What abUTA You?

Due Tuesday, February 25 at 8 a.m.

CSE 1325 - Sprint #3 of 4 - Homework #5 - 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 is sprint 3 of 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 third sprint (P05), we'll add a Menu-Driven Interface (mdi). Unlike your C courses, where you hard-coded your menu and used a switch to code the behavior of each menu item, we'll take the object-oriented approach as discussed in Lecture 10.

If your first two sprints left something to be desired, contact the professor and ask about baselining the suggested solution. While we *prefer* that you make this project entirely your own, we want to ensure that you're able to complete it. Let's talk!

Sprint #3

Sprint Overview

For this sprint we will add a single new class, `Abuta` in package `abuta`. This will include our main method, which instances `Abuta` and then runs its `mdi` method.

The `Abuta` constructor adds some predefined `Account` and `Group` objects to the `accounts` and `groups` fields, instances the root message for field `message`, and creates the `Menu` object and adds four `MenuItem` objects to it.

The `mdi` method is the main loop prints the title, menu, output, and message, gets an integer from the user representing the desired action on the menu, and calls the `menu.run` method with that integer. The `Menu` class manages both the text of each menu item and the method to call when that text is selected.

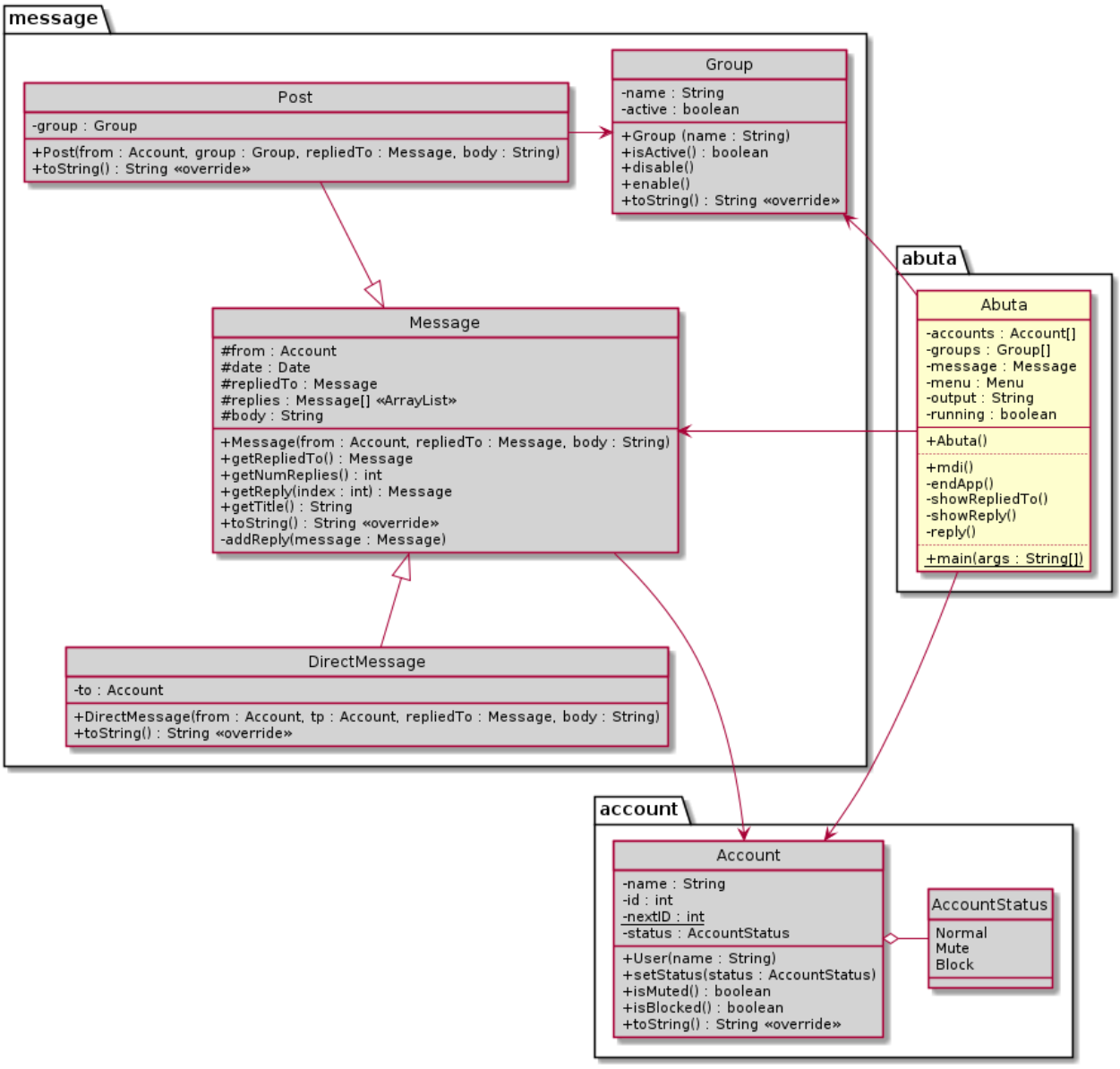
The menu should offer 4 behaviors: Exit (method `endApp`), show reply (`showReply`), show replied to (`showRepliedTo`), and add reply (`reply`). Each method is responsible for updating the fields to implement those behaviors for our social media app.

Overview of the New Abuta Class

Consider this class diagram on the next page. The more darkly shaded classes are *mostly* unchanged from the last sprint, while the more lightly colored classes are new this sprint.

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!

Class `Abuta` (in package `abuta`) will include our main method, which simply instances Abuta and calls its `mdi` method. Method `mdi` is the main loop, which in the suggested solution is only 8 statements long. How? We rely on classes to encapsulate the management of our menu items and associated behaviors!



A Small Addition to Message

Class `Message` needs a slight modification. Instead of just listing the *account names* for each of the `replies` to the Message object, we also need its *index*, like this:

```
Group: Power
Date: Mon Feb 17 19:10:46 CST 2025
From: Gandalf the Grey (1)
Replies: [0] Frodo Baggins, [1] Arwen, [2] Balrog

Welcome to Middle Earth! The journey doesn't end here; it is only the beginning!
```

This provides the "menu" for selecting a reply as part of the message itself. This should change only a line or two of your `Message` class (or, if you didn't follow DRY, of both your `Post` and `DirectMessage` classes).

No other changes to your Sprint 2 code should be required.

Creating Abuta

The `Abuta` class stores our `Account` objects in field `accounts` and our `Group` objects in field `groups`. Both of these are ArrayLists - make the field of interface type `List` but assign an `ArrayList` object to it. The current message being viewed is referenced by field `message`.

In addition, we have field `menu` that holds our `Menu` object, the `output` field which collects messages to show the user at the next main menu screen, and the `running` boolean that will be set to false when `endApp` is called by `menu`.

To pull `Abuta` together, then, we'll need our `Menu` and `MenuItem` classes developed in Lecture 10 as shown on the next page. (It's common to have a set of class diagrams, each emphasizing a different aspect of a design. In this project, we have two.)

The `menu` package is available at cse1325-prof/P05/menu. Feel free to use it as-is (hint: It is fully documented in Javadoc), although you may write your own Menu and MenuItem class if you prefer.

WARNING: You may NOT just write a monolithic main with a hard-coded menu and a switch statement for the behaviors. The point of this assignment is to use object-oriented techniques to implement a menu-driven interface. You will receive few points if you don't do this!

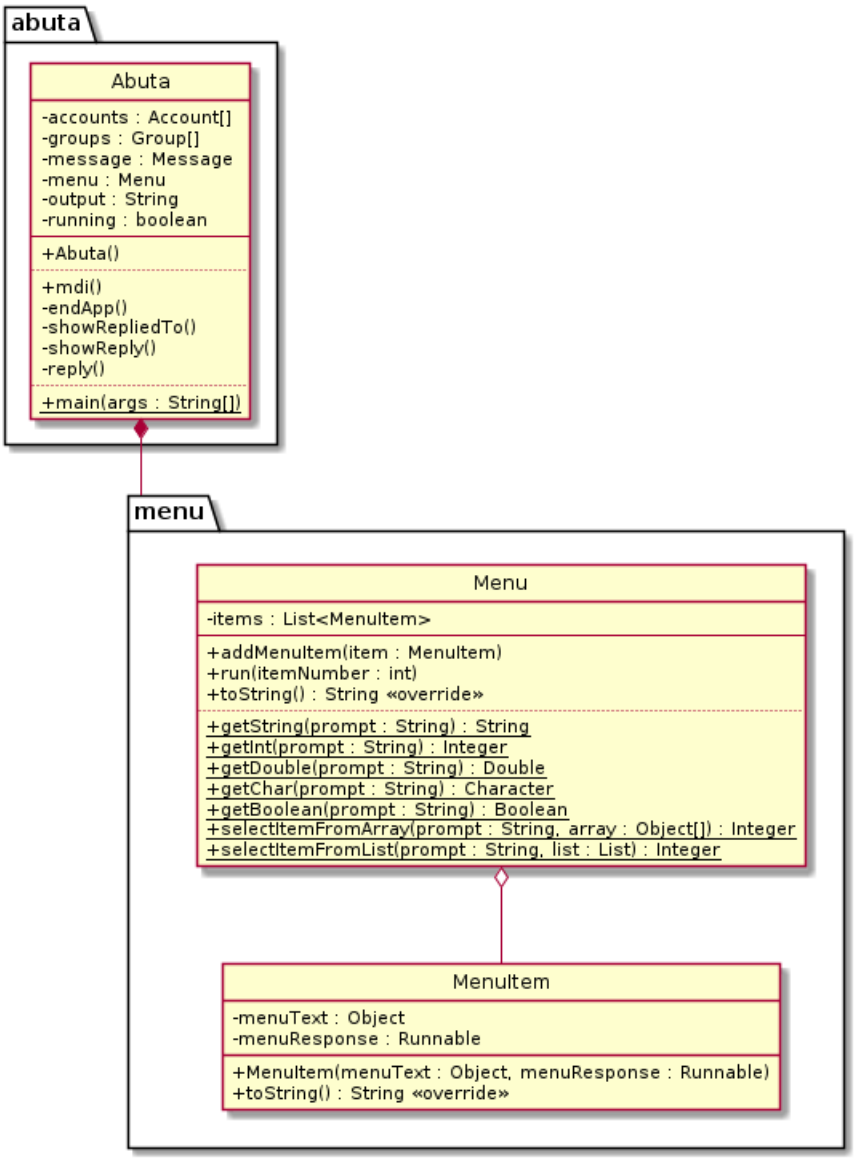
Main Method

The main method is simplicity itself: Instance class `Abuta` and call its `mdi` method. This is very common.

Abuta Constructor

- Construct the ArrayList instance for field `accounts` and then add at least 5 `Account` instances to it.
- Construct the ArrayList instance for field `groups` and add at least 5 `Group` instances to it.
- Construct a new `Post` using an element from `accounts` and from `groups`, setting `repliedTo` to null and with any text you like, and reference it from field `message`. This is the start of your tree of messages.

- Construct a new `Menu` and add 4 `MenuItem` objects to it (method `addMenuItem`):
    - An exit option that invokes method `endApp` when selected.
    - A show reply option that invokes method `showReply` when selected.
    - An show replied to option that invokes method `showRepliedTo` when selected.
    - An add reply option that invokes method `reply` when selected.



mdi Method

This is your main loop, in which you will

- Print the title, menu (from the `menu` field), output (from the `output` field),
- Clear `output`,
- Get the user's selection (an integer), and
- Invoke `Menu`'s `run` method with the user's selection.

Be careful to handle any exceptions that may occur. Our programs should NEVER unexpectedly abort!

endApp Method

This method (called from the `Menu` object) simply sets `running` to false. Your main loop should exit when `running` becomes false.

showRepliedTo Method

This method switches the active message (referenced by the `message` field) to the current message's `repliedTo` field.

However, the first message created is not in reply to any message, so in that case, use the `output` field to report that the current message was not in reply to any other message and do NOT change the `message` field.

showReply Method

This method switches to one of the replies to the active message (referenced by the `message` field). You can determine how many replies the active message has using its `getNumReplies` method.

- If 0, use the `output` field to report that the current message has no replies. Do NOT change field `message`!
- If 1, change the `message` field to the reply.
- If 2 or more, ask the user to which reply Abuta should switch. (Hint: The `Menu` class has a static `getInt` method that may help!) Only switch messages if the user's response is valid.

reply Method

This is the crux of this week's sprint - allow the user to write replies to existing messages. Class `Menu` has several static methods that may help.

- `getString` allows you to read a line of text from the user. Similarly, `getChar` allows you to read a single character.
- `selectItemFromList` handles listing the items in a list (such as an ArrayList) with their associated indices, getting an integer from the user, and returning the user-provided index.

These static methods should make it practical to ask the user if they want a `Post` or `DirectMessage`, specify the Author, Group or recipient Account, and the body of the message. The `repliedTo` field will always be the active message referenced by the `message` field, of course.

As implemented last week, when you instance the reply message, it should add itself to the list of `replies` in the message to which you are replying, completing the double-linking of our messages and allowing the user to navigate upward and downward through the message tree.

Bonus

You have two options for bonus points, for +5 each.

Add Account

Add a `MenuItem` with associated method to the `Menu` to enable the user to enter a name with which to create a new `Account` object, and append it to the `accounts` field.

Add Group

Similarly, add a `MenuItem` with associated method to the `Menu` to enable the user to enter a name with which to create a new `Group` object, and append it to the `groups` field.

If you implement both bonuses, you should have 6 rather than 4 menu items in your main menu.

Extreme Bonus

You have several options to flesh out the Abuta application. Pick the one that most interests you.

Account and Group Management

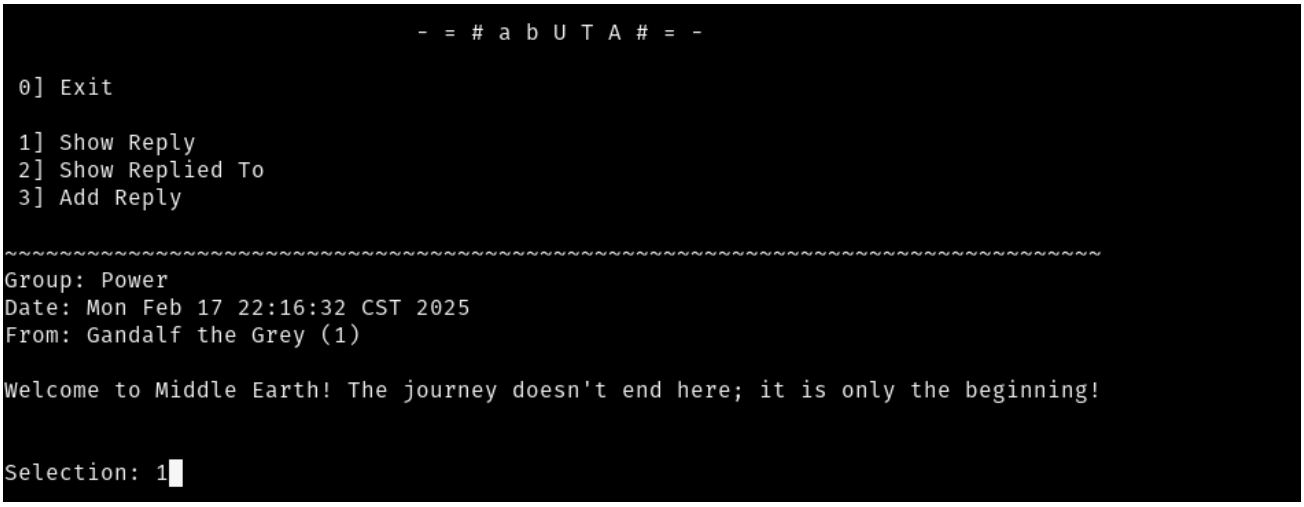
Accounts may be muted or blocked - and, conversely, unmuted and unblocked. Similarly, Groups may be disabled and subsequently enabled. Add menu items to implement these behaviors.

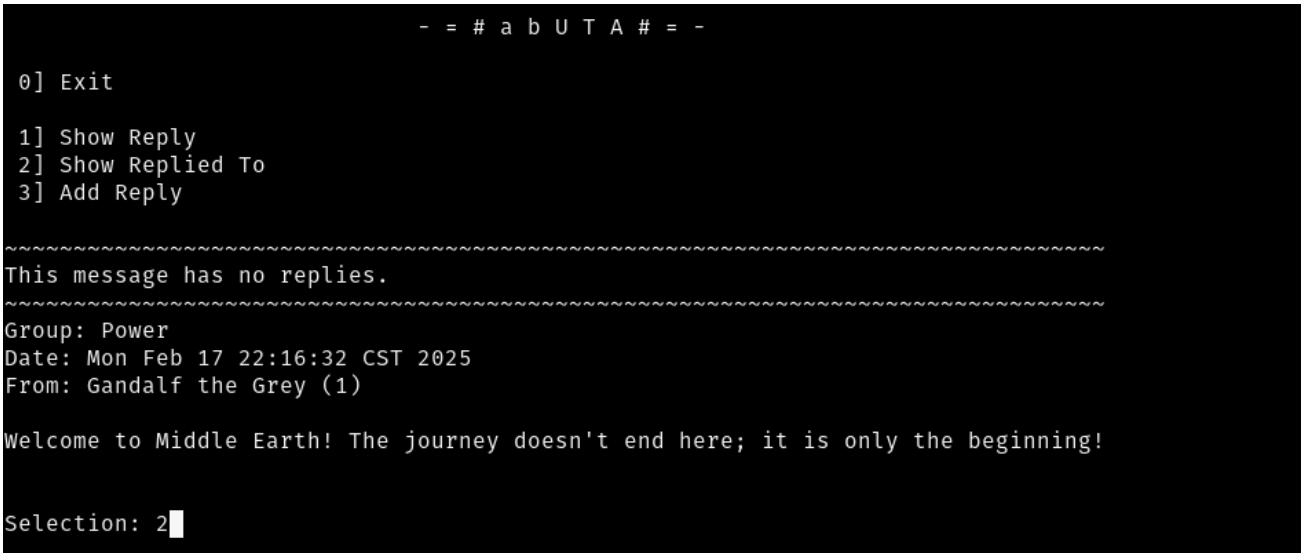
Filter

A bigger challenge is to (for example) navigate only among messages in a specified Group, or only among direct messages for which a specified Account is the author or recipient. This is rather challenging with the current design. Can you implement it?

Screenshots

These series of screenshots illustrate some of the operations of Abuta after Sprint 3. **Your output need NOT match these screenshots** - make this project your own! As long as you implement the requirements, you will receive full credit for your work. If you are unsure at any point, ask the professor. Have I mentioned we love questions yet?



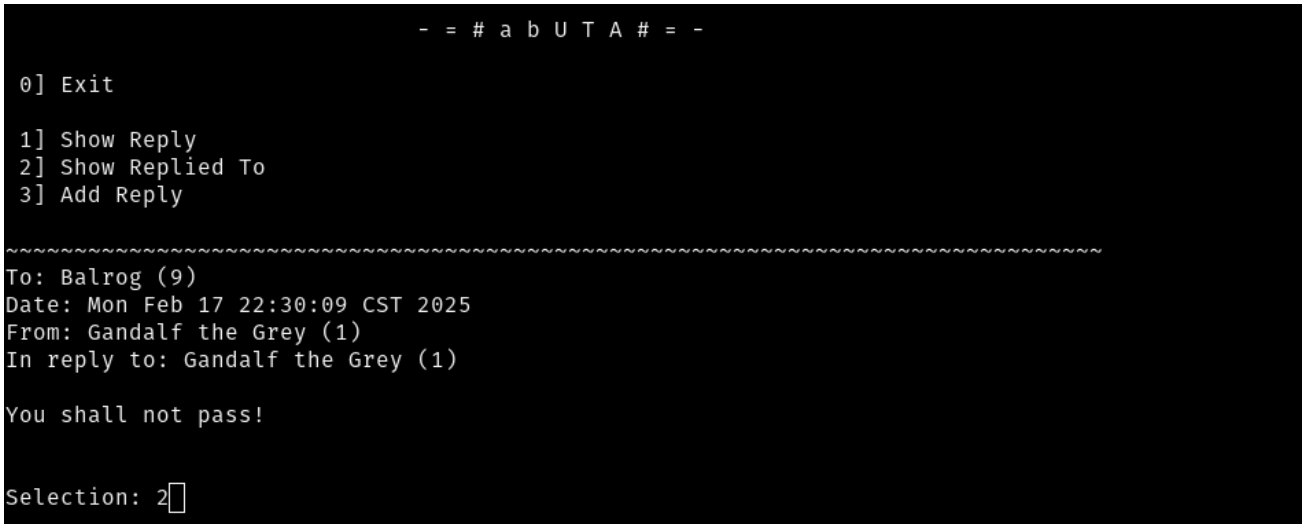


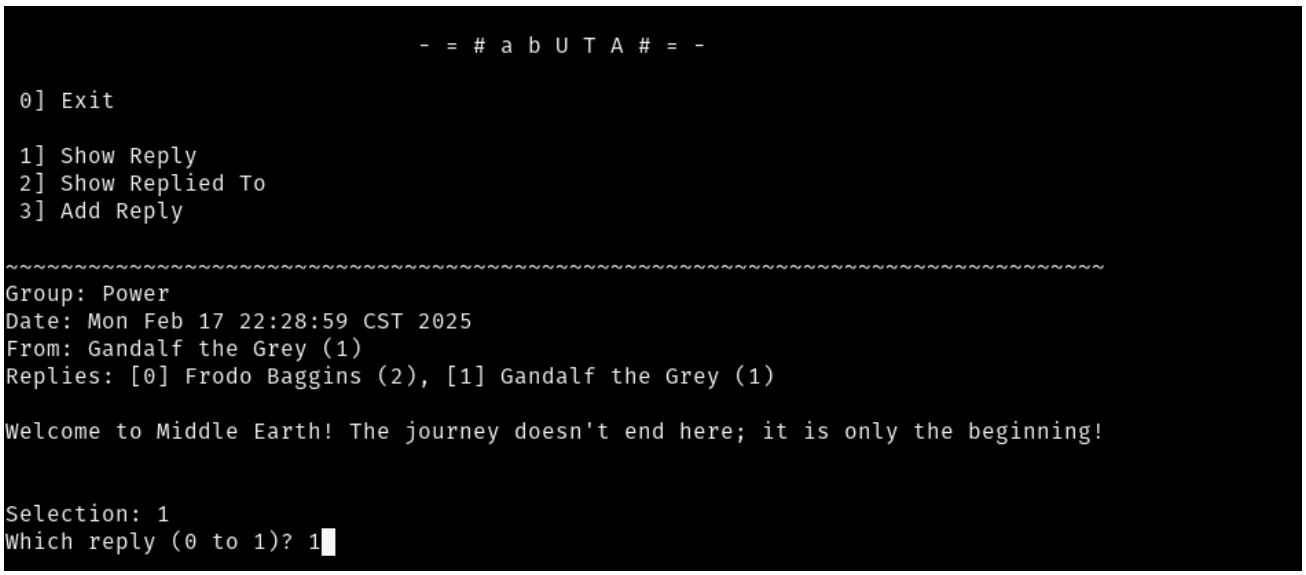
```
                              - = # a b U T A # = -

 0] Exit

 1] Show Reply
 2] Show Replied To
 3] Add Reply

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This is the original message, in reply to no one in Middle Earth.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Group: Power
Date: Mon Feb 17 22:16:32 CST 2025
From: Gandalf the Grey (1)

Welcome to Middle Earth! The journey doesn't end here; it is only the beginning!


Selection: 3
(P)ost or (D)irect Message? p
   0] Gandalf the Grey (1)
   1] Frodo Baggins (2)
   2] Samwise Gamgee (3)
   3] Galadriel (4)
   4] Aragon (5)
   5] Arwen (6)
   6] Legolas (7)
   7] Gimli (8)
   8] Balrog (9)
Author? 1
   0] Power
   1] Friendship
   2] Tenacity
   3] Good vs. Evil
   4] Redemption
   5] Immortality
Group? 1
Post from Frodo Baggins (2) in group Friendship:

Thanks! Happy to be here, although I sometimes feel invisible.
```

```
                            - = # a b U T A # = -

 0] Exit

 1] Show Reply
 2] Show Replied To
 3] Add Reply

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Group: Friendship
Date: Mon Feb 17 22:22:24 CST 2025
From: Frodo Baggins (2)
In reply to: Gandalf the Grey (1)

Thanks! Happy to be here, although I sometimes feel invisible.


Selection: 2
```

```
                            - = # a b U T A # = -

 0] Exit

 1] Show Reply
 2] Show Replied To
 3] Add Reply

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Group: Power
Date: Mon Feb 17 22:16:32 CST 2025
From: Gandalf the Grey (1)
Replies: [0] Frodo Baggins (2)

Welcome to Middle Earth! The journey doesn't end here; it is only the beginning!


Selection: 3
```

```
                              - = # a b U T A # = -

 0] Exit

 1] Show Reply
 2] Show Replied To
 3] Add Reply

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Group: Power
Date: Mon Feb 17 22:16:32 CST 2025
From: Gandalf the Grey (1)
Replies: [0] Frodo Baggins (2)

Welcome to Middle Earth! The journey doesn't end here; it is only the beginning!


Selection: 3
(P)ost or (D)irect Message? d
   0] Gandalf the Grey (1)
   1] Frodo Baggins (2)
   2] Samwise Gamgee (3)
   3] Galadriel (4)
   4] Aragon (5)
   5] Arwen (6)
   6] Legolas (7)
   7] Gimli (8)
   8] Balrog (9)
Author? 0
   0] Gandalf the Grey (1)
   1] Frodo Baggins (2)
   2] Samwise Gamgee (3)
   3] Galadriel (4)
   4] Aragon (5)
   5] Arwen (6)
   6] Legolas (7)
   7] Gimli (8)
   8] Balrog (9)
To? 8
Message from Gandalf the Grey (1) to Balrog (9):

You shall not pass!
```


- = # a b U T A # = -
0] Exit
1] Show Reply
2] Show Replied To
3] Add Reply
To: Balrog (9)
Date: Mon Feb 17 22:30:09 CST 2025
From: Gandalf the Grey (1)
In reply to: Gandalf the Grey (1)
You shall not pass!
Selection: 2


- = # a b U T A # = -
0] Exit
1] Show Reply
2] Show Replied To
3] Add Reply
Group: Power
Date: Mon Feb 17 22:28:59 CST 2025
From: Gandalf the Grey (1)
Replies: [0] Frodo Baggins (2), [1] Gandalf the Grey (1)
Welcome to Middle Earth! The journey doesn't end here; it is only the beginning!
Selection: 1
Which reply (0 to 1)? 1

```
                              - = # a b U T A # = -

 0] Exit

 1] Show Reply
 2] Show Replied To
 3] Add Reply

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
To: Balrog (9)
Date: Mon Feb 17 22:30:09 CST 2025
From: Gandalf the Grey (1)
In reply to: Gandalf the Grey (1)

You shall not pass!


Selection: 0
```