Java Lab9: Memo Creator Fall 2022

This lab is due on Sunday, October 2. Create a project named Lab9. Download MakeANote.java and add it to your project.

Problem statement: The Make-a-Note application will let users create several kinds of documents. Each document will be created according to these specifications:

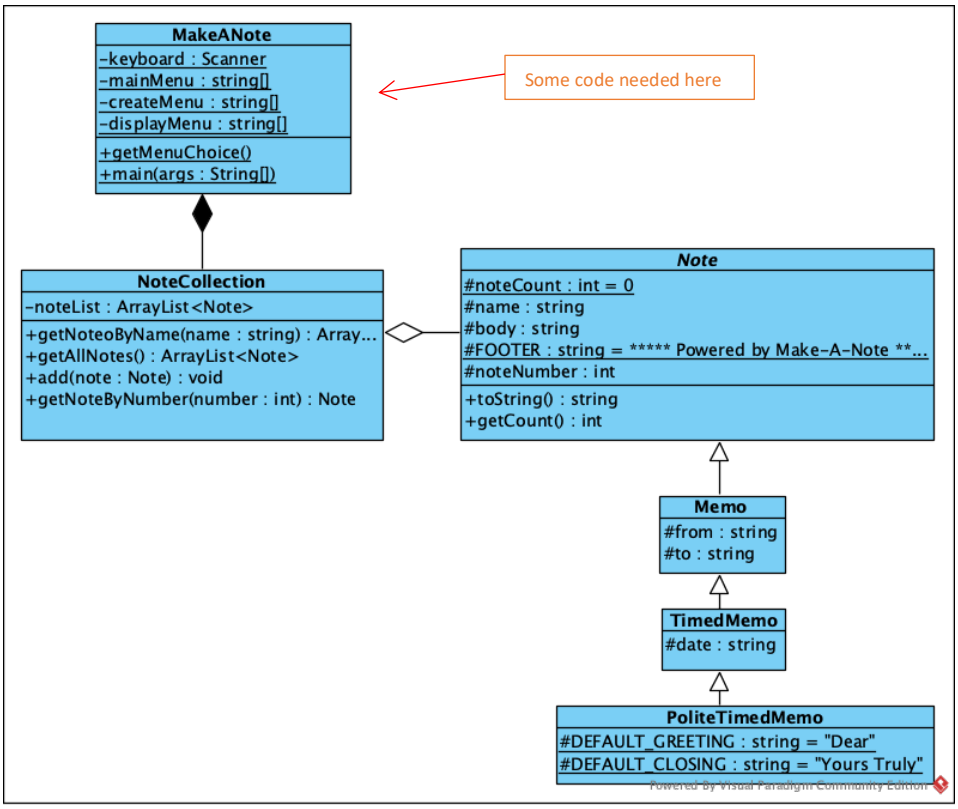
Note: abstract classs that adds one to static int noteCount every time its constructor is called and sets noteNumber to that value; contains the note's name and body; adds the note footer (message at the bottom of every note).
Memo: adds the “From” and “To” fields.
NoteCollection: a collection of your notes.
MakeANote: contains the main program; a generic getMenuChoice( ) method; gets user input. This class is partly coded.

TimedMemo: adds the date of the document’s creation.
PoliteTimed Memo: adds a greeting and a standard closing.
FormattedDate: a helper class for Date handling

Each Memo, TimedMemo, and PoliteTimedMemo will return the note data in the toString() method to be displayed by the main program. These are progressively longer documents, adding the fields described above. See the sample output for an example of each one.

See the table for more details about all the classes.

The main method is in MakeANote. The menu method, getMenuChoice(), will allow a user to create and display multiple documents. As note objects are created, store them in the NoteCollection. The display options are display all notes; display all memos; display all timed memos; display a specific note, chosen by name. See the second table for sample console output.



Class	Member variables	Methods
Note (abstract)	• **name**: String, note name • **body:** String, the text of the note or memo • **noteCount: static** int, count of the number of notes created (i.e. number of times Note constructor is called); initially 0. • **noteNumber:** int • **FOOTER: static String "***** Powered by Make-A-Note *****"**	• **Note( ):** default constructor • **Note**(name, body): overloaded constructor; besides settting name and body, increment noteCount and set noteNumber to noteCount's new value. • **getNoteNumber():** returns noteNumber • **toString**(): return a formatted String containing the name, body, and noteNumber
NoteCollection	• **noteList:** ArrayList<Note>	• **add(Note):** adds a Note object to noteList • **getAllNotes():** return the ArrayList containing all the Notes (of all types) • **getNoteByNumber(int):** return the Note with the given number (or null if not found). • **getNoteByName(String name):** return only those Notes with the given name, or an empty list if not found.
MakeANote	• **static keyboard**: Scanner • **static final**: • **String[] mainMenu** = {"Main Menu", "Create a New Note", "Display existing Note(s)", "Quit"}; • **String[] createMenu** = {"Note Creation", "Create a Memo", "Create a Timed Memo", "Create a Polite Memo", "Return to previous menu"}; • **String[] displayMenu** = {"Display Options", "Display all Notes", "Display all Memos", "Display Note by Number", "Display Notes by Name", "Return to previous menu"};	• **main**(): Starts the program, gets a menu choice from the user, creates the correct type of Note, until the user quits. It will re-use getMenuChoice three times: use the three String[] arrays for each of the three menus • **getMenuChoice**(**String[] menu):** displays a menu, gets a user choice; for now, don't worry about error checking. The zero-th entry is the menu name; the other entries are the menu choices – display these numbered from 1. If you use nextInt(), follow it with nextLine() to clear out the \n. This class is partly coded. See the Console I/O table below for some examples of the menus.
Memo extends Note	• **from**: String, who wrote the memo • **to**: String, who the memo is for	• **Memo( )** : default constructor • **Memo(name, body, from, to)**: overloaded constructor; call super(name, body) • **toString**(): override of abstract method
TimedMemo extends Memo	• **today**: String	• **TimedMemo**(name, body, from, to): overloaded constructor. Use java.time.LocalDate.now( ).toString() to get the current date. • **toString**(): override of Memo::toString()
PoliteTimedMemo extends TimedMemo	• **DEFAULT_GREETING**: "Dear" • **DEFAULT_CLOSING**: "Yours truly,"	**PoliteTimedMemo**(name, body, from, to): constructor **toString**(): override of TimedMemo::toString()

Console I/O Examples

Console I/O	Method and some description
Main Menu	main() method calls getMenuChoice(mainMenu)
1. Create a new Note	

2. Display existing Note(s) 3. Quit	
Enter your choice:	
1	*User wants to create a Note; show submenu*
Note Creation	main() method calls getMenuChoice(createMenu)
1. Create a Memo	
2. Create a Timed Memo	
3. Create a Polite Memo	
4. Return to previous menu	Continue asking until the user chooses #4
Enter your choice:	
1	*User wants to create a Memo*
Enter Memo name:	Use nextLine( ) for all entries
Intro	
Enter Memo body:	
Hi, I'm Alice	
Enter who this is from:	
Alice	
Enter who this is to:	
Bob	
From: Alice To: Bob Name: Intro Body: Hi, I'm Alice Note# 1 ***** Powered by Make-a-Note *****	This message printed using Memo's toString() method, which in turn uses Note's toString(); the Note# is 1, because this is the first note created; the footer comes from Note's FOOTER
Note Creation	main() method calls getMenuChoice() in a loop
1. Create a Memo	
2. Create a Timed Memo	
3. Create a Polite Memo	
4. Return to previous menu	
Enter your choice:	
Note Creation	
4	*User wants to return to main menu*
Main Menu	main() method calls getMenuChoice(mainMenu)
1. Create a new Note	
2. Display existing Note(s)	
3. Quit	
Enter your choice:	
2	*User wants to display Notes; show submenu*
Display Options	main() method calls getMenuChoice(createMenu)
1. Display all Notes	
2. Display Note by Number	
3. Display Notes by Name	
4. Return to previous menu	Continue asking until the user chooses #4
Enter your choice:	
1	*User wants to display all Notes*
From: Alice To: Bob Name: Intro Body: Hi, I'm Alice Note# 1 ***** Powered by Make-a-Note *****	This is the only note so far.
For choices 2 and 3, prompt for the information needed for the search. If no matching notes are found, print "None found". Keep prompting until the user chooses #4. Redisplay the main menu; keep prompting there until the user chooses Quit, #3.	

Example of a Timed Memo:

Date: 2020-10-05 From: Charlie To: Donna Name: Meeting Body: Meeting today – don't forget! Note#: 2 ***** Powered by Make-a-Note *****	This message printed using TimedMemo's toString(), which in turn uses Note's toString( ); Note# is 2, because this is the second Note created.
Example of a Polite Memo:	
Date: 2020-10-05 Name: Birthday Dear Fiona: Happy Birthday! Yours truly, Erica Note#: 3 ***** Powered by Make-a-Note *****	This message printed using PoliteTimedMemo's toString() method using a FormattedDate and greeting; Note# is 3, because this is the third Note created.