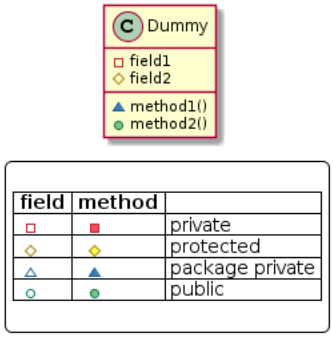
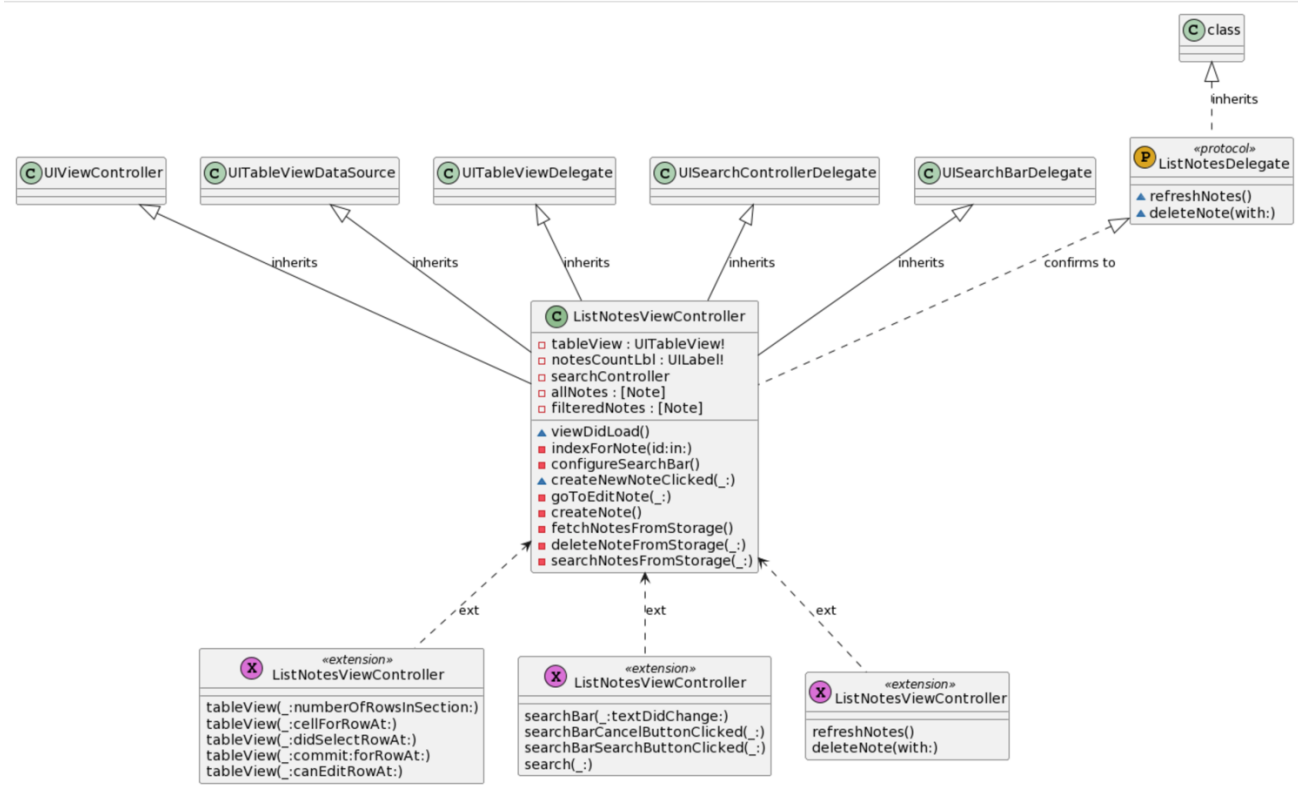
CIS641
Fall 2022

CLASS DIAGRAMS



Legend



Class Name – ListNotesViewController

Controller class responsible to update the Notes list view. Holds list of Notes in the allNotes and filteredNotes arrays. Implements delegator pattern to implement methods to delete and refresh notes whenever

goToEditNote() –

- **Precondition** - Note selected in list
- **Postcondtion -** User is navigated to edit note view

createNote() –

- **Precondition** – Create button tapped
- **Postcondtion –** User is navigated to edit note view with a blank note

fetchNotesFromStorage() –

- **Precondition** – App is launched and lands on notes list screen
- **Postcondition –** Notes are fetched from the local database and displayed

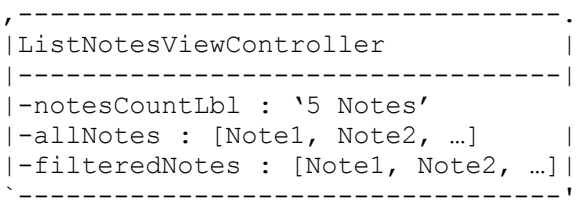
deleteNoteFromStorage() –

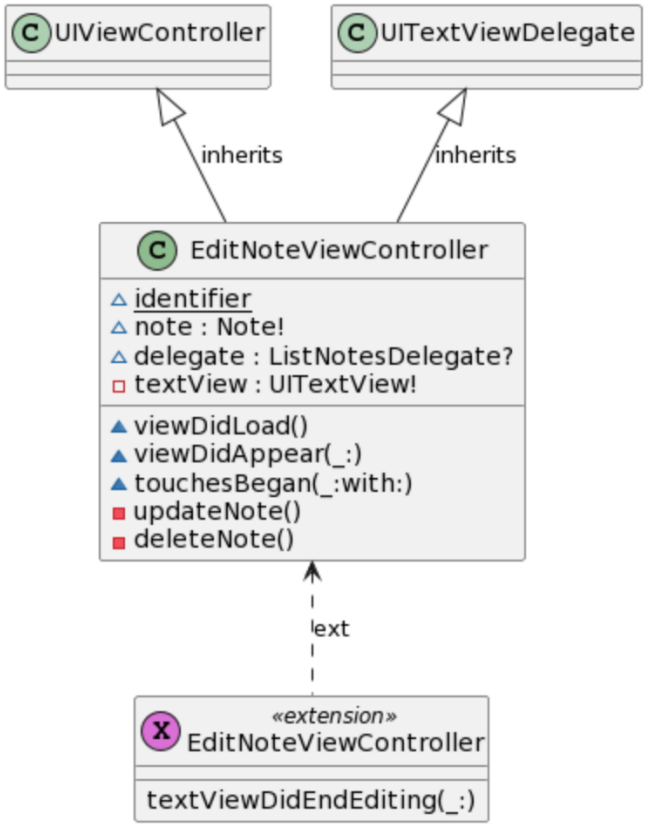
- **Precondition** – User selects delete note
- **Postcondition –** Note gets deleted from the local database and the list is refreshed

searchNotesFromStorage() –

- **Precondition** – User searches for a note using the search bar
- **Postcondition –** Local database is querie and the results are displayed in the notes list

Object Diagram





Class Name – EditNoteViewController

Controller class responsible to update the Edit note view. Holds the note being edited in note property.

updateNote() –

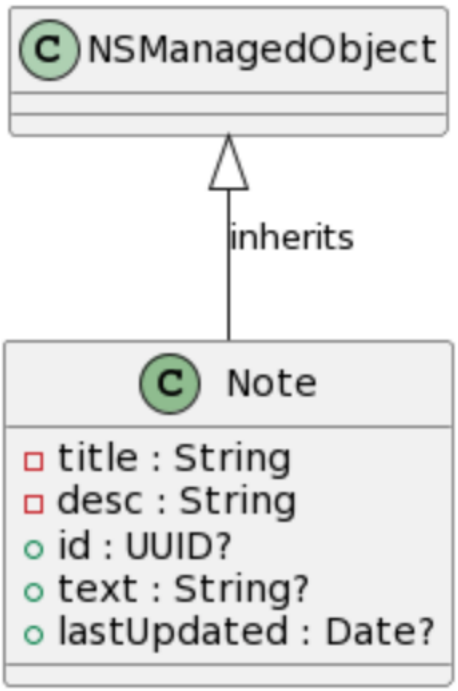
- **Precondition** – User edits text in the text view.
- **Postcondtion –** Note is saved to the database.

deleteNote() –

- **Precondition** – User deletes all text in note.
- **Postcondtion –** Note is deleted from the database.

Object Diagram

```
,-----------------------------------------------------------.
|EditNoteViewController                                     |
|-----------------------------------------------------------|
|~note : {title: 'Grocery list', desc:'potatoes, eggs'}     |
`-----------------------------------------------------------'
```

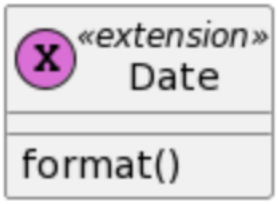


Class Name – Note

Model class responsible to hold note data. title and desc are computed properties generated at runtime by using the text property. title is the first line in text and desc is the second line onwards.

Object Diagram

```
,------------------------
|Note                    |
|------------------------|
|-title :"Grocery List"  |
|-desc :"Potatoes\n Eggs"|
|+id : "12"              |
|+text : String?         |
|+lastUpdated :"10:47 PM"|
`------------------------
```



Class Name – Date

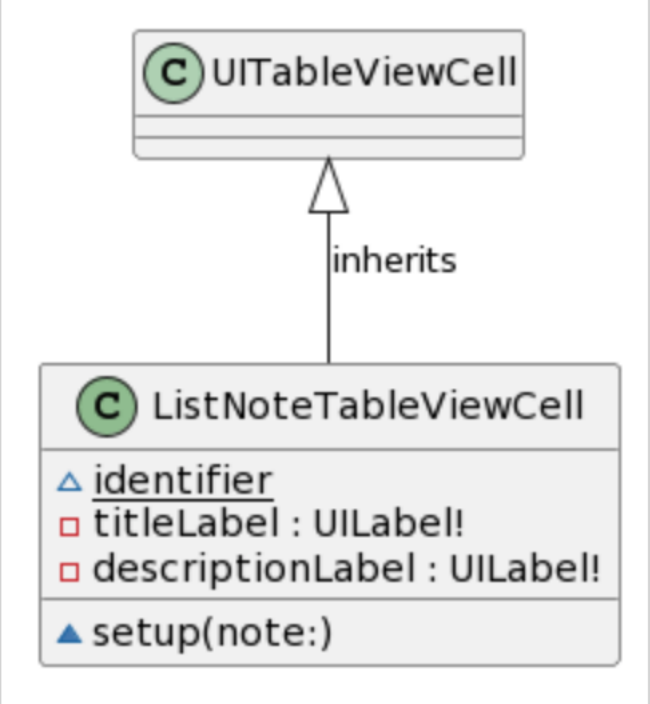
Extension to the Swift Date class. Sets format to “h:mm a” or ‘dd/MM/yy’ based on the date lying on today’s date or not.

```
func format() -> String {
        let formatter = DateFormatter()
        if Calendar.current.isDateInToday(self) {
            formatter.dateFormat = "h:mm a"

        } else {
            formatter.dateFormat = "dd/MM/yy"
        }
        return formatter.string(from: self)
    }
```

updateNote() –

- **Precondition** – format is called on the date object holding the note’s last updated time.
- **Postcondtion –** The date is formatted accordingly based on the condition.



Class name – ListNoteTableViewCell

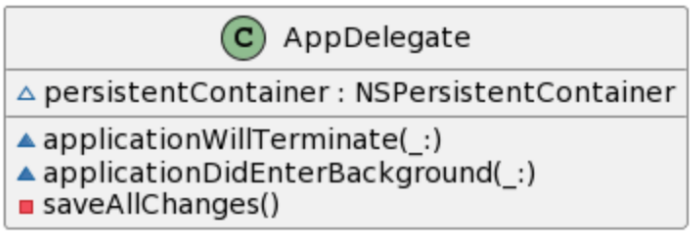
View class for representing the items in the list. Has private view properties which are labels – titleLabel and descriptionLabel.

setup() –

- **Precondition** – view is put into a table view and a note object is passed to the function
- **Postcondtion –** The labels are set to the title and desc properties of the note object passed

Object Diagram

```
,---------------------------.
|ListNoteTableViewCell      |
|---------------------------|
|-titleLabel : "Grocery List"|
|-descriptionLabel : "Eggs". |
`---------------------------'
```



Class name – AppDelegate

This class is implemented to save the changes to the local database. The functions here are called when there is a change in the application lifecycle i.e. user closes or minimizes the app. The objective here is to make sure the local database is committed with the changes that were made during the editing of the note.

applicationWillTerminate() –

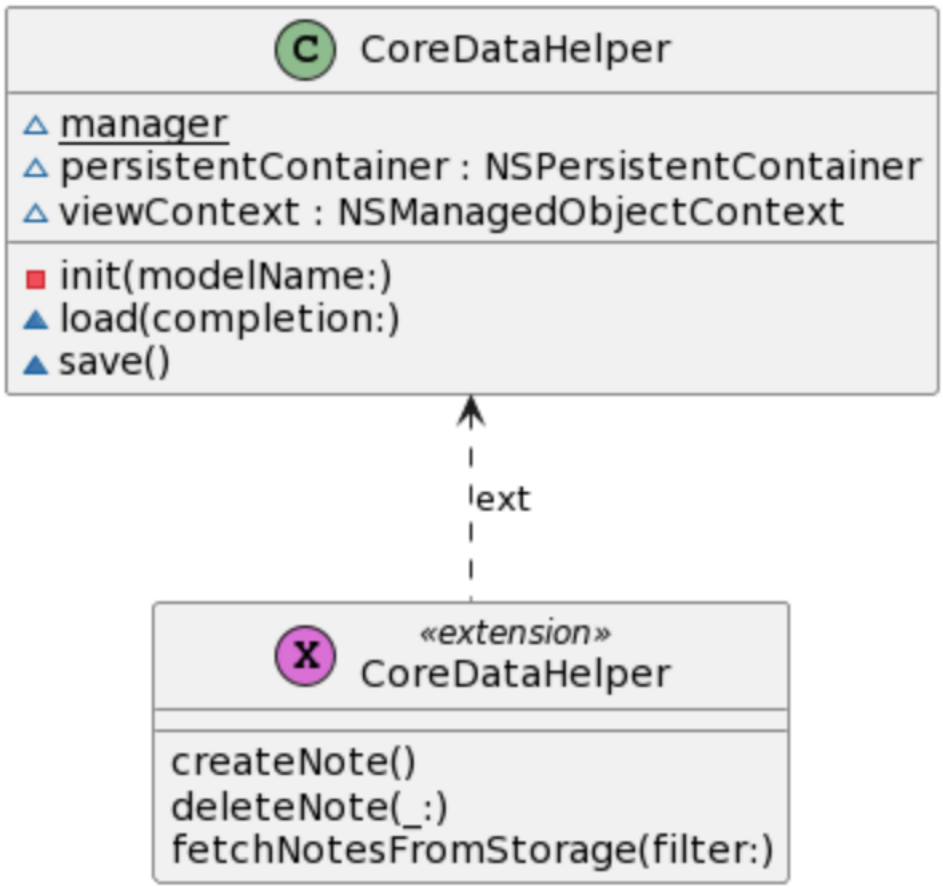
- **Precondition** – User kills application
- **Postcondtion –** function is called, and the code is executed

applicationDidEnterBackground() –

- **Precondition** – User minimizes the app
- **Postcondtion –** function is called, and the code is executed

saveAllChanges() –

- **Precondition** – Application is terminated or minimized to background
- **Postcondtion –** Local database is updated with the changes.


CoreDataHelper
manager
persistentContainer : NSPersistentContainer
viewContext : NSManagedObjectContext
init(modelName:)
load(completion:)
save()
ext
«extension»
CoreDataHelper
createNote()
deleteNote(_:)
fetchNotesFromStorage(filter:)