What abUTA You?

Due Tuesday, February 4 at 8 a.m.

CSE 1325 - Sprint #1 of 4 - Homework #3 - Rev 0

Assignment Overview

Everyone seems to be unhappy with their social media apps these days - Facebook, YouTube, WhatsApp, Instagram, the almost-late (in the US) TikTok, and X (née Twitter). What should any self-respecting geek do? Write our own, of course!

This will be a 4-sprint, 4-week project (with the first exam in the middle - sorry) that you can add to your growing resume, with emphasis on writing a user interface, file I/O, inheritance, and polymorphism (all coming soon to a lecture near you). We'll have a final, stand-alone Java assignment to practice threading before Exam #2. Implementation guidance will be provided each week to help you with your implementation, and you may have a few checkpoints where you may switch to the professor's suggested solution if you lose your footing.

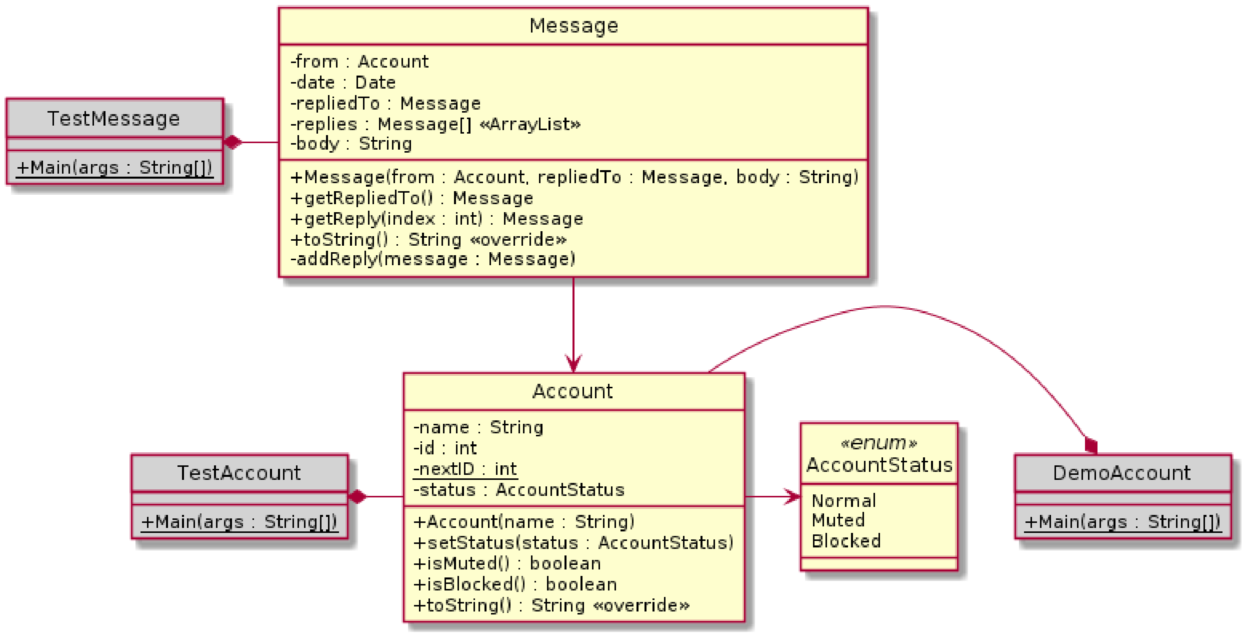
For the first sprint (P03), one or more regression tests (see Lecture 5) will be used to verify that your implementation of the class diagram is correct. We'll add packages with JavaDoc, inheritance, and ArrayList in sprint 2 (P04), then after Exam #1 add a menu-driven interface (MDI) in sprint 3 (P05). We'll wrap up by adding file I/O (save and reload the data) and polymorphism in sprint 4 (P06). We'll aim to provide additional opportunities for bonus points along the way.

Sprint #1

Consider this class diagram on the next page. Which class or enum would you write first? Which second? Which would you write last? Why?

The more darkly shaded classes - the TestAccount and TestMessage regression tests and the DemoAccount demonstrator - will be provided for you at cse1325-prof/P03/baseline to help you verify your classes have been written correctly. Just copy them over and use them to test your code. Note that they can be *very* picky about getting the right number of spaces!

You may ONLY add the public members shown on the diagram to your classes. No additional public members are permitted. Don't duplicate any class responsibilities in your other methods, including main!

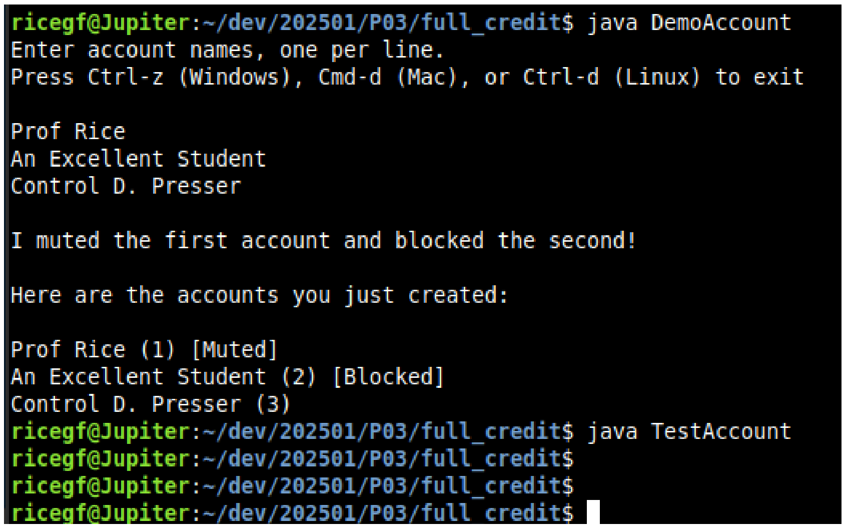


In your GitHub-managed working directory **cse1325/P03/full_credit**, write enum `AccountStatus` that enumerates the 3 states of an account. This will be used in later sprints: `Normal` will be an account with no restrictions, `Mute` will be able to read any posts but their own posts won't be visible to others, and `Block` will not be able to log in at all.

Then write class `Account` as shown in the UML class diagram. Remember that underlined members are static. The `id` for the `Account` of the first user should be 1, NOT 0. The executable members should behave as follows:

- The constructor sets the `name` field to the corresponding parameter UNLESS the corresponding parameter is zero-length, in which case throw an IllegalArgumentException instead. `id` is set to static field `nextID`, which is then incremented. `status` is set to `Normal`.
- Method `setStatus``simply sets the ``status` field to the parameter.
- Method `isMuted` returns `true` if `status` is not `Normal`. That is, an account is muted if its status is EITHER muted or blocked.
- Method `isBlocked` returns `true` if and only if `status` is `Blocked`.
- Method `toString` should return the name followed by the ID in parentheses. If `status` is not `Normal`, append the `status` in square brackets. For example, `toString` might return "Julia Childs (5)" or "Dr. No (13) [Blocked]".

The provided `DemoAccount` class should compile with your class and allow you to create and print Account objects and see if their `toString` method returns what is expected. If written correctly, your `Account` and `AccountStatus` classes should compile with the provided `TestAccount` class and pass when run with `java TestAccount`. Remember, "pass" means "no output at all"!



```
ricegf@Jupiter:~/dev/202501/P03/full_credit$ java DemoAccount
Enter account names, one per line.
Press Ctrl-z (Windows), Cmd-d (Mac), or Ctrl-d (Linux) to exit

Prof Rice
An Excellent Student
Control D. Presser

I muted the first account and blocked the second!

Here are the accounts you just created:

Prof Rice (1) [Muted]
An Excellent Student (2) [Blocked]
Control D. Presser (3)
ricegf@Jupiter:~/dev/202501/P03/full_credit$ java TestAccount
ricegf@Jupiter:~/dev/202501/P03/full_credit$
ricegf@Jupiter:~/dev/202501/P03/full_credit$
ricegf@Jupiter:~/dev/202501/P03/full_credit$
```

Finally, write your `Message` class as shown in the diagram. (We wrote `Account` first, because `Message` depends on `Account`!) The behavior of the executable members should be:

- The constructor sets the `from`, `repliedTo`, and `body` fields to the corresponding parameter. For new messages, `repliedTo` will be `null`, but if NOT `null`, call its `addReply` method, passing this message as the parameter to complete the double-linked list. Also construct the `replies` field, which is an ArrayList (we'll discuss this on Thursday).
- Method `getRepliedTo``simply returns the ``repliedTo` field.
- Method `getReply` returns the message corresponding to `index` in the `replies` ArrayList, or if the index is out of range, return `null`.
- Method `toString` should format the message like this. Names in angle brackets are field names. If `repliedTo` is null, omit the "In reply to:" line. If `replies` is empty, omit the "Replies:" line.

```
Date: <date>
From: <from>
In reply to: <repliedTo.from>
Replies: <.from for each member of field replies>
<body>
```

- *Private* method `addReply` should simply add its parameter to field `replies`. (ArrayList has an `add` method for exactly this, as you'll see in Lecture 06.)

You MUST provide a working build.xml file so that the `ant` tool works. A working build.xml is provided at `cse1325-prof/P01/full_credit/build.xml` or in any of the lecture code.

You MUST include at least 3 substantial commits of your evolving work. A "one and done" commit, even with trivial additional commits just to get to 3, will incur *substantial* grade penalties.

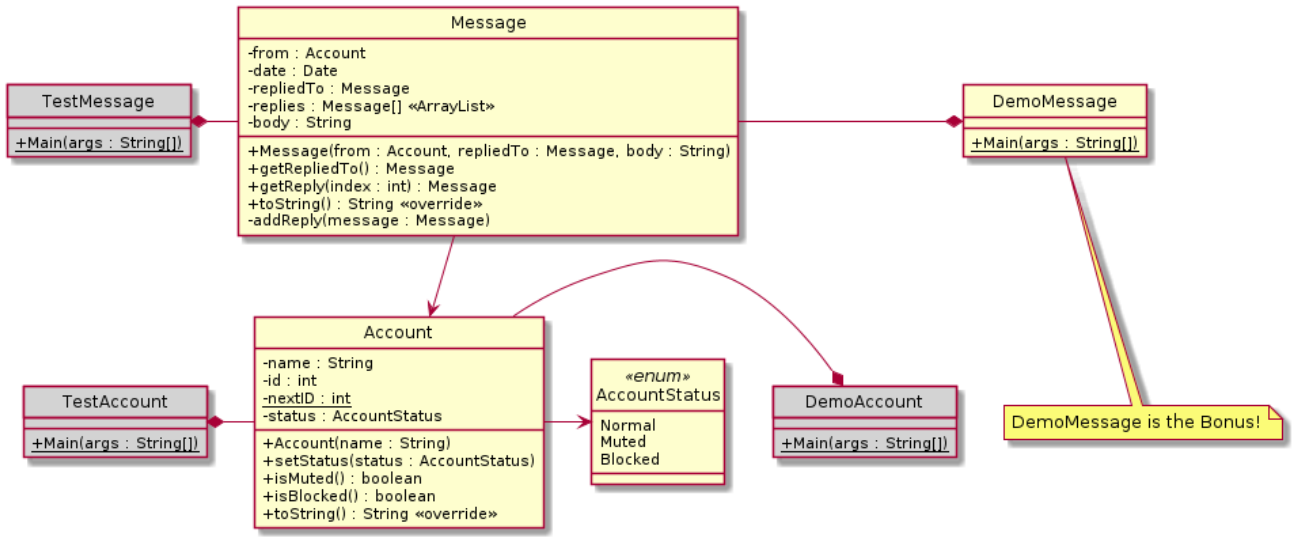
Practice good version control!

Bonus

We provided you a `DemoAccount` class with a main to test out your `Account` class. For the bonus, write a simple `DemoMessage` that does the same with the `Message` class. Your demo may be interactive or hard-coded, or a mixure of each (for example, hard-coded Account objects but allow entering message text).

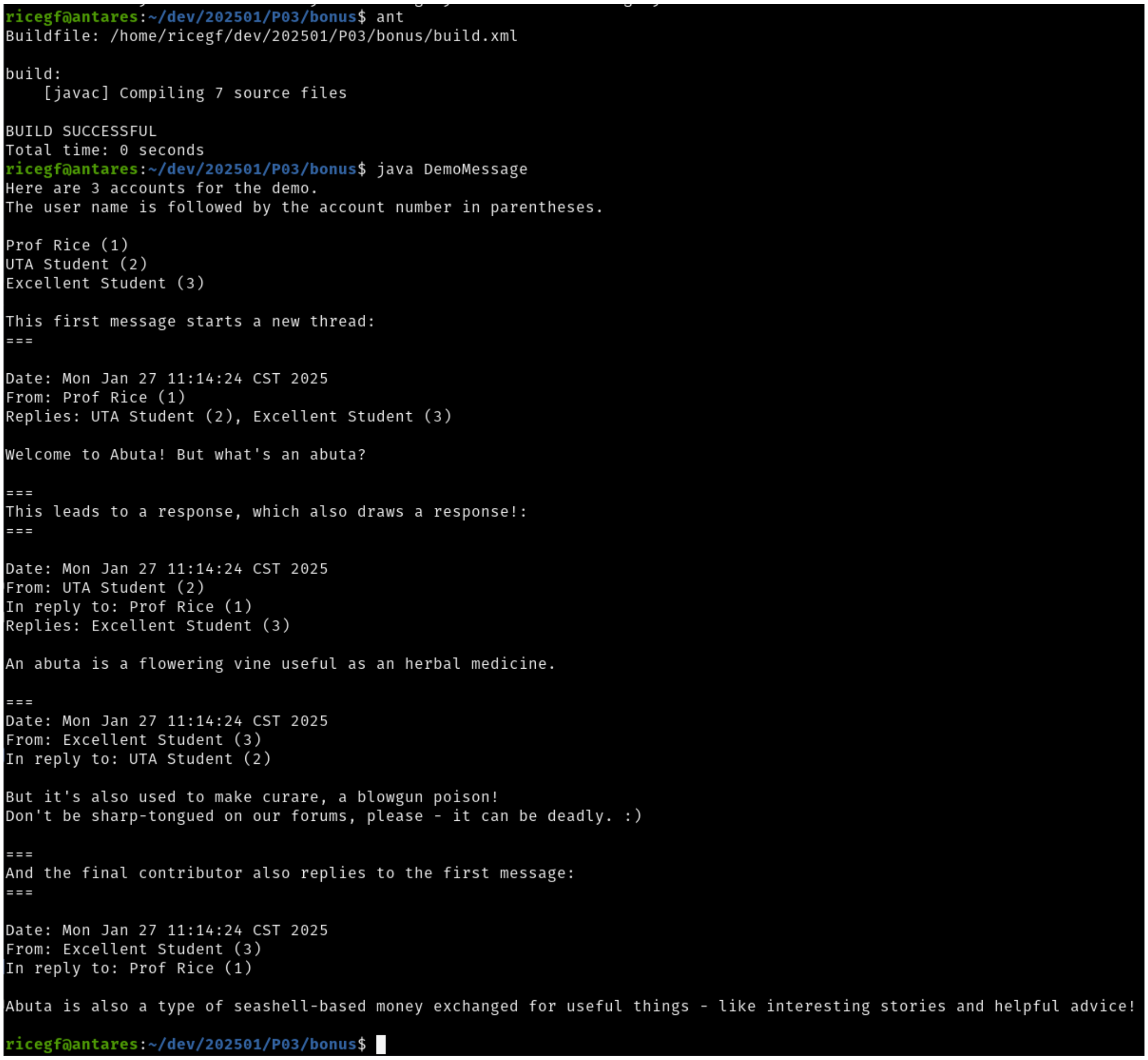
The suggested solution for the bonus level shown below literally includes only 18 lines of code. All the heavy lifting is done by the two classes you wrote!

Here's the class diagram for the bonus.



Include at least 3 substantial commits of your evolving Bonus work!

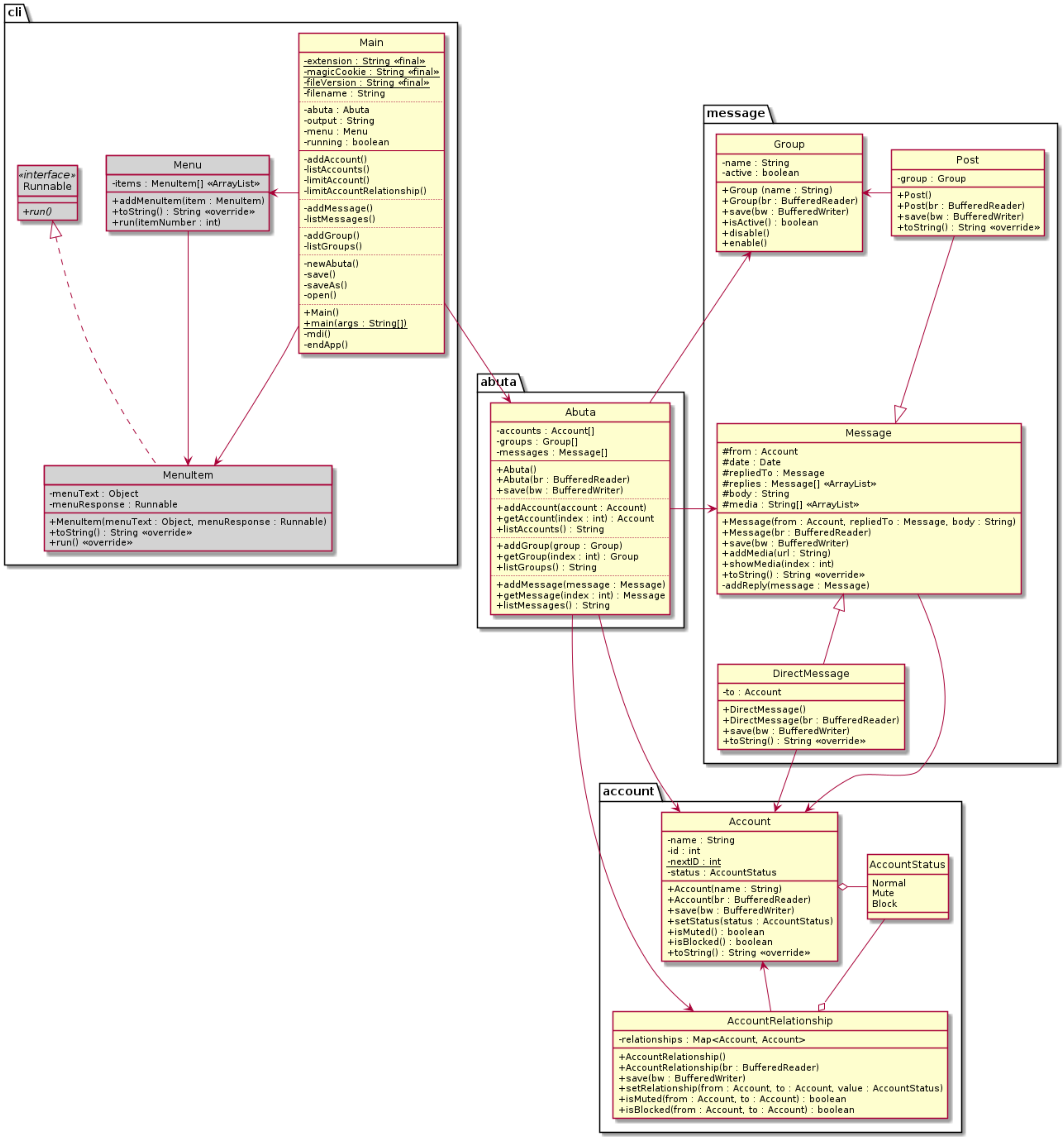
Compiling and running your program may now look something like the screenshot on the next page (but NOT exactly like - yours may vary substantially!).



```
ricegf@antares:~/dev/202501/P03/bonus$ ant
Buildfile: /home/ricegf/dev/202501/P03/bonus/build.xml

build:
    [javac] Compiling 7 source files

BUILD SUCCESSFUL
Total time: 0 seconds
ricegf@antares:~/dev/202501/P03/bonus$ java DemoMessage
Here are 3 accounts for the demo.
The user name is followed by the account number in parentheses.

Prof Rice (1)
UTA Student (2)
Excellent Student (3)

This first message starts a new thread:
===

Date: Mon Jan 27 11:14:24 CST 2025
From: Prof Rice (1)
Replies: UTA Student (2), Excellent Student (3)

Welcome to Abuta! But what's an abuta?

===
This leads to a response, which also draws a response!:
===

Date: Mon Jan 27 11:14:24 CST 2025
From: UTA Student (2)
In reply to: Prof Rice (1)
Replies: Excellent Student (3)

An abuta is a flowering vine useful as an herbal medicine.

===
Date: Mon Jan 27 11:14:24 CST 2025
From: Excellent Student (3)
In reply to: UTA Student (2)

But it's also used to make curare, a blowgun poison!
Don't be sharp-tongued on our forums, please - it can be deadly. :)

===
And the final contributor also replies to the first message:
===

Date: Mon Jan 27 11:14:24 CST 2025
From: Excellent Student (3)
In reply to: Prof Rice (1)

Abuta is also a type of seashell-based money exchanged for useful things - like interesting stories and helpful advice!

ricegf@antares:~/dev/202501/P03/bonus$ 
```

Where We're Going

We'll be implementing much of the larger class diagram on the next page over the remaining sprints. Note that this will still change substantially before the end of the project.


cli
Main
-extension : String «final»
-magicCookie : String «final»
-fileVersion : String «final»
-filename : String
-abuta : Abuta
-output : String
-menu : Menu
-running : boolean
-addAccount()
-listAccounts()
-limitAccount()
-limitAccountRelationship()
-addMessage()
-listMessages()
-addGroup()
-listGroups()
-newAbuta()
-save()
-saveAs()
-open()
+Main()
+main(args : String[])
-mdi()
-endApp()
«interface»
Runnable
+run()
Menu
-items : MenuItem[] «ArrayList»
+addMenuItem(item : MenuItem)
+toString() : String «override»
+run(itemNumber : int)
MenuItem
-menuText : Object
-menuResponse : Runnable
+MenuItem(menuText : Object, menuResponse : Runnable)
+toString() : String «override»
+run() «override»
abuta
Abuta
-accounts : Account[]
-groups : Group[]
-messages : Message[]
+Abuta()
+Abuta(br : BufferedReader)
+save(bw : BufferedWriter)
+addAccount(account : Account)
+getAccount(index : int) : Account
+listAccounts() : String
+addGroup(group : Group)
+getGroup(index : int) : Group
+listGroups() : String
+addMessage(message : Message)
+getMessage(index : int) : Message
+listMessages() : String
message
Group
-name : String
-active : boolean
+Group (name : String)
+Group(br : BufferedReader)
+save(bw : BufferedWriter)
+isActive() : boolean
+disable()
+enable()
Post
-group : Group
+Post()
+Post(br : BufferedReader)
+save(bw : BufferedWriter)
+toString() : String «override»
Message
#from : Account
#date : Date
#repliedTo : Message
#replies : Message[] «ArrayList»
#body : String
#media : String[] «ArrayList»
+Message(from : Account, repliedTo : Message, body : String)
+Message(br : BufferedReader)
+save(bw : BufferedWriter)
+addMedia(url : String)
+showMedia(index : int)
+toString() : String «override»
-addReply(message : Message)
DirectMessage
-to : Account
+DirectMessage()
+DirectMessage(br : BufferedReader)
+save(bw : BufferedWriter)
+toString() : String «override»
account
Account
-name : String
-id : int
-nextID : int
-status : AccountStatus
+Account(name : String)
+Account(br : BufferedReader)
+save(bw : BufferedWriter)
+setStatus(status : AccountStatus)
+isMuted() : boolean
+isBlocked() : boolean
+toString() : String «override»
AccountStatus
Normal
Mute
Block
AccountRelationship
-relationships : Map<Account, Account>
+AccountRelationship()
+AccountRelationship(br : BufferedReader)
+save(bw : BufferedWriter)
+setRelationship(from : Account, to : Account, value : AccountStatus)
+isMuted(from : Account, to : Account) : boolean
+isBlocked(from : Account, to : Account) : boolean