Globals.java
include the following:

declare global int constant END_OF_MESSAGE with a value of -1
declare global char constant BLANK with a value of 32 (ascii code of blank)

Record.java class
Record class imports java.io.*

notes
This class will manage the most fundamental reading and writing of information to the _messages.dat file

One record is a fixed number of bytes. This class will contain the necessary variables and methods to be able to:

1. Read one record from the _messages.dat file and store it in an array of bytes
2. Write an array of bytes as one record to the _messages.dat file

The size of the record is determined by the contant RECORD_LEN as specified in the Globals class

private variables
declare and instantiate an array of byte called **data,** of size RECORD_DATA_LEN
declare an integer called **next** and initialize to END_OF_MESSAGE

constructors
When using loops, lengths, etc. be sure to always use symbolic constants. Do not use actual numbers. They are used only in the declaration of the constants in the Globals class.

The default constructor initialzes the **data** array and the **next** variable. We define the default constructor to do the following:

1) initialize every element of the array **data** to BLANK cast to byte
2) initialize **next** to END_OF_MESSAGE

A second constructor has parameters (String s, int nextRecord) and contains a single call **setData(s, nextRecord)** This method setData() is defined below.

access methods
Two of these methods convert the array of bytes to a String and viceversa. The third method returns the value of the variable **next**

public String getData()
create a String composed of all the bytes of **data**; each byte must be cast to a character before being concatenated to the final String. Then the String is returned.

public void setData(String txt, int nextRecord)
every byte of **data** is filled with a byte cast of every character of the parameter **txt**; if there are any left over bytes in **data**, they are filled with a byte cast of BLANK. For example: If **txt** has the value **Hello cat** then the **data** array becomes

where the **data** array has a size of RECORD_DATA_LEN as defined above

Then, the private variable **next** is set to the parameter **nextRecord**

public int getNext()
returns **next**

EmailServer.java

notes
Code from our previous day has been deleted or commented so we can test the Record class

There are two tests here:
Test 1. To check that the default constructor of the Record class is working properly
Test 2. To check that the second constructor is working properly since it makes calls to the access methods.

test 1

```
main() method {
     declare and instantiate an object of the Record class with the name record using the
     default constructor.

     create a loop that runs through the entire data array of the Record class and check
     that each element of the array contains a blank (ascii 32). You may have to write a
     temporary access method for this.

     check that the private variable next contains END_OF_MESSAGE

     make a call to getData() and check that the returned string is made up of
     RECORD_DATA_LEN blanks


     /* Code that opens/closes the messages file
     open/create the messages file using the method of the class FileIO
     if all goes well
          close the messages file using the method of the class FileIO
     else
          print an error saying that the file could not be opened */
}
```

test 2

```
main() method {
     declare and instantiate an object of the Record class with the name record using the
     second constructor. As testing parameters use the words "Hello Sun!" for the s
     parameter and the value 6 for the nextRecord parameter

     create a loop that runs through the entire data array of the Record class and check
     that each element of the array contains the bytes

     H
     e
     l
     l
     o

     S
     u
     n
     !
     plus another 70 blanks. You may have to write a temporary access method for this.

     check that the private variable next contains the value 6

     make a call to getData() and check that the returned string contains the original
     string "Hello Sun!" plus an extra 70 blanks

     /* Code that opens/closes the messages file
     open/create the messages file using the method of the class FileIO
     if all goes well
          close the messages file using the method of the class FileIO
     else
          print an error saying that the file could not be opened */
}
```