you ha
80                  return true;
81              }
82              else{
83                  return false;
84              }
85      }
```

Figure 14: checkDueTask method

This method compares the day, month and year variables of the task with the system’s local date using the *time.LocalDate* library. If the due date has already passed, the method calls sendMessage() from the Settings object.

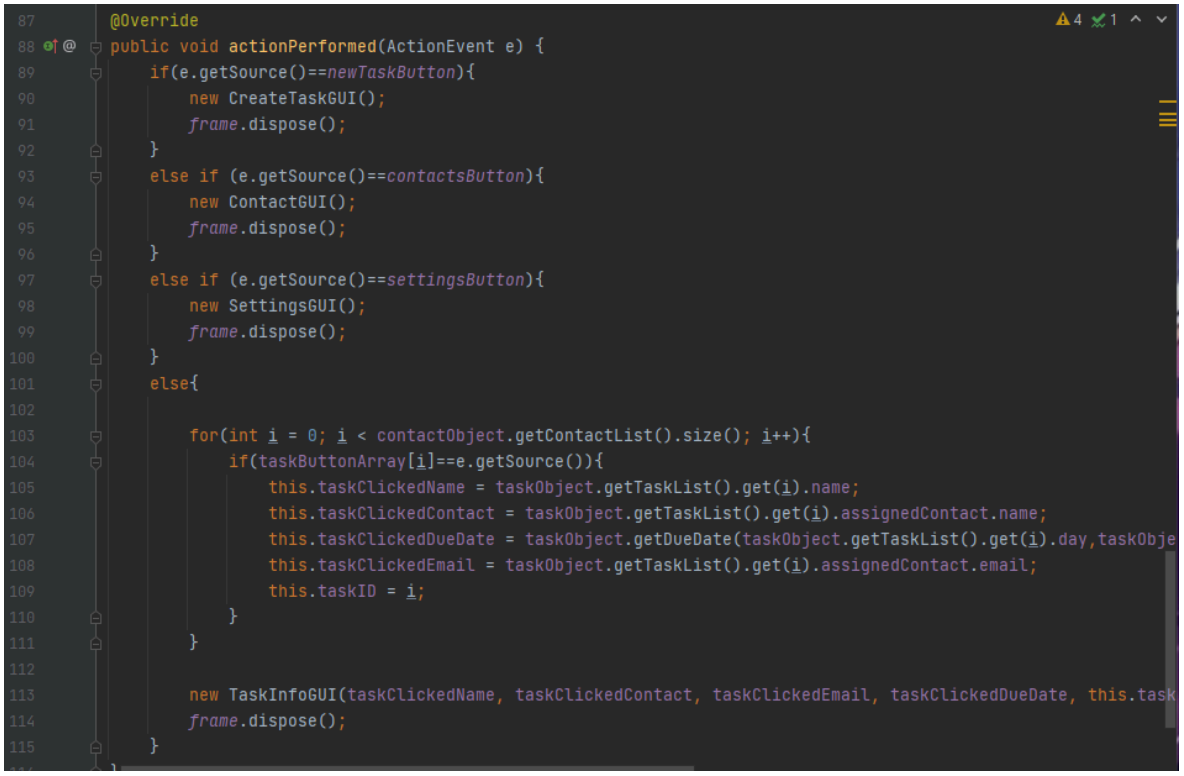


Figure 15: Overridden method actionPerformed, executed when a button is pressed.

This method checks which button is pressed through e.getSource() to create the appropriate menu object but also checks if any of the tasks are pressed to create the TaskInfo menu passing the necessary data as parameters.

New Task Menu:

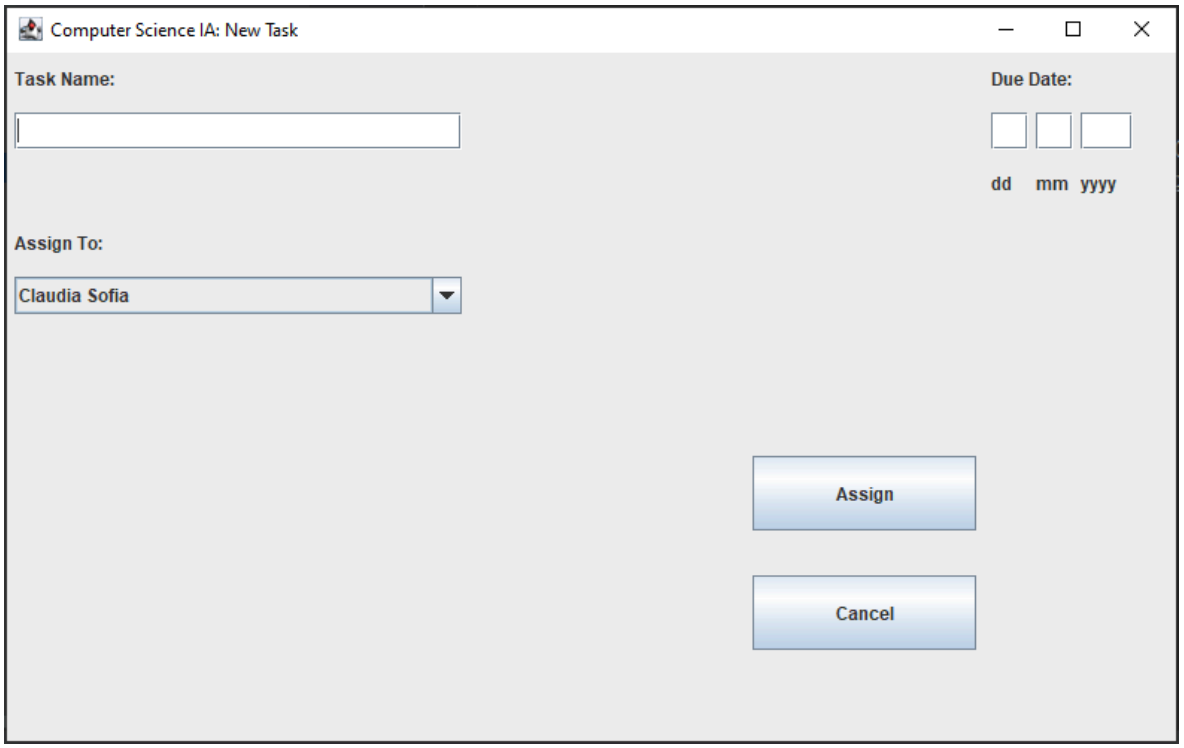


Figure 16: New Task menu. CreateTaskGUI object

New Task menu made following design specifications. The date is in dd/mm/yyyy format because the client prefered it that way.

```
49          assignText = new JComboBox();
50          assignText.setBounds( x: 5, y: 150, width: 300, height: 25);
51          for(int i = 0; i < contactObject.getContactList().size(); i++){
52              assignText.addItem(contactObject.getContactList().get(i).name);
```

Figure 17: code of a JComboBox object. Used for the drop down menu

For creating the drop down menu, a JComboBox object was created with a for loop to add every contact on the contact ArrayList as a selectable item.

Figure 18: CreateTaskGUI overriden actionPerformed method

Opposed to the MainGUI actionPerformed method, when clicking the assign button, this one checks if the date is valid (not a past date or an unexisting date) and creates the new task adding it to the Task class' ArrayList, and calls save() to save the changes in the RandomAccessFiles as well as calling sendMessage() to send an email informing the contact's email of the assigned task.

Task Info Menu:

Figure 19: Task Info Menu. TaskInfoGUI object

Task Info Menu according to proposed design with a slight change. Adding a delete task button, approved by the client.

```java
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource() == cancelButton){
        new MainGUI();
        frame.dispose();
    }
    else if (e.getSource()==deleteButton){
        taskObject.delete(id);
        new MainGUI();
        frame.dispose();
    }
    else if (e.getSource()== advancementsButton){
        new AdvancementsGUI(name, contact, mail, due);
        frame.dispose();
    }

}
```

Figure 20: Use of the delete() method when clicking the delete button.

If clicking the delete button, delete() is called passing as a parameter the task id to remove it from the ArrayList.

Advancements Menu:

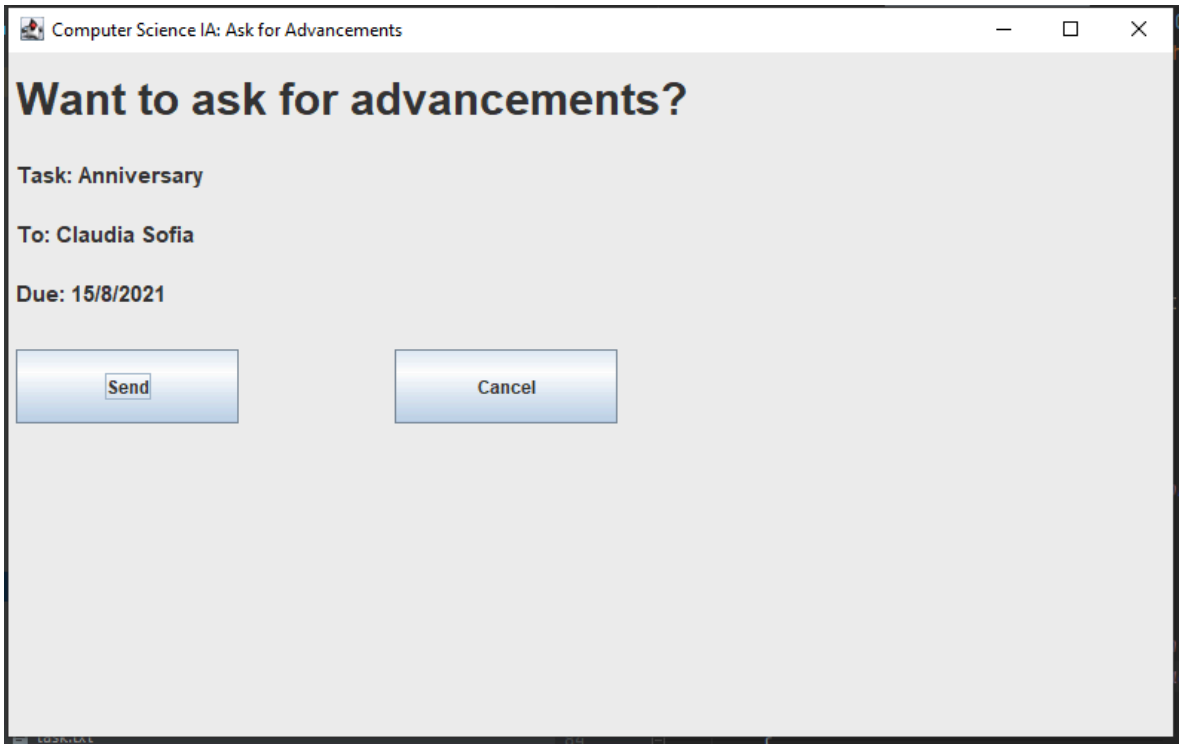


Figure 21: Advancements Menu, AdvancementsGUI object.

This menu was created as planned in the design.

```
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource() == cancelButton){
        new MainGUI();
        frame.dispose();
    }
    else if (e.getSource()== sendButton){
        settingsobject.sendMessage(mailAdress, nameOfTask);
        new MainGUI();
        frame.dispose();

    }
```

Figure 22: Override actionPerformed method of AdvancementsGUI. Demonstrating the use of the method sendMessage().

Contacts Menu:

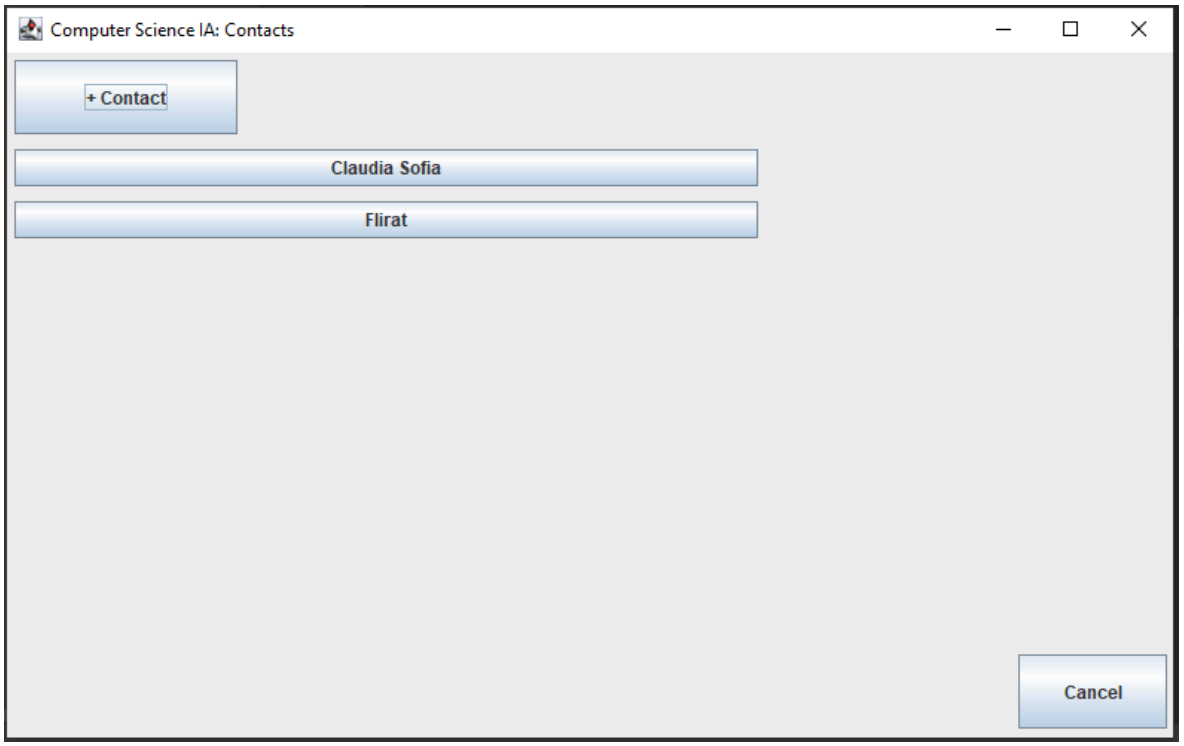


Figure 23: Contact menu, ContactGUI object.

Similar to the main menu. The contacts are presented as buttons that the user can interact with to have information about the contact, or click the "+Contact" button to create a new one.

```
contactButtonArray = new JButton[10];

for(int i = 0; i < contactObject.getContactList().size(); i++){
    contactButtonArray[i] = new JButton(contactObject.getContactList().get(i).name);
    contactButtonArray[i].setBounds( x: 5, y: 65+(35*i), width: 500, height: 25);
    contactButtonArray[i].addActionListener( l: this);

    panel.add(contactButtonArray[i]);

}
```

```java
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource()==cancelButton){
        new MainGUI();
        frame.dispose();
    }
    else if (e.getSource()==newContactButton){
        new NewContactGUI();
        frame.dispose();
    }
    else {

        for(int i = 0; i < contactObject.getContactList().size(); i++){
            if(contactButtonArray[i]==e.getSource()){
                this.contactClickedName = contactObject.getContactList().get(i).name;
                this.contactClickedEmail = contactObject.getContactList().get(i).email;
                this.index = i;
            }
        }

        new ContactInfoGUI(contactClickedName, contactClickedEmail,this.index);
        frame.dispose();
    }
```

Figure 24: Similar uses of a for loop to create a button for each contact, and to pass the needed info to the ContactInfoGUI as seen on Figures 13 and 18.

The difference between the uses of Figures 13 and 18 compared to Figure 24, are that in this case the objects are Contact objects.

New Contact menu:

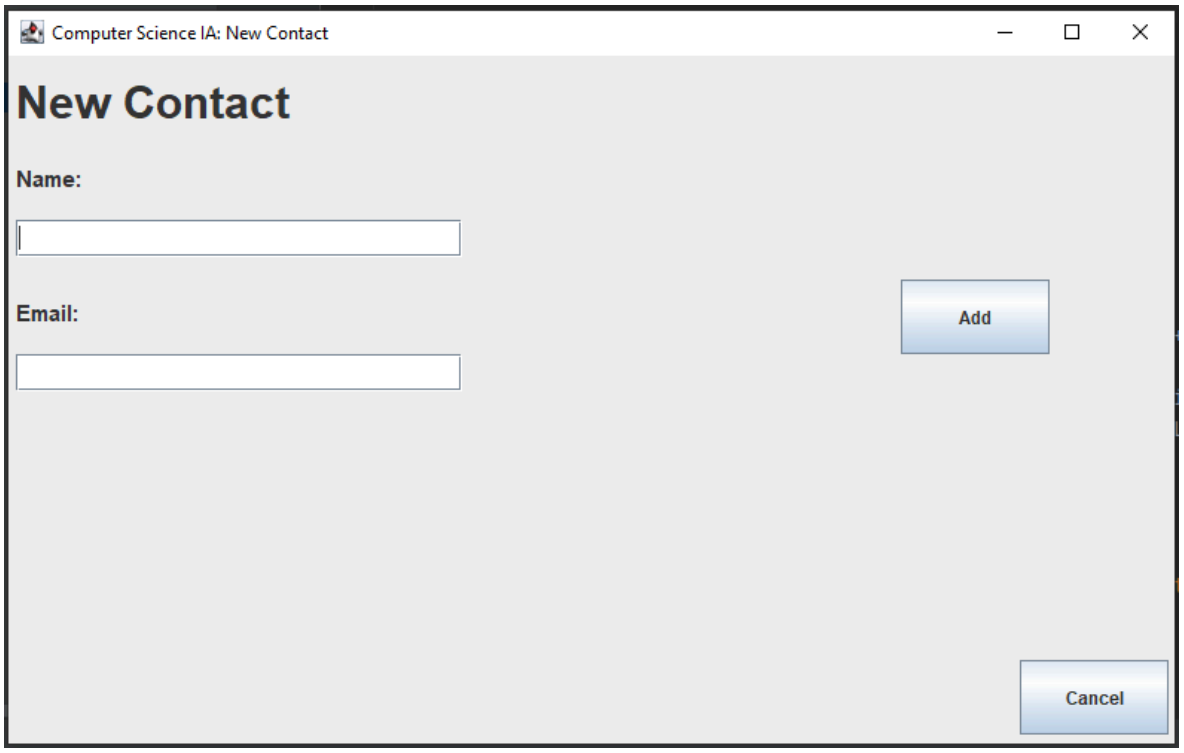


Figure 25: New contact menu, NewContactGUI object

Created following design specifications.

```
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource()==addButton){
        contactObject.add(nameText.getText(), emailText.getText());
        contactObject.save();
        new ContactGUI();
        frame.dispose();
    }
    else if (e.getSource()==cancelButton){
        new ContactGUI();
        frame.dispose();
    }

}
```

Figure 26: Similar method used as seen in Figure 15 to create a new contact. Name and Email are gotten from the text field objects.

Contact Info menu:

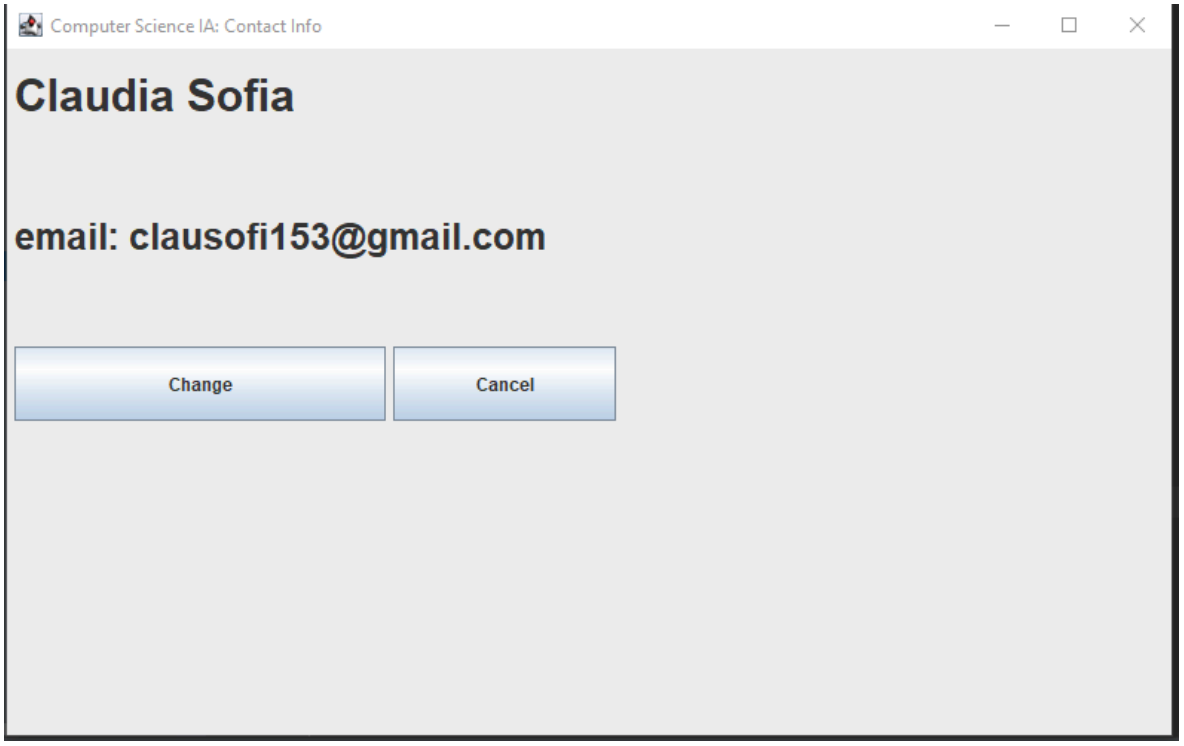
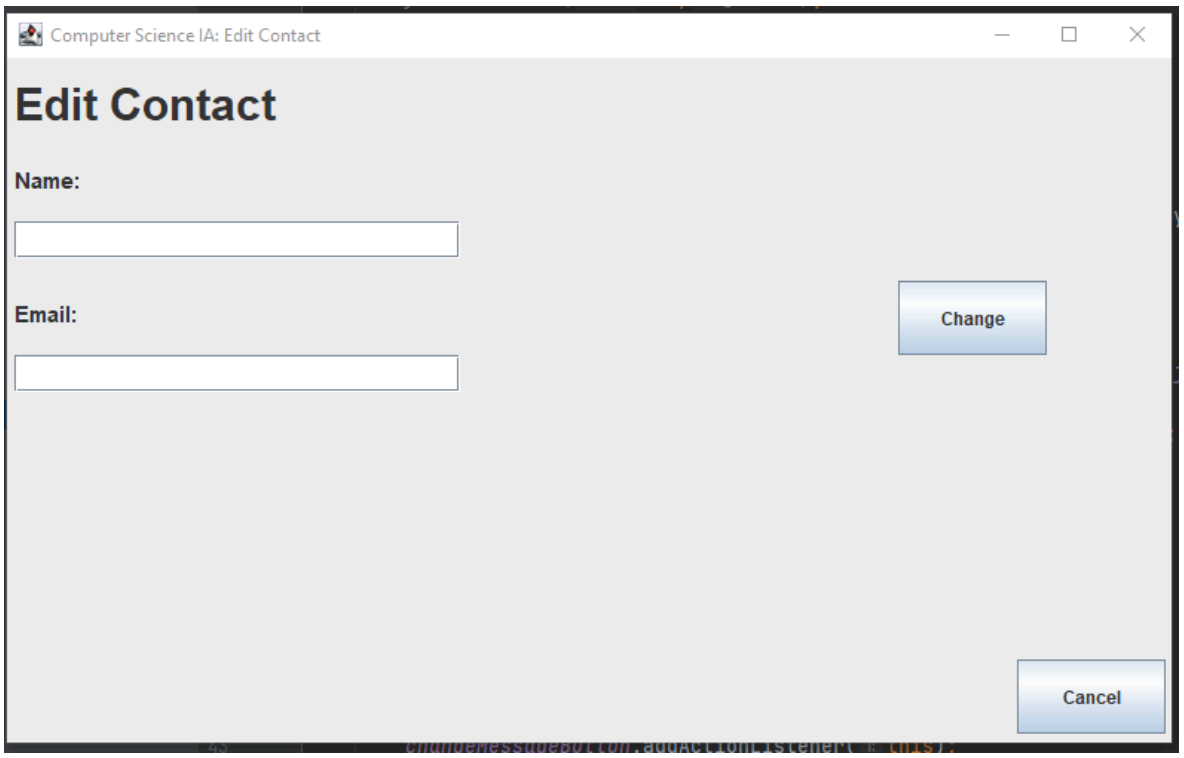


Figure 27: Contact Info menu, ContactInfoGUI

Created following design specifications. The code is similar to the one in the Task Info menu.




```java
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource()==changeButton){
        contactObject.getContactList().set(index, new Contact(nameText.getText(), emailText.getText()));
        contactObject.save();
        new ContactGUI();
        frame.dispose();
    }
    else if (e.getSource()==cancelButton){
        new ContactGUI();
        frame.dispose();
    }

}
```

Figure 28: ChangeContactGUI object, and code showing how the method set() is used to change the value in the ArrayList.

Settings menu:

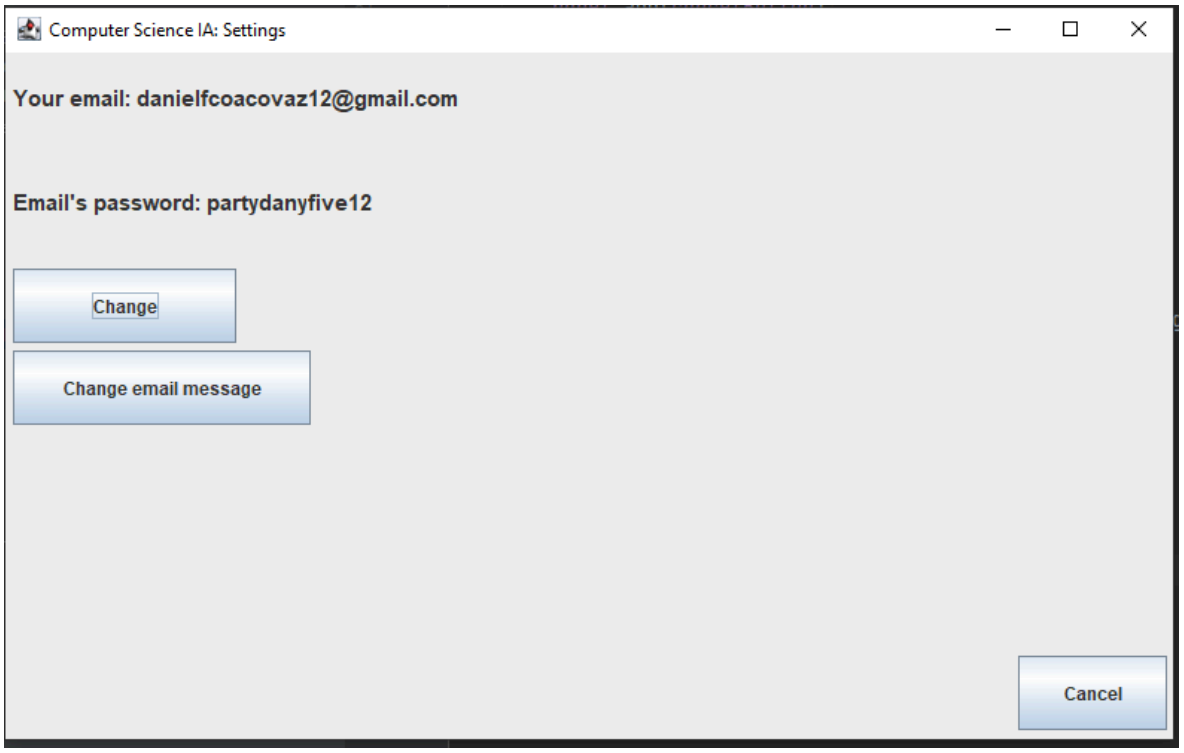


Figure 29: Settings menu, SettingsGUI object.

```
emailLabel = new JLabel( text: "Your email: " + settingsObject.getMyEmail());
emailLabel.setBounds( x: 5, y: 5, width: 300, height: 50);
emailLabel.setFont(emailLabel.getFont().deriveFont(15.0f));
panel.add(emailLabel);

passwordLabel = new JLabel( text: "Email's password: " + settingsObject.getPassword());
passwordLabel.setBounds( x: 5, y: 75, width: 300, height: 50);
passwordLabel.setFont(passwordLabel.getFont().deriveFont(15.0f));
panel.add(passwordLabel);
```

Figure 30: Code showing that the labels that show your email and password are gotten from the Settings class.

Change Email menu:

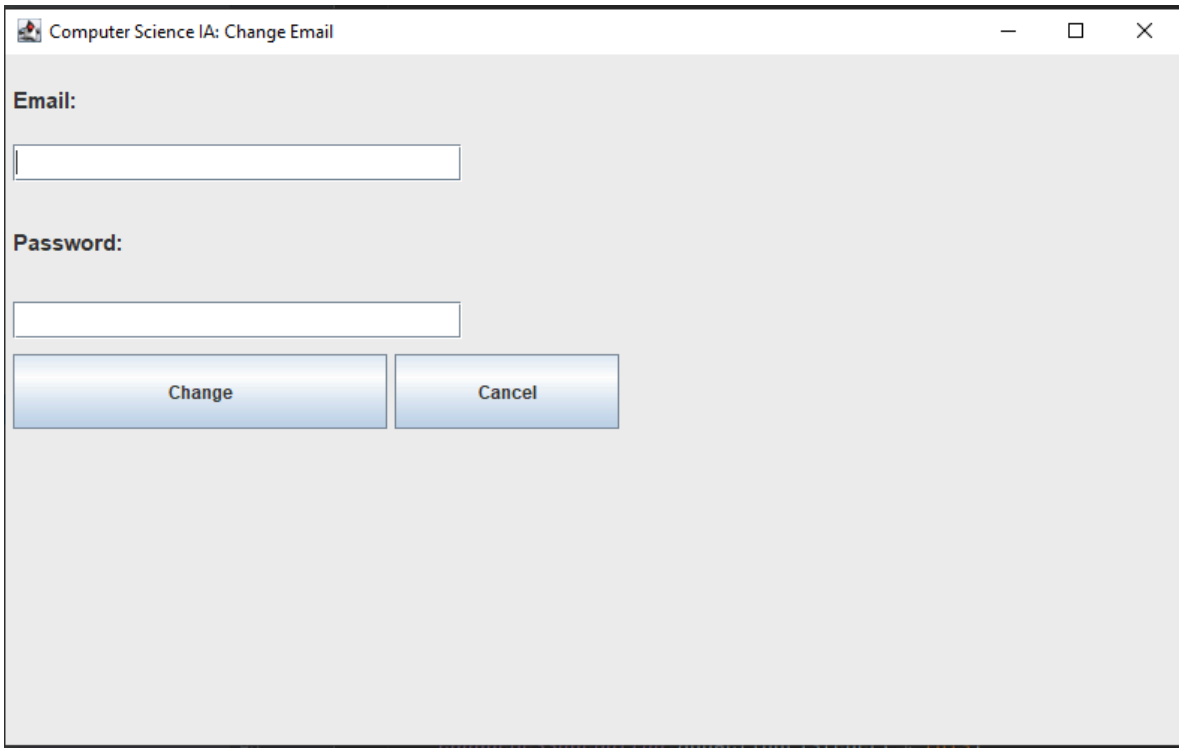
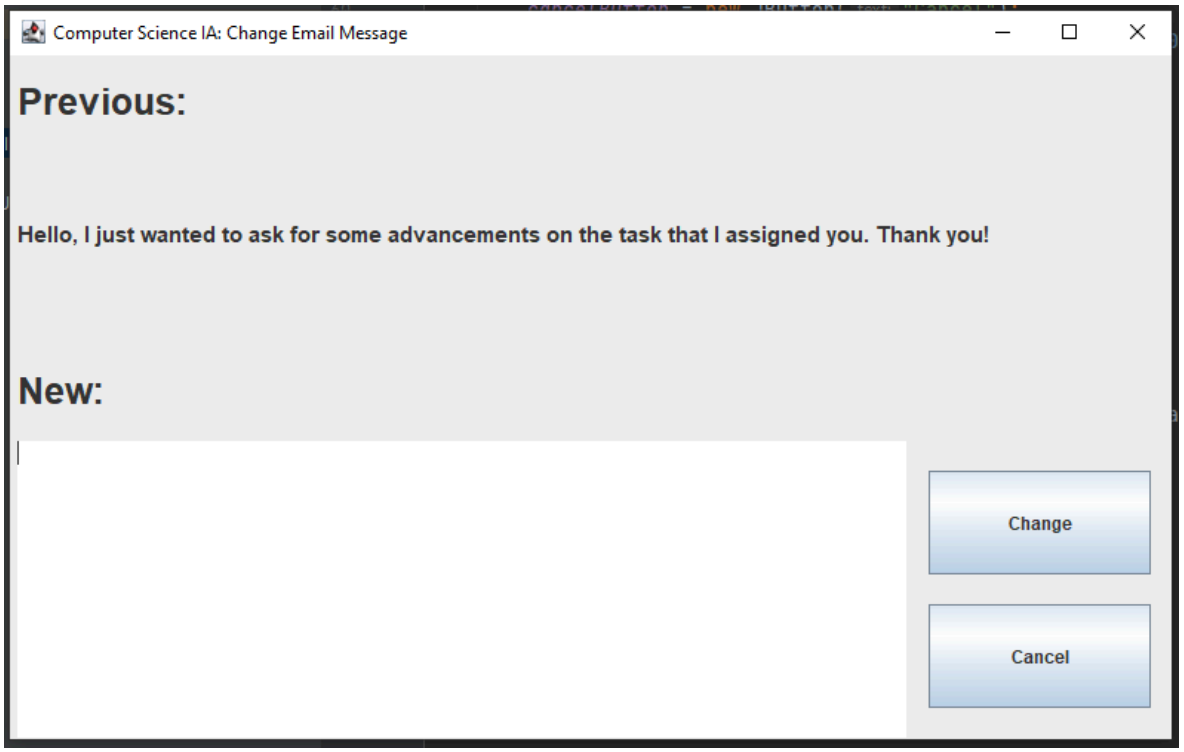


Figure 31: Change Mail menu, ChangeEmailGUI object.

Created following the planned design. Works the same as the change button in the Contact Info menu.

Change Mail Message menu:




```
        if(e.getSource()==changeButton){
            settingsObject.setMailMessage(emailText.getText());
            settingsObject.save();
            new SettingsGUI();
            frame.dispose();
        }
        else if (e.getSource()==cancelButton){
            new SettingsGUI();
            frame.dispose();
        }

    }
```

Figure 32: ChangeMailGUI object and code showing the use of the method setMailMessage() to change the value of the variable storing the mail message in the Settings class

Task class:

```java
public class Task {
    String name;
    Contact assignedContact;
    Contact contactObject = new Contact();
    int day, month, year;
    public static ArrayList<Task> taskList = new ArrayList<Task>();

    public Task(String name, Contact assignedContact, int day, int month, int year) {
        this.name = name;
        this.assignedContact = assignedContact;
        this.day = day;
        this.month = month;
        this.year = year;

    }

    public Task(){

    }
```

Figure 33: Task class constructor. It can be seen the use of the ArrayList, “taskList”

```java
45      public String getDueDate(int day, int month, int year){
46          return (day + "/" + month + "/" + year);
47      }
```

Figure 34: getDueDate() method. Returns a string of date by using as parameters, day, month, and year. Used in Figure 13

```
public void save(){
    Task[] taskArray = new Task[taskList.size()];
    this.taskList.toArray(taskArray);
    String path = new String();
    for(int i = 0; i < taskArray.length; i++){

        switch(i){
            case 0:
                path = "task.txt";
                break;
            case 1:
                path = "task1.txt";
                break;
            case 2:
                path = "task2.txt";
                break;

            case 3:
                path = "task3.txt";
                break;

            case 4:
                path = "task4.txt";
                break;
            case 5:
                path = "task5.txt";
                break;

            case 6:
                path = "task6.txt";
```

```
103         try {
104             RandomAccessFile file = new RandomAccessFile(path,  mode: "rw");
105             file.setLength(0);
106             file.writeBytes( s: taskArray[i].name + "|" + taskArray[i].assignedContact.index + "%" + taskAr
107         }
108         catch (FileNotFoundException e) {
109             e.printStackTrace();
110         } catch (IOException e) {
111             e.printStackTrace();
112         }
113
114     }
```

Figure 35: Save() method in Task class. Used to save the data of tasks in a RandomAccessFile

The save() method works by first converting all the tasks in the ArrayList into a normal array, then uses a for loop to set the path of the file according to the index of the task in the array. Thus, having each task in a separate file. Creates a RandomAccessFile with that path for read/write and saves the data using several indexes to separate it.

```
162     try{
163         RandomAccessFile file = new RandomAccessFile(path,  mode: "r");
164         if(file.length() != 0) {
165             String line = file.readLine();
166             String temporalName = line.substring(0, line.indexOf("|"));
167             int temporalIndex = Integer.parseInt(line.substring(line.indexOf("|") + 1,line.indexOf("%")));
168             int temporalDay = Integer.parseInt(line.substring(line.indexOf("%")+1, line.indexOf("#")));
169             int temporalMonth = Integer.parseInt(line.substring(line.indexOf("#")+1,line.indexOf("$")));
170             int temporalYear = Integer.parseInt(line.substring(line.indexOf("$")+1));
171
172             add(temporalName, contactObject.contactList.get(temporalIndex), temporalDay, temporalMonth, temporal
173             System.out.println("Load completed");
174         }
175     }
176     catch (FileNotFoundException e) {
177         e.printStackTrace();
178     }
179     catch (IOException e) {
180         e.printStackTrace();
181         break;
182     }
```

Figure 36: Example of code for the load() method, used for recovering the data in the RandomAccessFile

The load method functions the same as the save method except for two three things. First, the ArrayList is cleared. The RandomAccessFile is set to read only, and finally, Instead of writing, the program reads the line and stops at the indexes created in the save file to assign values to the variables.

Contact class:

```
9   public class Contact   {
10      public String name;
11      public String email;
12      int index;
13      public static ArrayList<Contact> contactList = new ArrayList<Contact>();
14
15      public Contact(String name, String email, int index) {
16          this.name = name;
17          this.email = email;
18          this.index = index;
19      }
20
21      public Contact(String name, String email) {
22          this.name = name;
23          this.email = email;
24
25      }
26
27      public Contact() {
28
29      }
```

Figure 37: Contact class constructor

Aside from the constructor and the pertinent getters and setters, the Contact class functions the same as the Task class.

Settings class:

```
45    public void save(){
46        try{
47            RandomAccessFile file = new RandomAccessFile( name: "settings.txt", mode: "rw");
48            file.setLength(0);
49            file.writeBytes( s: myEmail + "|" + password + "%" + mailMessage);
50        }
51        catch (FileNotFoundException e) {
52            e.printStackTrace();
53        }
54        catch (IOException e) {
55            e.printStackTrace();
56        }
57
58     }
59
60    public void load(){
61        try{
62            RandomAccessFile file = new RandomAccessFile( name: "settings.txt", mode: "r");
63            if(file.length() != 0){
64                String line = file.readLine();
65                myEmail = line.substring(0, line.indexOf("|"));
66                password = line.substring(line.indexOf("|") + 1, line.indexOf("%"));
67                mailMessage = line.substring(line.indexOf("%") + 1);
68            }
69            else {
70                myEmail = "myemail@default.com";
71                password = "defaultPassword123";
72                mailMessage = "Hello, I just wanted to ask for some advancements on the task that I assigned you. Thank you!";
73                save();
74            }
```

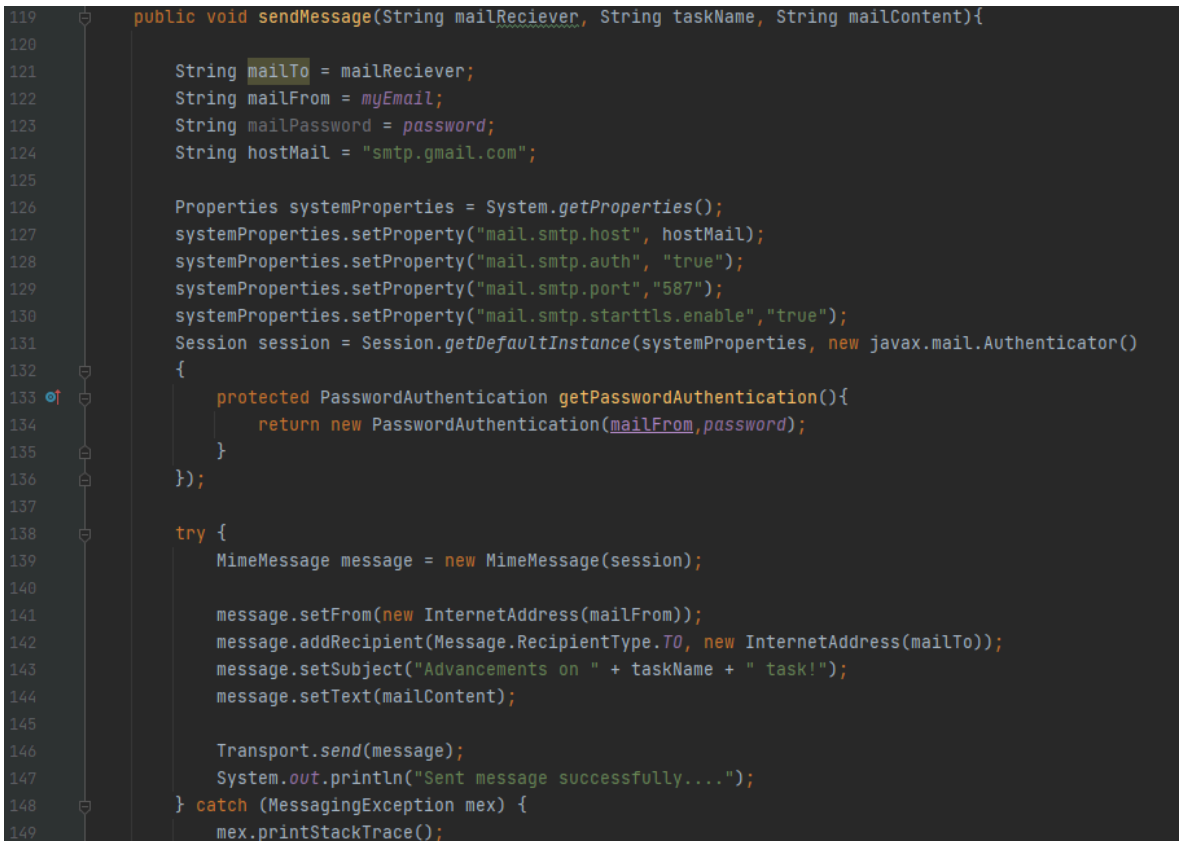
Figure 37: save() and load() methods in settings class

These methods are a simplified version of the ones used in Task and Contact objects, with the difference that if the settings.txt file is empty, default values are loaded.

```java
public void sendMessage(String mailReciever, String taskName){

    String mailTo = mailReciever;
    String mailFrom = myEmail;
    String mailPassword = password;
    String hostMail = "smtp.gmail.com";

    Properties systemProperties = System.getProperties();
    systemProperties.setProperty("mail.smtp.host", hostMail);
    systemProperties.setProperty("mail.smtp.auth", "true");
    systemProperties.setProperty("mail.smtp.port","587");
    systemProperties.setProperty("mail.smtp.starttls.enable","true");
    Session session = Session.getDefaultInstance(systemProperties, new javax.mail.Authenticator()
    {
        protected PasswordAuthentication getPasswordAuthentication(){
            return new PasswordAuthentication(mailFrom,password);
        }
    });

    try {
        MimeMessage message = new MimeMessage(session);

        message.setFrom(new InternetAddress(mailFrom));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(mailTo));
        message.setSubject("Advancements on " + taskName + " task!");
        message.setText(this.getMailMessage());

        Transport.send(message);
        System.out.println("Sent message successfully....");
    } catch (MessagingException mex) {
        mex.printStackTrace();
```

Figure 38: sendMessage() method in settings class.

This method establishes as host the smtp server of gmail and enables the port 587 to send emails. It then authenticates the account using the email address and then does a password authentication. It then starts composing the message and finally sends it.



```
public void sendMessage(String mailReciever, String taskName, String mailContent){

    String mailTo = mailReciever;
    String mailFrom = myEmail;
    String mailPassword = password;
    String hostMail = "smtp.gmail.com";

    Properties systemProperties = System.getProperties();
    systemProperties.setProperty("mail.smtp.host", hostMail);
    systemProperties.setProperty("mail.smtp.auth", "true");
    systemProperties.setProperty("mail.smtp.port","587");
    systemProperties.setProperty("mail.smtp.starttls.enable","true");
    Session session = Session.getDefaultInstance(systemProperties, new javax.mail.Authenticator()
    {
        protected PasswordAuthentication getPasswordAuthentication(){
            return new PasswordAuthentication(mailFrom,password);
        }
    });

    try {
        MimeMessage message = new MimeMessage(session);

        message.setFrom(new InternetAddress(mailFrom));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(mailTo));
        message.setSubject("Advancements on " + taskName + " task!");
        message.setText(mailContent);

        Transport.send(message);
        System.out.println("Sent message successfully....");
    } catch (MessagingException mex) {
        mex.printStackTrace();
```

Figure 39: Example of method overloading with the sendMessage method. Used for sending emails when tasks are assigned or due.

Main method:

```
4   public class Main {
5
6       public static void main(String[] args) {
7
8           Contact contactObject = new Contact();
9           Task taskObject = new Task();
10          Settings settingsObject = new Settings();
11
12          contactObject.load();
13          taskObject.load();
14          settingsObject.load();
15
16          new MainGUI();
17
18
19      }
20  }
```

Figure 40: Main class and main method. On startup, loads the data from the RandomAccessFiles and then creates the main menu.

RandomAccessFiles:

```
1   Graphic Design|1%7#2$2021
```

Figure 41: Example of content of a RandomAccessFile (Task name, task id, due date [day,month,year])

Sources used:

In order to learn how to create a GUI in java I recurred to these three videos:

-Lee, A. (2020, February 6). *Java GUI Tutorial - Make a GUI in 13 Minutes* [Video file]. Retrieved from https://www.youtube.com/watch?v=5o3fMLPY7qY

--Lee, A. (2020, February 13). *Java GUI Tutorial - Make a login GUI* [Video file]. Retrieved from https://www.youtube.com/watch?v=iE8tZ0hn2Ws

-McProgrammer. (2013, December 23). Java Swing GUI 4 - Event Handling with multiple sources (Buttons) [Video File]. Retrieved from: https://www.youtube.com/watch?v=E-yToUMXQck

In order to learn how to use RandomAccessFiles I used:

-Jenkov, J. (2019, April 12). Java RandomAccessFile. Retrieved from http://tutorials.jenkov.com/java-io/randomaccessfile.html

In order to learn how to use an ArrayList I used:

-W3Schools. (n.d.). Java ArrayList. Retrieved from https://www.w3schools.com/java/java_arraylist.asp

In order to learn to use an ItemListener I used:

-Oracle. (n.d.). How to Write an Item Listener. Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/events/itemlistener.html

I based my sendMessage method out of this code:

-Tutorialspoint. (n.d.). Java - Sending Email. Retrieved from https://www.tutorialspoint.com/java/java_sending_email.htm

I used the program IntelliJ Idea Community Edition 2020.3.1

Libraries used:

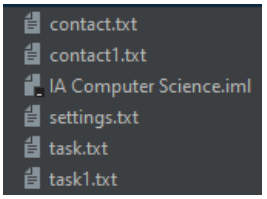
- **Java.swing**
- **Java.awt (java.awt.event.ActionEvent/java.awt.event.ActionListener/ItemEvent/ItemListener)**
- **Java.time (LocalDate)**
- **Java.io (RandomAccessFile/FileNotFoundException/IOException)**
- **Java.util (ArrayList)**
- **Java.mail (internet.InternetAddress/internet.MimeMessage) [gotten from the JavMail API: https://www.oracle.com/java/technologies/javamail145.html]**
- **JAF (JavaBeans Activation Framework): https://www.oracle.com/java/technologies/downloads.html**

Word count: 1294 words

CRITERION D:

Video attached with this file.

CRITERION E

Success Criteria	Evaluation	Evidence
-The software is able to assign multiple tasks to coworkers	Yes, although taking into account the client's feedback (Appendix B), it lacks being able to assign multiple contacts to one task.	-Figures 18 and 33 in Criterion C -Feedback found on Appendix B
-The software is able to assign individual deadlines for each of the tasks	Yes, the software assigns to each task a specific deadline. It is in the format of dd/mm/yyyy to fulfill the client's preferences.	-Figures 13 and 34 in Criterion C
-The software sends an email to notify the coworker of the task assigned	Yes, as soon as the task is assigned, an email is sent notifying the coworker of the assigned task.	-Figure 18 in Criterion C
-The user is able to ask for advancements on the project, which the program will send an email to ask the coworker for advancements	Yes, by clicking the "Ask for Advancements" button on the Advancements menu, the program sends an email asking the coworker for advancements.	-Figures 21 and 22 in Criterion C
-In case a deadline is up or there is no answer at a certain time after asking for the advancement, an alarm in the form of an email can be sent to both the user's and the coworker's email	Yes, when the program detects a due date has passed or today is the last day, the email gets sent notifying both the user and the coworker. The program does this by getting the system's local date.	-Figures 13 and 34 in Criterion C
-All the data, such as the content of emails, emails themselves, or name of the coworkers or the user are strictly private and secured	yes, emails and the content in those emails is securely passed to the service provider (google, Gmail more specifically) in a secure manner. All other info is locally stored in RandomAccessFiles.	 -Figure 41 in Criterion C
-The software is user-friendly	Yes, the software is user friendly and easy to understand with basic buttons and text fields that helps the user know what they are doing and interact with the entirety of the program	-Figure 10 in Criterion C and in Appendix B question 1

Recommendations for further development:

- Add an exit button in case the user can't click on the close button of the window
- Be able to have more contacts and tasks than only 10
- Be able to delete contacts
- Have more email service providers as options *
- Have labels that tell the user the emails have been sent *
- Be able to hide the email password if the user wants it that way
- Be able to add multiple contacts to one task *
- Make clear that in the first time of use you need to set your email and password

*Taking into account client feedback found on Appendix B

Word count: 104 words

Total word count: 2,103 words

Appendix A:

-Interview with Client:
Client: Felix Acosta

1. **What is a problem you have identified that could potentially be solved with the development of some software?**

Due to the pandemic and the quarantine, we were forced to start working each one by themselves in their respective home, which brings up a problem which is that continuing projects becomes increasingly difficult as well as making sure that everyone works exactly the time they are required to.

2. **What can the software do to help you solve the problem?**

Being able to assign work with due times with their respective deadlines, as well as being able to consult the advancements made in the projects. Have alarms that go off to both the assigner and the one in charge of having advancements if the project is not due at the deadline.

3. **Are there any implications or things to consider when developing the software?**

That the information is secured and that the user can easily and securely access their information.

4. **What would you like the software to achieve?**

I would like the software to be assertive, be able to assign things in time and the way I want, that it is user-friendly, and that it is safe.

5. **Do you agree that the software would be successful if it achieves being able to assign tasks with due dates, send off alarms if there is no advancement in the assigned projects, and if the data of both tasks and users are highly secured?**

Yes, of course, I would consider it successful if I am able to assign tasks, that I can see advancements of the projects, that the one working is able to send the advancements they have, and that the alarms work, as well as the privacy of the data involved.

Appendix B:

-Feedback Given By Client:

1.- What feature is the one you consider the most useful?

The feature I consider more useful is the creation of tasks. It is simple and easy to do as well as concise. The alarms work great and it is really easy to know which tasks are due.

2.-What feature would need more improvements?

The accessibility of the mails. It is great to be able to send mails from my gmail account, but not being able to use another email service provider or multiple emails is a downside of this program.

3.- What feedback would you suggest for improving the program?

Be able to have more email accounts as well as different email services. Be able to have a confirmation label for when the emails are sent, as well as being able to assign multiple contacts to one task instead of just one.

4.- Would you consider this program successful? Will it serve you?

Yes, it is certainly successful and it accomplished what I expected. It has its downsides, but for now this program will really solve the problem and would help in this tough times.

Appendix C:

Code of the program:

-Main class:

```
package com.company;


public class Main {

    public static void main(String[] args) {

        Contact contactObject = new Contact();
        Task taskObject = new Task();
        Settings settingsObject = new Settings();

        contactObject.load();
        taskObject.load();
        settingsObject.load();

        new MainGUI();


    }
}
```

-MainGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalDate;

public class MainGUI implements ActionListener {

        private static JFrame frame;
        private static JPanel panel;
        private static JButton newTaskButton, contactsButton, settingsButton;
        private Contact contactObject = new Contact();
        private Task taskObject = new Task();
        private static Settings settingsObject = new Settings();
        int taskID;
        String taskClickedName, taskClickedContact, taskClickedDueDate,
taskClickedEmail;
        JButton[] taskButtonArray;
        JLabel[] taskLabelArray;

    public MainGUI () {

        frame = new JFrame();
```

```
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("Computer Science IA");
        frame.add(panel);

        panel.setLayout(null);

        newTaskButton = new JButton("+ Task");
        newTaskButton.setBounds(5,5,150,50);
        newTaskButton.addActionListener(this);
        panel.add(newTaskButton);

        contactsButton = new JButton("Contacts");
        contactsButton.setBounds(630,5,150,50);
        contactsButton.addActionListener(this);
        panel.add(contactsButton);

        settingsButton = new JButton("Settings");
        settingsButton.setBounds(680,405,100,50);
        settingsButton.addActionListener(this);
        panel.add(settingsButton);


        taskButtonArray = new JButton[10];
        taskLabelArray = new JLabel[10];

        for(int i = 0; i < taskObject.getTaskList().size(); i++){
            taskButtonArray[i] = new
JButton(taskObject.getTaskList().get(i).name);
            taskButtonArray[i].setBounds(5,65+(35*i),500,25);
            taskButtonArray[i].addActionListener(this);

            taskLabelArray[i] = new JLabel("Due: " +
taskObject.getDueDate(taskObject.getTaskList().get(i).day,taskObject.getTaskL
ist().get(i).month, taskObject.getTaskList().get(i).year));
            taskLabelArray[i].setBounds(535,65+(35*i),100,25);

            if(checkDueTask(i)){
                taskLabelArray[i].setForeground(Color.RED);
            }

            panel.add(taskButtonArray[i]);
            panel.add(taskLabelArray[i]);

        }

        frame.setVisible(true);

    }

    public  boolean checkDueTask(int i){
```

```
            if((taskObject.getTaskList().get(i).year<=
LocalDate.now().getYear())&&(taskObject.getTaskList().get(i).month<=LocalDate
.now().getMonthValue())&&

taskObject.getTaskList().get(i).day<=LocalDate.now().getDayOfMonth()){

settingsObject.sendMessage(taskObject.getTaskList().get(i).assignedContact.e
mail,
                        taskObject.getTaskList().get(i).name, "Hey! the
deadline I assigned you is up! Please send me all your advancements now!");

settingsObject.sendMessage(settingsObject.getMyEmail(),taskObject.getTaskLis
t().get(i).name,
                        "The Deadline you assigned for this task is up! Ask
for advancements if you havent received any!");
                return true;
            }
            else{
                return false;
            }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==newTaskButton){
            new CreateTaskGUI();
            frame.dispose();
        }
        else if (e.getSource()==contactsButton){
            new ContactGUI();
            frame.dispose();
        }
        else if (e.getSource()==settingsButton){
            new SettingsGUI();
            frame.dispose();
        }
        else{

            for(int i = 0; i < contactObject.getContactList().size(); i++){
                if(taskButtonArray[i]==e.getSource()){
                    this.taskClickedName =
taskObject.getTaskList().get(i).name;
                    this.taskClickedContact =
taskObject.getTaskList().get(i).assignedContact.name;
                    this.taskClickedDueDate =
taskObject.getDueDate(taskObject.getTaskList().get(i).day,taskObject.getTaskL
ist().get(i).month, taskObject.getTaskList().get(i).year);
                    this.taskClickedEmail =
taskObject.getTaskList().get(i).assignedContact.email;
                    this.taskID = i;
                }
            }
```

```
            new TaskInfoGUI(taskClickedName, taskClickedContact,
taskClickedEmail, taskClickedDueDate, this.taskID);
            frame.dispose();
        }
    }
}
```

-Task class:

```
package com.company;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;

public class Task {
    String name;
    Contact assignedContact;
    Contact contactObject = new Contact();
    int day, month, year;
    public static ArrayList<Task> taskList = new ArrayList<Task>();

    public Task(String name, Contact assignedContact, int day, int month, int
year) {
        this.name = name;
        this.assignedContact = assignedContact;
        this.day = day;
        this.month = month;
        this.year = year;

    }

    public Task(){

    }

    public ArrayList<Task> getTaskList() {
        return this.taskList;
    }

    public void add(String name, Contact contact, int day, int month, int
year){

        this.taskList.add(new Task(name, contact, day, month, year));

    }

    public void delete(int index){

        this.taskList.remove(index);
        save();

    }
```

```
    public String getDueDate(int day, int month, int year){
        return (day + "/" + month + "/" + year);
    }

    public void print(){

        System.out.println(taskList.size());
        for(int i = 0; i < taskList.size(); i++){
            System.out.println(taskList.get(i).name + " " +
taskList.get(i).assignedContact + " Due: " + getDueDate(taskList.get(i).day,
taskList.get(i).month,taskList.get(i).year));
        }

    }

    public void save(){
        Task[] taskArray = new Task[taskList.size()];
        this.taskList.toArray(taskArray);
        String path = new String();
        for(int i = 0; i < taskArray.length; i++){

            switch(i){
                case 0:
                    path = "task.txt";
                    break;
                case 1:
                    path = "task1.txt";
                    break;
                case 2:
                    path = "task2.txt";
                    break;

                case 3:
                    path = "task3.txt";
                    break;

                case 4:
                    path = "task4.txt";
                    break;
                case 5:
                    path = "task5.txt";
                    break;

                case 6:
                    path = "task6.txt";
                    break;

                case 7:
                    path = "task7.txt";
                    break;

                case 8:
```

```
                    path = "task8.txt";
                    break;

                case 9:
                    path = "task9.txt";
                    break;
            }

            try {
                RandomAccessFile file = new RandomAccessFile(path, "rw");
                file.setLength(0);
                file.writeBytes(taskArray[i].name + "|" +
taskArray[i].assignedContact.index + "%" + taskArray[i].day + "#" +
taskArray[i].month + "$" + taskArray[i].year);
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

        }


    }

    public void load() {

        this.taskList.clear();
        String path = new String();
        for(int i = 0; i < 10; i++){
            switch(i){
                case 0:
                    path = "task.txt";
                    break;
                case 1:
                    path = "task1.txt";
                    break;
                case 2:
                    path = "task2.txt";
                    break;

                case 3:
                    path = "task3.txt";
                    break;

                case 4:
                    path = "task4.txt";
                    break;
                case 5:
                    path = "task5.txt";
                    break;
```

```
                case 6:
                    path = "task6.txt";
                    break;

                case 7:
                    path = "task7.txt";
                    break;

                case 8:
                    path = "task8.txt";
                    break;

                case 9:
                    path = "task9.txt";
                    break;
            }
            try{
                RandomAccessFile file = new RandomAccessFile(path, "r");
                if(file.length() != 0) {
                    String line = file.readLine();
                    String temporalName = line.substring(0,
line.indexOf("|"));
                    int temporalIndex =
Integer.parseInt(line.substring(line.indexOf("|") + 1,line.indexOf("%")));
                    int temporalDay =
Integer.parseInt(line.substring(line.indexOf("%")+1, line.indexOf("#")));
                    int temporalMonth =
Integer.parseInt(line.substring(line.indexOf("#")+1,line.indexOf("$")));
                    int temporalYear =
Integer.parseInt(line.substring(line.indexOf("$")+1));

                    add(temporalName,
contactObject.contactList.get(temporalIndex), temporalDay, temporalMonth,
temporalYear);
                    System.out.println("Load completed");
                }
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            catch (IOException e) {
                e.printStackTrace();
                break;
            }
        }

    }

}
```

-Contact Class:

```
package com.company;


import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;

public class Contact    {
    public String name;
    public String email;
    int index;
    public static ArrayList<Contact> contactList = new ArrayList<Contact>();

    public Contact(String name, String email, int index) {
        this.name = name;
        this.email = email;
        this.index = index;
    }

    public Contact(String name, String email) {
        this.name = name;
        this.email = email;

    }

    public Contact() {

    }

    public ArrayList<Contact> getContactList() {
        return this.contactList;
    }

    public void add(String name, String email){

        this.contactList.add(new Contact(name,email, contactList.size()));

    }

    public void print(){

        for(int i = 0; i < contactList.size(); i++){
            System.out.println(contactList.get(i).name + " " +
contactList.get(i).email);
        }

    }

    public void save(){
        Contact[] contactArray = new Contact[getContactList().size()];
        this.contactList.toArray(contactArray);
```

```
        String path = new String();
        for(int i = 0; i < contactArray.length; i++){

            switch(i){
                case 0:
                    path = "contact.txt";
                    break;
                case 1:
                    path = "contact1.txt";
                    break;
                case 2:
                    path = "contact2.txt";
                    break;

                case 3:
                    path = "contact3.txt";
                    break;

                case 4:
                    path = "contact4.txt";
                    break;
                case 5:
                    path = "contact5.txt";
                    break;

                case 6:
                    path = "contact6.txt";
                    break;

                case 7:
                    path = "contact7.txt";
                    break;

                case 8:
                    path = "contact8.txt";
                    break;

                case 9:
                    path = "contact9.txt";
                    break;
            }

            try {
                RandomAccessFile file = new RandomAccessFile(path, "rw");
                file.setLength(0);
                file.writeBytes(contactArray[i].name + "|" +
contactArray[i].email);
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
        }


    }

    public void load() {

        this.contactList.clear();
        String path = new String();
        for(int i = 0; i < 10; i++){
            switch(i){
                case 0:
                    path = "contact.txt";
                    break;
                case 1:
                    path = "contact1.txt";
                    break;
                case 2:
                    path = "contact2.txt";
                    break;

                case 3:
                    path = "contact3.txt";
                    break;

                case 4:
                    path = "contact4.txt";
                    break;
                case 5:
                    path = "contact5.txt";
                    break;

                case 6:
                    path = "contact6.txt";
                    break;

                case 7:
                    path = "contact7.txt";
                    break;

                case 8:
                    path = "contact8.txt";
                    break;

                case 9:
                    path = "contact9.txt";
                    break;
            }
            try{
                RandomAccessFile file = new RandomAccessFile(path, "r");
                if(file.length() != 0) {
                    String line = file.readLine();
```

```
                    String temporalName = line.substring(0,
line.indexOf("|"));
                    String temporalEmail = line.substring(line.indexOf("|") +
1);
                    add(temporalName, temporalEmail);
                    System.out.println("Load completed");
                }
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            catch (IOException e) {
                e.printStackTrace();
                break;
            }
        }

    }

}
```

-Settings Class:

```
package com.company;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Properties;
import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;


public class Settings {

    private static String myEmail, password, mailMessage;
    private static Task taskObject = new Task();

     public Settings(){

     }

    public String getMyEmail() {
        return myEmail;
    }

    public String getPassword() {
        return password;
    }

    public String getMailMessage() {
        return mailMessage;
    }
```

```
    public void setMyEmail(String myEmail) {
        Settings.myEmail = myEmail;
    }

    public void setPassword(String password) {
        Settings.password = password;
    }

    public void setMailMessage(String mailMessage) {
        Settings.mailMessage = mailMessage;
    }

    public void save(){
            try{
                RandomAccessFile file = new
RandomAccessFile("settings.txt","rw");
                file.setLength(0);
                file.writeBytes(myEmail + "|" + password + "%" + mailMessage);
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            catch (IOException e) {
                e.printStackTrace();
            }

    }

    public void load(){
            try{
                RandomAccessFile file = new
RandomAccessFile("settings.txt","r");
                if(file.length() != 0){
                    String line = file.readLine();
                    myEmail = line.substring(0, line.indexOf("|"));
                    password = line.substring(line.indexOf("|") + 1,
line.indexOf("%"));
                    mailMessage = line.substring(line.indexOf("%") + 1);
                }
                else {
                    myEmail = "myemail@default.com";
                    password = "defaultPassword123";
                    mailMessage = "Hello, I just wanted to ask for some
advancements on the task that I assigned you. Thank you!";
                    save();
                }
            }
            catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            catch (IOException e) {
                e.printStackTrace();
```

```
        }
    }

    public void sendMessage(String mailReciever, String taskName){

        String mailTo = mailReciever;
        String mailFrom = myEmail;
        String mailPassword = password;
        String hostMail = "smtp.gmail.com";

        Properties systemProperties = System.getProperties();
        systemProperties.setProperty("mail.smtp.host", hostMail);
        systemProperties.setProperty("mail.smtp.auth", "true");
        systemProperties.setProperty("mail.smtp.port","587");
        systemProperties.setProperty("mail.smtp.starttls.enable","true");
        Session session = Session.getDefaultInstance(systemProperties, new
javax.mail.Authenticator()
        {
            protected PasswordAuthentication getPasswordAuthentication(){
                return new PasswordAuthentication(mailFrom,password);
            }
        });

        try {
            MimeMessage message = new MimeMessage(session);

            message.setFrom(new InternetAddress(mailFrom));
            message.addRecipient(Message.RecipientType.TO, new
InternetAddress(mailTo));
            message.setSubject("Advancements on " + taskName + " task!");
            message.setText(this.getMailMessage());

            Transport.send(message);
            System.out.println("Sent message successfully....");
        } catch (MessagingException mex) {
            mex.printStackTrace();
        }

    }

    public void sendMessage(String mailReciever, String taskName, String
mailContent){

        String mailTo = mailReciever;
        String mailFrom = myEmail;
        String mailPassword = password;
        String hostMail = "smtp.gmail.com";

        Properties systemProperties = System.getProperties();
        systemProperties.setProperty("mail.smtp.host", hostMail);
        systemProperties.setProperty("mail.smtp.auth", "true");
        systemProperties.setProperty("mail.smtp.port","587");
        systemProperties.setProperty("mail.smtp.starttls.enable","true");
```

```
        Session session = Session.getDefaultInstance(systemProperties, new
javax.mail.Authenticator()
        {
            protected PasswordAuthentication getPasswordAuthentication(){
                return new PasswordAuthentication(mailFrom,password);
            }
        });

        try {
            MimeMessage message = new MimeMessage(session);

            message.setFrom(new InternetAddress(mailFrom));
            message.addRecipient(Message.RecipientType.TO, new
InternetAddress(mailTo));
            message.setSubject("Advancements on " + taskName + " task!");
            message.setText(mailContent);

            Transport.send(message);
            System.out.println("Sent message successfully....");
        } catch (MessagingException mex) {
            mex.printStackTrace();
        }

    }

}
```

-CreateTaskGUI class:

```
package com.company;


import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.time.LocalDate;

public class CreateTaskGUI implements ActionListener, ItemListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel taskNameLabel, dueDateLabel, assignToLabel,
dayLabel, monthLabel, yearLabel, errorLabel;
    private static JTextField taskNameText,dayText, monthText, yearText;
    private static JButton assignButton, cancelButton;
    private JComboBox assignText;
    Contact contactObject = new Contact();
    Contact selectedContact = new Contact();
    Task taskObject = new Task();
    Settings settingsObject = new Settings();
    int day, month,year;
```

```
    public CreateTaskGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: New Task");
        frame.add(panel);

        panel.setLayout(null);

        taskNameLabel = new JLabel("Task Name:");
        taskNameLabel.setBounds(5,5,80,25);
        panel.add(taskNameLabel);

        taskNameText = new JTextField(20);
        taskNameText.setBounds(5,40,300,25);
        panel.add(taskNameText);

        assignToLabel = new JLabel("Assign To:");
        assignToLabel.setBounds(5,115,80,25);
        panel.add(assignToLabel);

        assignText = new JComboBox();
        assignText.setBounds(5,150,300,25);
        for(int i = 0; i < contactObject.getContactList().size(); i++){
            assignText.addItem(contactObject.getContactList().get(i).name);
        }
        assignText.addItemListener(this);
        panel.add(assignText);

        dueDateLabel = new JLabel("Due Date:");
        dueDateLabel.setBounds(660,5,80,25);
        panel.add(dueDateLabel);

        dayText = new JTextField();
        dayText.setBounds(660,40,25,25);
        panel.add(dayText);

        dayLabel = new JLabel("dd");
        dayLabel.setBounds(660,75,25,25);
        panel.add(dayLabel);

        monthText = new JTextField();
        monthText.setBounds(690,40,25,25);
        panel.add(monthText);

        monthLabel = new JLabel("mm");
        monthLabel.setBounds(690,75,25,25);
        panel.add(monthLabel);

        yearText = new JTextField();
```

```
        yearText.setBounds(720,40,35,25);
        panel.add(yearText);

        yearLabel = new JLabel("yyyy");
        yearLabel.setBounds(720,75,35,25);
        panel.add(yearLabel);

        assignButton = new JButton("Assign");
        assignButton.setBounds(500,270,150,50);
        assignButton.addActionListener(this);
        panel.add(assignButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(500,350,150,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        errorLabel = new JLabel("Please enter a valid Date");
        errorLabel.setBounds(480,150,180,50);
        errorLabel.setFont(errorLabel.getFont().deriveFont(15.0f));
        panel.add(errorLabel);
        errorLabel.setVisible(false);

        frame.setVisible(true);
    }

    public int setDayLimit(int month, int year){
        switch(month) {
            case 1:
                return 31;
            case 2:
                return (year%4==0) ? 29 : 28;
            case 3:
                return 31;
            case 4:
                return 30;
            case 5:
                return 31;
            case 6:
                return 30;
            case 7:
                return 31;
            case 8:
                return 31;
            case 9:
                return 30;
            case 10:
                return 31;
            case 11:
                return 30;
            case 12:
                return 31;
            default:
```

```
                return 31;
            }
        }


    @Override
    public void actionPerformed(ActionEvent e) {
            if(e.getSource() == cancelButton){
                new MainGUI();
                frame.dispose();
            }
            else if (e.getSource()== assignButton) {
                day = Integer.parseInt(dayText.getText());
                month = Integer.parseInt(monthText.getText());
                year = Integer.parseInt(yearText.getText());
                    if ((month > 0 && month < 13) && (year >=
LocalDate.now().getYear()) && (day > 0 && day <= setDayLimit(month, year))) {
                        if (year > LocalDate.now().getYear() || month >
LocalDate.now().getMonthValue() || (month == LocalDate.now().getMonthValue()
&& day >= LocalDate.now().getDayOfMonth())) {
                            taskObject.add(taskNameText.getText(),
selectedContact, day, month, year);
                            taskObject.print();
                            taskObject.save();
                            settingsObject.sendMessage(selectedContact.email,
taskNameText.getText(), "Hey " + selectedContact.name + " I just assigned
you a new task! For more info, contact me!");
                            new MainGUI();
                            frame.dispose();
                        } else {
                            errorLabel.setVisible(true);
                        }
                    } else {
                        errorLabel.setVisible(true);
                    }

            }

    }


    @Override
    public void itemStateChanged(ItemEvent e) {

            for(int i = 0; i < contactObject.getContactList().size(); i++){

                if(e.getItem().toString()==
contactObject.getContactList().get(i).name&&e.getStateChange()==1){
                    selectedContact = (new
Contact(contactObject.getContactList().get(i).name,contactObject.getContactLi
st().get(i).email,i));
                    System.out.println(selectedContact.name + " " +
selectedContact.email + " " + selectedContact.index);
```

```
            }

        }

    }
}
```

-TaskInfoGUI Class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TaskInfoGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel taskNameLabel, assignedToLabel, dueLabel;
    private static JButton advancementsButton, cancelButton, deleteButton;
    private static Task taskObject = new Task();
    private static int id;
    private static String name, contact, due, mail;


    public TaskInfoGUI(String taskName, String taskContact, String
contactEmail, String dueDate, int index) {

        id = index;
        name = taskName;
        contact = taskContact;
        due = dueDate;
        mail = contactEmail;

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Task Info");
        frame.add(panel);

        panel.setLayout(null);

        taskNameLabel = new JLabel(taskName);
        taskNameLabel.setBounds(5,5,300,50);
        taskNameLabel.setFont(taskNameLabel.getFont().deriveFont(30.0f));
        panel.add(taskNameLabel);

        assignedToLabel = new JLabel("Assigned to: " + taskContact);
        assignedToLabel.setBounds(5,70,300,25);
        assignedToLabel.setFont(assignedToLabel.getFont().deriveFont(15.0f));
```

```
        panel.add(assignedToLabel);

        dueLabel = new JLabel("Due: " + dueDate);
        dueLabel.setBounds(5,110,300,25);
        dueLabel.setFont(dueLabel.getFont().deriveFont(15.0f));
        panel.add(dueLabel);

        advancementsButton = new JButton("Ask for Advancements");
        advancementsButton.setBounds(5,200,250,50);
        advancementsButton.addActionListener(this);
        panel.add(advancementsButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(260,200,150,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        deleteButton = new JButton("Delete");
        deleteButton.setBounds(420,200,150,50);
        deleteButton.addActionListener(this);
        panel.add(deleteButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == cancelButton){
            new MainGUI();
            frame.dispose();
        }
        else if (e.getSource()==deleteButton){
            taskObject.delete(id);
            new MainGUI();
            frame.dispose();
        }
        else if (e.getSource()== advancementsButton){
            new AdvancementsGUI(name, contact, mail, due);
            frame.dispose();
        }

    }
}
```

-AdvancementsGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
public class AdvancementsGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel askLabel, taskLabel, assignedToLabel, dueLabel;
    private static JButton sendButton, cancelButton;
    private static Settings settingsobject = new Settings();
    private static  String mailAdress, nameOfTask;

    public AdvancementsGUI(String taskName, String taskContact, String
contactEmail, String dueDate){

        mailAdress = contactEmail;
        nameOfTask = taskName;

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Ask for Advancements");
        frame.add(panel);

        panel.setLayout(null);

        askLabel = new JLabel("Want to ask for advancements?");
        askLabel.setBounds(5,5,500,50);
        askLabel.setFont(askLabel.getFont().deriveFont(30.0f));
        panel.add(askLabel);

        taskLabel = new JLabel("Task: " + taskName);
        taskLabel.setBounds(5,70,300,25);
        taskLabel.setFont(taskLabel.getFont().deriveFont(15.0f));
        panel.add(taskLabel);

        assignedToLabel = new JLabel("To: " + taskContact);
        assignedToLabel.setBounds(5,110,300,25);
        assignedToLabel.setFont(assignedToLabel.getFont().deriveFont(15.0f));
        panel.add(assignedToLabel);

        dueLabel = new JLabel("Due: " + dueDate);
        dueLabel.setBounds(5,150,300,25);
        dueLabel.setFont(dueLabel.getFont().deriveFont(15.0f));
        panel.add(dueLabel);

        sendButton = new JButton("Send");
        sendButton.setBounds(5,200,150,50);
        sendButton.addActionListener(this);
        panel.add(sendButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(260,200,150,50);
        cancelButton.addActionListener(this);
```

```
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == cancelButton){
            new MainGUI();
            frame.dispose();
        }
        else if (e.getSource()== sendButton){
            settingsobject.sendMessage(mailAdress, nameOfTask);
            new MainGUI();
            frame.dispose();

        }

    }
}
```

-ContactGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ContactGUI implements ActionListener {

    public static JFrame frame;
    public static JPanel panel;
    public static JButton newContactButton, cancelButton;
    private Contact contactObject = new Contact();
    public String contactClickedName, contactClickedEmail;
    public int index;
    JButton[] contactButtonArray;

    public ContactGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Contacts");
        frame.add(panel);

        panel.setLayout(null);

        newContactButton = new JButton("+ Contact");
        newContactButton.setBounds(5,5,150,50);
```

```
        newContactButton.addActionListener(this);
        panel.add(newContactButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(680,405,100,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        contactObject.print();

        contactButtonArray = new JButton[10];

        for(int i = 0; i < contactObject.getContactList().size(); i++){
            contactButtonArray[i] = new
JButton(contactObject.getContactList().get(i).name);
            contactButtonArray[i].setBounds(5,65+(35*i),500,25);
            contactButtonArray[i].addActionListener(this);

            panel.add(contactButtonArray[i]);

        }

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource()==cancelButton){
            new MainGUI();
            frame.dispose();
        }
        else if (e.getSource()==newContactButton){
            new NewContactGUI();
            frame.dispose();
        }
        else {

            for(int i = 0; i < contactObject.getContactList().size(); i++){
                if(contactButtonArray[i]==e.getSource()){
                    this.contactClickedName =
contactObject.getContactList().get(i).name;
                    this.contactClickedEmail =
contactObject.getContactList().get(i).email;
                    this.index = i;
                }
            }

            new ContactInfoGUI(contactClickedName,
contactClickedEmail,this.index);
            frame.dispose();
        }
```

```
    }
}
```

-ChangeContactGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeContactGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel changeContactLabel, nameLabel, emailLabel;
    private static JTextField emailText, nameText;
    private static JButton changeButton, cancelButton;
    public Contact contactObject = new Contact();
    int index;


    public ChangeContactGUI(int contactNo){

        index = contactNo;

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Edit Contact");
        frame.add(panel);

        panel.setLayout(null);

        changeContactLabel = new JLabel("Edit Contact");
        changeContactLabel.setBounds(5,5,300,50);

changeContactLabel.setFont(changeContactLabel.getFont().deriveFont(30.0f));
        panel.add(changeContactLabel);

        nameLabel = new JLabel("Name:");
        nameLabel.setBounds(5,70,150,25);
        nameLabel.setFont(nameLabel.getFont().deriveFont(15.0f));
        panel.add(nameLabel);

        nameText = new JTextField();
        nameText.setBounds(5,110,300,25);
        panel.add(nameText);

        emailLabel = new JLabel("Email:");
        emailLabel.setBounds(5,160,150,25);
```

```
        emailLabel.setFont(emailLabel.getFont().deriveFont(15.0f));
        panel.add(emailLabel);

        emailText = new JTextField();
        emailText.setBounds(5,200,300,25);
        panel.add(emailText);

        changeButton = new JButton("Change");
        changeButton.setBounds(600,150,100,50);
        changeButton.addActionListener(this);
        panel.add(changeButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(680,405,100,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource()==changeButton){
            contactObject.getContactList().set(index, new
Contact(nameText.getText(), emailText.getText()));
            contactObject.save();
            new ContactGUI();
            frame.dispose();
        }
        else if (e.getSource()==cancelButton){
            new ContactGUI();
            frame.dispose();
        }

    }

}
```

-NewContactGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class NewContactGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel newContactLabel, nameLabel, emailLabel;
```

```
    private static JTextField emailText, nameText;
    private static JButton addButton, cancelButton;
    private Contact contactObject = new Contact();


    public NewContactGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: New Contact");
        frame.add(panel);

        panel.setLayout(null);

        newContactLabel = new JLabel("New Contact");
        newContactLabel.setBounds(5,5,300,50);
        newContactLabel.setFont(newContactLabel.getFont().deriveFont(30.0f));
        panel.add(newContactLabel);

        nameLabel = new JLabel("Name:");
        nameLabel.setBounds(5,70,150,25);
        nameLabel.setFont(nameLabel.getFont().deriveFont(15.0f));
        panel.add(nameLabel);

        nameText = new JTextField();
        nameText.setBounds(5,110,300,25);
        panel.add(nameText);

        emailLabel = new JLabel("Email:");
        emailLabel.setBounds(5,160,150,25);
        emailLabel.setFont(emailLabel.getFont().deriveFont(15.0f));
        panel.add(emailLabel);

        emailText = new JTextField();
        emailText.setBounds(5,200,300,25);
        panel.add(emailText);

        addButton = new JButton("Add");
        addButton.setBounds(600,150,100,50);
        addButton.addActionListener(this);
        panel.add(addButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(680,405,100,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }
```

```
    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource()==addButton){
            contactObject.add(nameText.getText(), emailText.getText());
            contactObject.save();
            new ContactGUI();
            frame.dispose();
        }
        else if (e.getSource()==cancelButton){
            new ContactGUI();
            frame.dispose();
        }

    }
}
```

-ContactInfoGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ContactInfoGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel contactNameLabel, contactEmailLabel;
    private static JButton changeButton, cancelButton;
    String contactName, contactEmail;
    int contactNo;



    public ContactInfoGUI(String name, String email, int index){

        contactName = name;
        contactEmail = email;
        contactNo = index;


        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Contact Info");
        frame.add(panel);

        panel.setLayout(null);

        contactNameLabel = new JLabel(contactName);
```

```
        contactNameLabel.setBounds(5,5,300,50);

contactNameLabel.setFont(contactNameLabel.getFont().deriveFont(30.0f));
        panel.add(contactNameLabel);

        contactEmailLabel = new JLabel("email: " + contactEmail);
        contactEmailLabel.setBounds(5,100,600,50);

contactEmailLabel.setFont(contactEmailLabel.getFont().deriveFont(25.0f));
        panel.add(contactEmailLabel);

        changeButton = new JButton("Change");
        changeButton.setBounds(5,200,250,50);
        changeButton.addActionListener(this);
        panel.add(changeButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(260,200,150,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == cancelButton){
            new ContactGUI();
            frame.dispose();
        }
        else if (e.getSource()== changeButton){

            new ChangeContactGUI(contactNo);
            frame.dispose();

        }

    }
}
```

-SettingsGUI class

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SettingsGUI implements ActionListener {

    public static JFrame frame;
```

```
    public static JPanel panel;
    public static JLabel emailLabel, passwordLabel;
    public static JButton changeButton, changeMessageButton, cancelButton;
    public static Settings settingsObject = new Settings();

    public SettingsGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Settings");
        frame.add(panel);

        panel.setLayout(null);

        emailLabel = new JLabel("Your email: " +
settingsObject.getMyEmail());
        emailLabel.setBounds(5,5,300,50);
        emailLabel.setFont(emailLabel.getFont().deriveFont(15.0f));
        panel.add(emailLabel);

        passwordLabel = new JLabel("Email's password: " +
settingsObject.getPassword());
        passwordLabel.setBounds(5,75,300,50);
        passwordLabel.setFont(passwordLabel.getFont().deriveFont(15.0f));
        panel.add(passwordLabel);

        changeButton = new JButton("Change");
        changeButton.setBounds(5,145,150,50);
        changeButton.addActionListener(this);
        panel.add(changeButton);

        changeMessageButton = new JButton("Change email message");
        changeMessageButton.setBounds(5,200,200,50);
        changeMessageButton.addActionListener(this);
        panel.add(changeMessageButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(680,405,100,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource()==cancelButton){
            new MainGUI();
            frame.dispose();
```

```
        }
        else if(e.getSource()==changeButton){
            new ChangeEmailGUI();
            frame.dispose();
        }
        else if(e.getSource()==changeMessageButton){
            new ChangeMessageGUI();
            frame.dispose();
        }

    }
}
```

-ChangeEmailGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeEmailGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel emailLabel, passwordLabel;
    private static JTextField emailText;
    private static JPasswordField passwordText;
    private static JButton changeButton, cancelButton;
    Settings settingsObject = new Settings();

    public ChangeEmailGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Change Email");
        frame.add(panel);

        panel.setLayout(null);

        emailLabel = new JLabel("Email:");
        emailLabel.setBounds(5,5,150,50);
        emailLabel.setFont(emailLabel.getFont().deriveFont(15.0f));
        panel.add(emailLabel);

        emailText = new JTextField(20);
        emailText.setBounds(5,60,300,25);
        panel.add(emailText);

        passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(5,100,150,50);
```

```
        passwordLabel.setFont(passwordLabel.getFont().deriveFont(15.0f));
        panel.add(passwordLabel);

        passwordText = new JPasswordField();
        passwordText.setBounds(5,165,300,25);
        panel.add(passwordText);

        changeButton = new JButton("Change");
        changeButton.setBounds(5,200,250,50);
        changeButton.addActionListener(this);
        panel.add(changeButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(260,200,150,50);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == cancelButton){
            new SettingsGUI();
            frame.dispose();
        }
        else if (e.getSource()== changeButton){
            settingsObject.setMyEmail(emailText.getText());
            settingsObject.setPassword(passwordText.getText());
            settingsObject.save();
            new SettingsGUI();
            frame.dispose();

        }

    }
}
```

-ChangeMessageGUI class:

```
package com.company;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChangeMessageGUI implements ActionListener {

    private static JFrame frame;
    private static JPanel panel;
    private static JLabel previousLabel, messageLabel, newLabel;
```

```
    private static JTextArea emailText;
    private static JButton changeButton, cancelButton;
    Settings settingsObject = new Settings();


    public ChangeMessageGUI(){

        frame = new JFrame();
        panel = new JPanel();
        frame.setSize(800,500);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setTitle("Computer Science IA: Change Email Message");
        frame.add(panel);

        panel.setLayout(null);

        previousLabel = new JLabel("Previous:");
        previousLabel.setBounds(5,5,300,50);
        previousLabel.setFont(previousLabel.getFont().deriveFont(25.0f));
        panel.add(previousLabel);

        messageLabel = new JLabel(settingsObject.getMailMessage() );
        messageLabel.setBounds(5,70,700,100);
        messageLabel.setFont(messageLabel.getFont().deriveFont(15.0f));
        panel.add(messageLabel);

        newLabel = new JLabel("New:");
        newLabel.setBounds(5,200,300,50);
        newLabel.setFont(newLabel.getFont().deriveFont(25.0f));
        panel.add(newLabel);

        emailText = new JTextArea();
        emailText.setBounds(5,260,600,200);
        emailText.setLineWrap(true);
        panel.add(emailText);

        changeButton = new JButton("Change");
        changeButton.setBounds(620,280,150,70);
        changeButton.addActionListener(this);
        panel.add(changeButton);

        cancelButton = new JButton("Cancel");
        cancelButton.setBounds(620,370,150,70);
        cancelButton.addActionListener(this);
        panel.add(cancelButton);

        frame.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {
```

```
        if(e.getSource()==changeButton){
            settingsObject.setMailMessage(emailText.getText());
            settingsObject.save();
            new SettingsGUI();
            frame.dispose();
        }
        else if (e.getSource()==cancelButton){
            new SettingsGUI();
            frame.dispose();
        }

    }
}
```