# Emailer

CPSC 24500 Homework #7  
Due November 3, 2023 at 6pm  
46 points

In this assignment, we will create a basic email application. You will implement it as an object-oriented program with three classes:

Email: a class that represents an individual email communication. An email has a list of recipients, a subject, a body, and a status that indicates whether it has been sent.

EmailPrinter: a class that can print a list of emails to the screen or to a file.

EmailApp: the main application that houses the public static void main function as well as other supporting functions that help the program achieve various tasks.

Together, these classes work together to create a menu-driven tool a user can employ to write, list, and send emails. The program must operate as shown here:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Emailer Version 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 1

Enter recipients' email: klumpra@lewisu.edu raymond.klump@gmail.com

Enter subject: This is a test

Enter body: This is a test of your email system. How do you like it?

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 1

Enter recipients' email: kklump@smip.org mom2cle@gmail.com

Enter subject: This is another test

Enter body: I hope the test is successful. Don't you?

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 2

Here are your emails:

To: klumpra@lewisu.edu,raymond.klump@gmail.com

Subject: This is a test

Status: not sent

Body:

This is a test of your email system. How do you like it?

To: kklump@smip.org,mom2cle@gmail.com

Subject: This is another test

Status: not sent

Body:

I hope the test is successful. Don't you?

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 3

All emails have been sent.

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 2

Here are your emails:

To: klumpra@lewisu.edu,raymond.klump@gmail.com

Subject: This is a test

Status: sent

Body:

This is a test of your email system. How do you like it?

To: kklump@smip.org,mom2cle@gmail.com

Subject: This is another test

Status: sent

Body:

I hope the test is successful. Don't you?

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 4

Enter the name of the file to save: c:\temp\emails.txt

Emails were saved successfully.

Here are your choices:

1. Write email

2. List emails

3. Send emails

4. Save emails to file

5. Exit

Enter the number of your choice: 5

Thank you for using this program.

Notice how after the user chooses option 3, the emails switch from a status of “not sent” to “sent”.

Also, regarding option #4, the file the program writes will list all the details about each email on one tab-delimited line. In other words, the file will look like this.



Note that if the user enters an invalid choice, either because they type a string instead of a number or a number that doesn’t correspond to a valid option, the program will correct them by saying they must enter a valid integer.

Here is how the program will be graded.

|  |  |
| --- | --- |
| Points | Criterion |
| 10 | You create an Email class in a file called Email.java. The class specifies storage for a list of recipients, a subject, a body, and a send status. It has getters and setters, a default and a non-default constructor, a toString function that enables us to easily present the details of an email to the screen, and another String-returning function that allows us to easily write a one-line tab-delimited summary of each email. The Email class must also have a send() function that sets the send status to true instead of false. |
| 6 | You create an EmailPrinter library-style class (meaning it has just static functions) in a file called EmailPrinter.java. The EmailPrinter class will have two functions, one that prints an ArrayList of Email objects to the screen and another that prints an ArrayList of Email objects to a file. The one that prints to a file must return true if successful and false if unsuccessful. |
| 6 | You create a class called EmailApp in a file called EmailApp.java that prints the welcome banner, repeatedly prints the menu until the user decides to quit, and responds to the user’s selection, including warning the user when they enter an invalid option. |
| 4 | The program successfully interviews the user for details about an email and then creates an Email object and adds it to an ArrayList of them. |
| 4 | The program prints the details of all the emails as shown in the sample output. It calls EmailPrinter’s function for writing to the screen to accomplish this. |
| 4 | The program tells all the Email objects to send() so that their send status switches to true when the user elects to send them. |
| 4 | When the user chooses to write to a file, the program asks the user for the name of the file. It then tells EmailPrinter to print the Email objects to the named file. If an error occurs, the program will indicate the failure; otherwise it will say that the emails were saved successfully. |
| 1 | The program begins with a centered welcome banner |
| 3 | The program contains Javadoc comments that documents the functions of all three classes. |
| 1 | The program ends with a “Thank you for using this program.” Message. |
| 3 | You submit your solution by posting your GitHub link on Blackboard. Your code must have been posted to GitHub using git’s commit and push tools, not through a file upload. You must make at least one commitment to your local repository no later than 10/27 at 11:59pm or you will get no credit for this item. |

In total, this program is worth 46 points.

If your program crashes, you will be penalized 9.2 points (20%)

If your program fails to compile, you will be penalized 23 points (50%)

If you copy any part of your code from someone or somewhere else, including from an AI agent, you will earn -46 point out of 46.

As always, extra credit will be awarded to *complete* solutions turned in prior to the due date. If you turn it in *complete* by October 30 at 11:59pm, you will be award 10% extra credit. If you turn it in *complete* by October 31 at 11:59pm, you will be awarded 5% extra credit.

As always, please let me know when you need help.