Bringing It Together - Mav’s Ice Cream Emporium (Sprint 2)

Due Tuesday, October 11 at 8 a.m.

CSE 1325 - Fall 2022 - Homework #6 / Sprint 2 - Rev 1 - 1

Assignment Background

Mav’s Ice Cream Emporium (MICE) is a fledgling start up in the dairy treat market. They are seeking bright young programmers to build a custom solution for defining new confections, tracking customers and the treats they buy, ensuring timely delivery of a quality product, and otherwise taking care of business. Your job is to win this project (with associated profit and glory) from Mav’s Ice Cream Emporium by producing a proposal package over 6 Sprints, including a prototype that wows and other creative and persuasive artifacts that prove you know your stuff.

This is Sprint 2 of 6, in which we add a simple Graphical User Interface (GUI) to our work from Sprint 1!

The Scrum Spreadsheet

Copy your P05 directory completely to P06, including last week's Scrum spreadsheet.

The Product Backlog has changed! Features CC and CS have moved to a later Sprint, as the estimate for the work required for the first two features GUI and IGUI were low.

As always, update your Product Backlog tab to reflect your plan and your actual work for Sprint 2. Then, on the Sprint 2 Backlog tab, plan the tasks you'll need to implement both features (the hints below should help) and update your status as you go.

Be sure to add, commit, and push the updated spreadsheet at cse1325/P06/Scrum_MICE.xlsx before the end of the sprint!

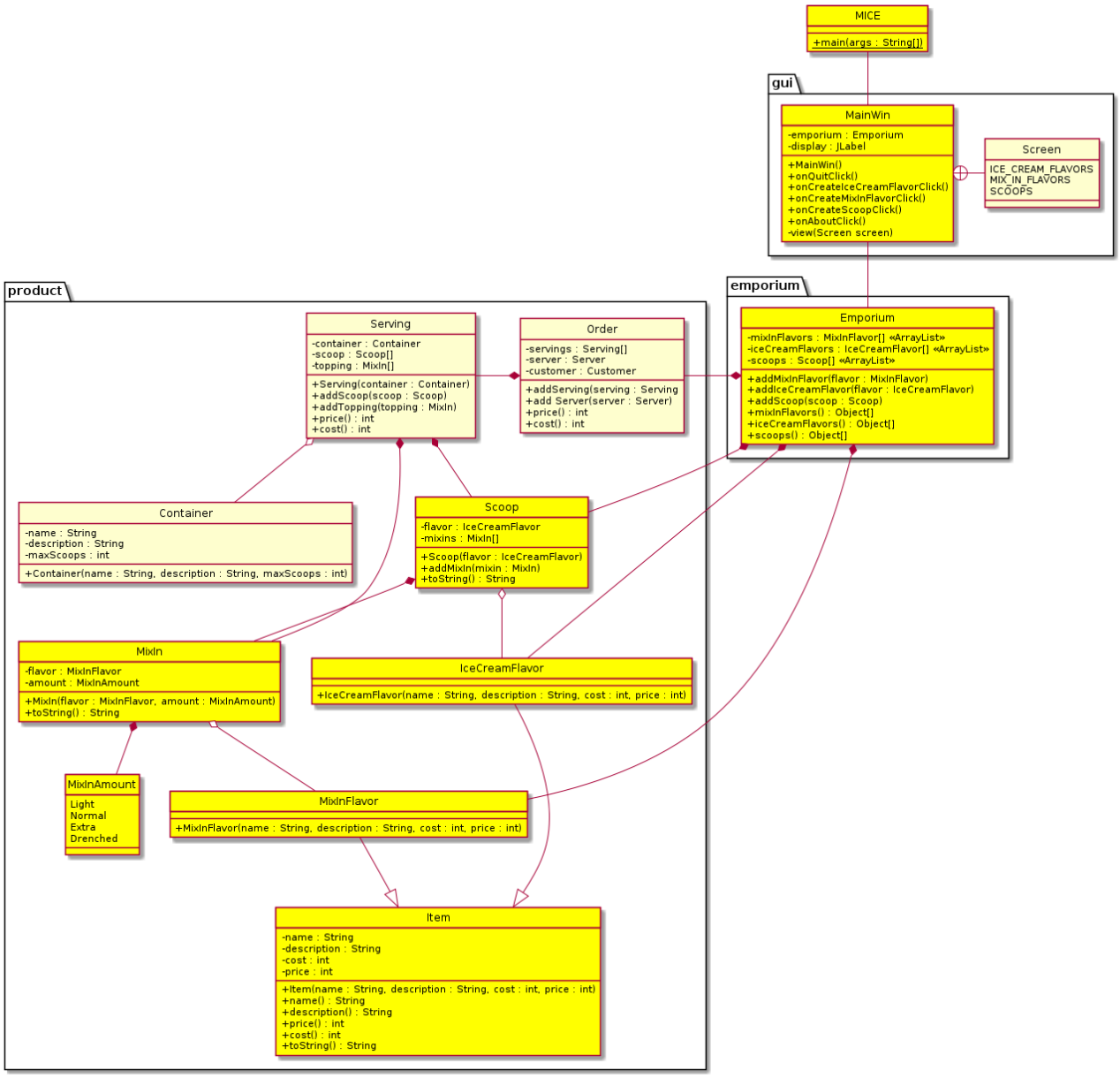
Class Diagrams

The diagram on the next page has been simplified by removing the person package (with Customer, Server, and so on) and by removing members of Emporium and MainWin compared to the final system that are not yet needed.

As with Sprint 1 you will only be implementing the dark yellow classes this Sprint.

As before, updates to fix bugs and design issues are all but inevitable.

(By the way, the ■ symbol on the relationship between Screen and MainWin indicates that enum Screen may be nested inside MainWin. But feel free to make Screen a stand-alone package-private or even public enum if you prefer. This symbol will NOT be on the exam!)


MICE
+main(args : String[])
gui
MainWin
-emporium : Emporium
-display : JLabel
+MainWin()
+onQuitClick()
+onCreateIceCreamFlavorClick()
+onCreateMixInFlavorClick()
+onCreateScoopClick()
+onAboutClick()
-view(Screen screen)
Screen
ICE_CREAM_FLAVORS
MIX_IN_FLAVORS
SCOOPS
product
emporium
Emporium
-mixInFlavors : MixInFlavor[] «ArrayList»
-iceCreamFlavors : IceCreamFlavor[] «ArrayList»
-scoops : Scoop[] «ArrayList»
+addMixInFlavor(flavor : MixInFlavor)
+addIceCreamFlavor(flavor : IceCreamFlavor)
+addScoop(scoop : Scoop)
+mixInFlavors() : Object[]
+iceCreamFlavors() : Object[]
+scoops() : Object[]
Serving
-container : Container
-scoop : Scoop[]
-topping : MixIn[]
+Serving(container : Container)
+addScoop(scoop : Scoop)
+addTopping(topping : MixIn)
+price() : int
+cost() : int
Order
-servings : Serving[]
-server : Server
-customer : Customer
+addServing(serving : Serving
+add Server(server : Server)
+price() : int
+cost() : int
Container
-name : String
-description : String
-maxScoops : int
+Container(name : String, description : String, maxScoops : int)
Scoop
-flavor : IceCreamFlavor
-mixins : MixIn[]
+Scoop(flavor : IceCreamFlavor)
+addMixIn(mixin : MixIn)
+toString() : String
MixIn
-flavor : MixInFlavor
-amount : MixInAmount
+MixIn(flavor : MixInFlavor, amount : MixInAmount)
+toString() : String
IceCreamFlavor
+IceCreamFlavor(name : String, description : String, cost : int, price : int)
MixInAmount
Light
Normal
Extra
Drenched
MixInFlavor
+MixInFlavor(name : String, description : String, cost : int, price : int)
Item
-name : String
-description : String
-cost : int
-price : int
+Item(name : String, description : String, cost : int, price : int)
+name() : String
+description() : String
+price() : int
+cost() : int
+toString() : String

Main Package

Class MICE

Class MICE contains the main method, which simply instances gui.MainWin. Thus your program will be run using the command `java MICE`.

Emporium Package

Class Emporium

Class Controller has been renamed Emporium to better fit the customer's view of the system. This is effectively the database, where our persistent data will be stored.

This sprint, Emporium contains 3 ArrayLists: mixInFlavors, iceCreamFlavors, and scoops. Objects are added to these via setter-like methods addMixInFlavor, addIceCreamFlavor, and addScoop.

An array Objects[] may be obtained from class Emporium for each of the 3 ArrayLists. This may look odd, but you'll discover that Swing can work *wonders* with an array of Objects. So it's no coincidence that, given `ArrayList al`, you can obtain an array of its elements as an Object array with a simple `al.toArray()`.

Trust me. It will work out wonderfully!

Product Package

Move classes `Item, MixInFlavor, MixInAmount, MixIn, IceCreamFlavor`, and `Scoop` into package `product`. This requires 2 changes:

- Move the .java files into a new subdirectory named `product`.
- Add `package product;` to the top of each of these .java files.

No other changes are required for our classes from last week.

Test Package

While not shown, you *may* move your TestScoop.java into package test if you like.

Put the file into a new subdirectory named test, then add this to the top of your TestScoop.java file to tell the compiler this directory is package test and to import the classes from the product package:

package test;

import product.IceCreamFlavor; import product.MixInFlavor; import product.MixInAmount; import product.MixIn; import product.Scoop;

It should then build properly from the P06 directory with the `ant` command, since `ant` by default compiles all .java file recursively.

To run your test program, from the P06 directory, type: `java test.TestScoop`

GUI Package

Most of your effort this sprint will be focused on class MainWin in package gui.

- Create a gui directory.
- I recommend that you *baseline* (copy) file cse1325-prof/12/code_from_slides/nim/MainWin.java and modify it to serve as your user interface for this sprint.

Class MainWin

Constructor

As with Nim, the constructor creates your main window and registers your Observers (ActionListeners). Your menu will look like this:

```
File ----- Quit
View ----- Ice Cream Flavor
        +- Mix In Flavor
        +- Scoop
Create --- Ice Cream Flavor
        +- Mix In Flavor
        +- Scoop
Help ----- About
```

- Quit should invoke `onQuitClick()` when selected, as in Nim.
- The 3 MenuItems under View should call the `view()`, indicating whether to display ice cream flavors, mix in flavors, or scoops in the main window display. I suggest an enum, with an enumerated value passed as a parameter to `view` from the lambda to indicate which to display, but you are free to select a different approach.
- The 3 MenuItems under Create should invoke `onCreateIceCreamFlavorClick`, `onCreateMixInFlavorClick`, and `onCreateScoopClick`, respectively.
- About should invoke `onAboutClick()`.

You are *not* required to add a toolbar this sprint, although since you have the code *right there* you could go ahead and get it over with. Your call.

Add JLabel `display` as the main data area of your main window. This is a field so that it can be updated with new text once the constructor exits.

Set the main window visible, as in Nim, and you're there. (Be sure you've instanced an Emporium either inline or somewhere in your constructor!)

OnQuitClick

A simple `System.exit(0)` is fine here.

onCreateIceCreamFlavorClick

You may use a sequence of 4 InputDialogs (from JOptionPane) OR a single input dialog to gather the data needed for the new flavor of ice cream. You shouldn't be surprised that in the final prototype, we'll be

expecting single dialogs, so that's the best option - but you won't lose points for a sequence of dialogs this sprint.

Once you have them, instance IceCreamFlavor and pass the new object to Emporium's addIceCreamFlavor method.

Set the view to a list of IceCreamFlavors.

onCreateMixInFlavorClick

This method is the mirror image of onCreateIceCreamFlavorClick, so the same hints apply.

Set the view to a list of MixInFlavors.

onCreateScoopClick

This is where it gets more interesting. You will want the user to specify an IceCreamFlavor instance so that you can instance a Scoop. How to do this?

JOptionPane.showInputDialog's 7-parameter overload is a quick and easy static method. If the 6th parameter is an Object[] it will render the toString of each Object into a combo box and then return the Object that was selected by the user. See why Emporium returns those Object arrays instead of the actual types?

- If the user clicked Cancel, it will return null. Be sure to handle this well.
- If the user clicked OK, it will return the Object selected. You must cast this back into an IceCreamFlavor object to use with the Scoop constructor.

You will also need to select MixInFlavors, which works very much the same way.

But how to select a MixInAmount? Happily, MixInAmount.values() can be used in the same way as MixInFlavors.toArray() with JOptionPane.showInputDialog's 7-parameter overload.

Once you have a MixInFlavor and MixInAmount, instance your MixIn and call your scoop's `addMixIn`.

Repeat until you get a null for MixInFlavor, at which point add the scoop to Emporium with its `addScoop` method.

Finally, set the view to a list of Scoops.

view

You need a way for the user to control what data is shown in the `display` JLabel. You may implement this as you like.

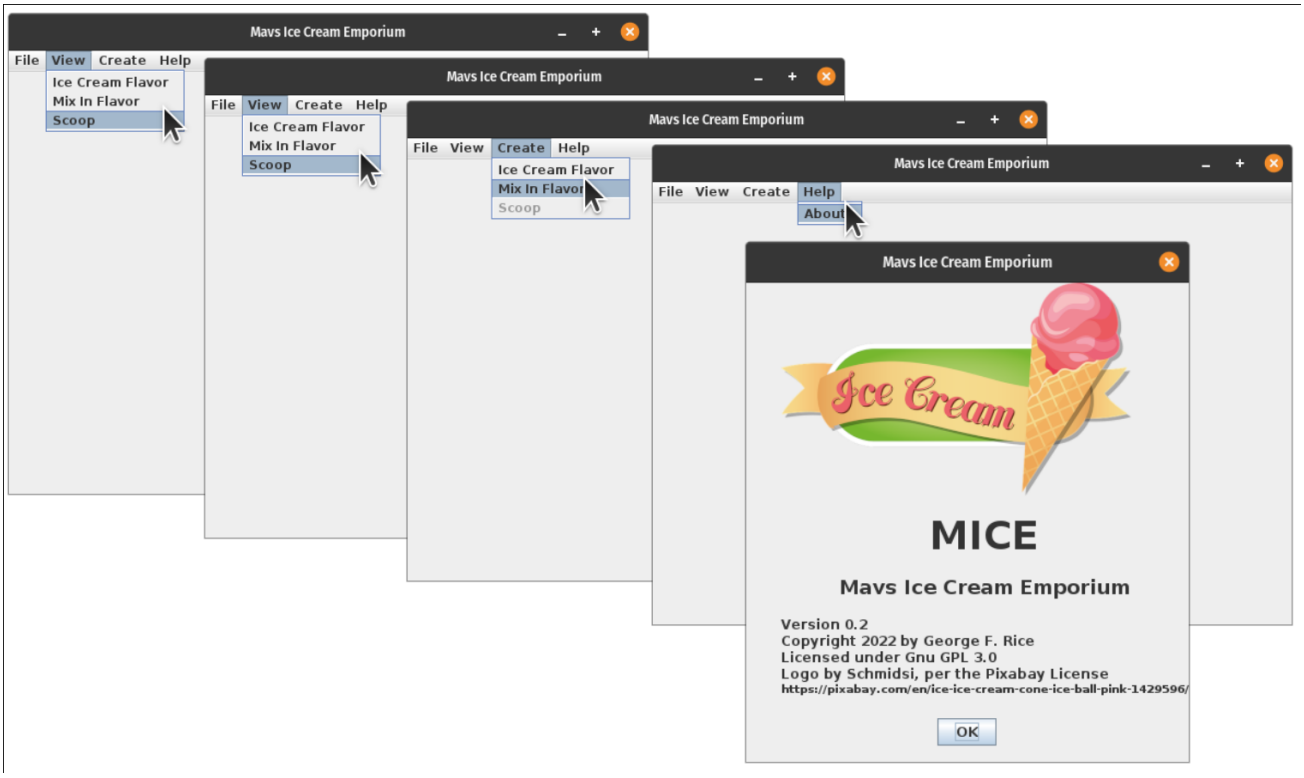
The suggested solution nests a private enum inside MainWin called Screen that enumerates the 3 displays currently available. The private view method accepts a Screen as a parameter, and uses `display.setText` to update the display with the requested data.

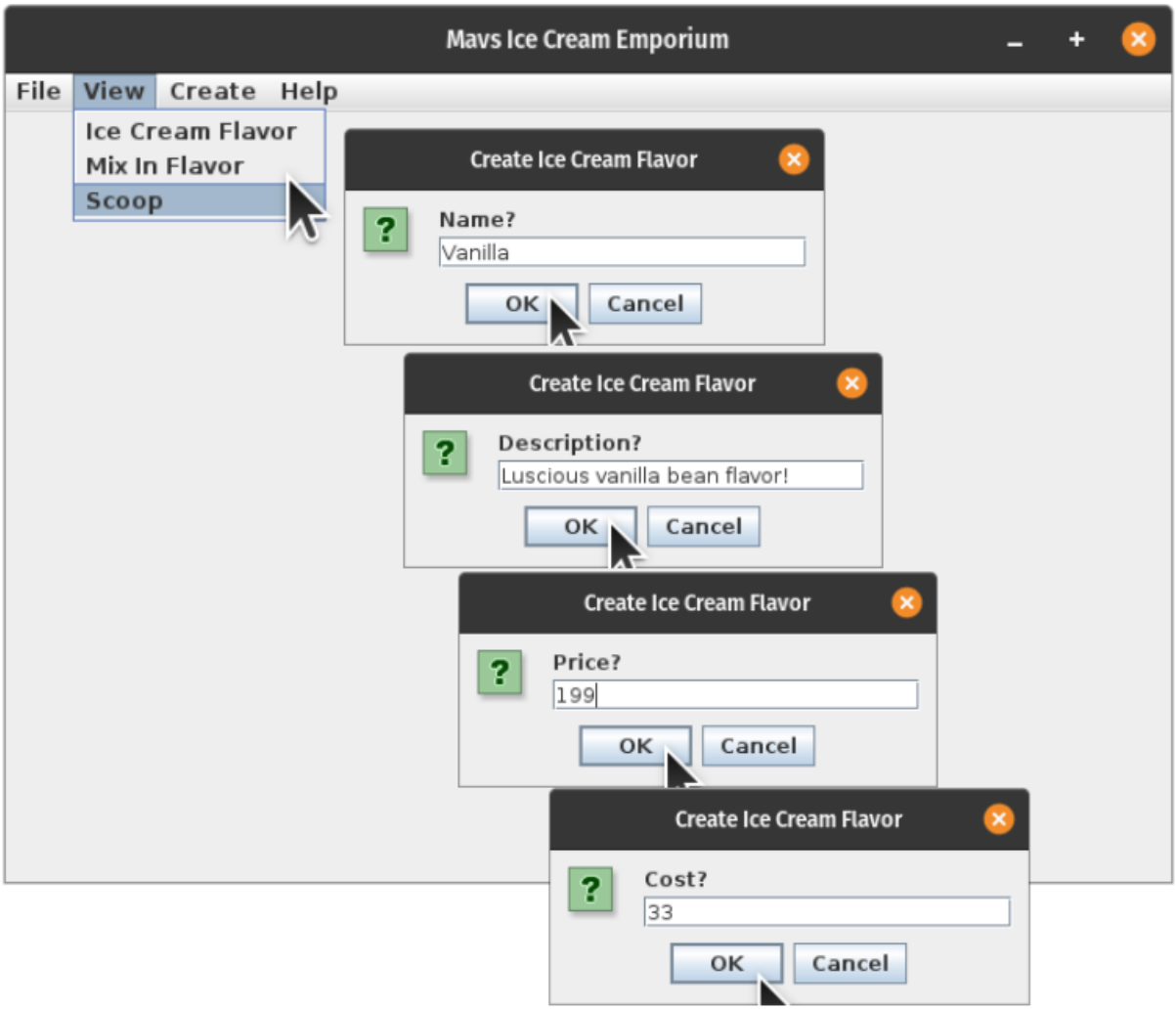
onAboutClick

You should be able to lightly adapt Nim's About dialog for your own use. Yes, this is required!

Screenshots

Your application is NOT required to look like this! You have significant freedom now to make this project your own, subject only to meeting the stated requirements. If you have questions as to whether your plans conform to the requirements, I'm happy to discuss with you via email or in my office.


Mavs Ice Cream Emporium
File View Create Help
Ice Cream Flavor
Mix In Flavor
Scoop
Mavs Ice Cream Emporium
File View Create Help
Ice Cream Flavor
Mix In Flavor
Scoop
Mavs Ice Cream Emporium
File View Create Help
Ice Cream Flavor
Mix In Flavor
Scoop
Mavs Ice Cream Emporium
File View Create Help
About
Mavs Ice Cream Emporium
Ice Cream
MICE
Mavs Ice Cream Emporium
Version 0.2
Copyright 2022 by George F. Rice
Licensed under Gnu GPL 3.0
Logo by Schmidsi, per the Pixabay License
https://pixabay.com/en/ice-ice-cream-cone-ice-ball-pink-1429596/
OK


Mavs Ice Cream Emporium
File
View
Create
Help
Ice Cream Flavor
Mix In Flavor
Scoop
Create Ice Cream Flavor
Name?
Vanilla
OK
Cancel
Create Ice Cream Flavor
Description?
Luscious vanilla bean flavor!
OK
Cancel
Create Ice Cream Flavor
Price?
199
OK
Cancel
Create Ice Cream Flavor
Cost?
33
OK
Cancel