Bringing It Together - Mav's Ice Cream Emporium

Due Each Tuesday at 8 a.m.

CSE 1325 - Fall 2022 - Semester Project Overview - 1

See Requirements (in separate documents) for P05-P07, P10-P11, and P13

Assignment Background

One-week projects are fairly rare in the "real world", though not extinct. "Short" projects take many weeks, while longer projects can run into months or years. We can't squeeze a multi-year project into our class, but multi-week – that we can do! In this homework, you'll apply the simplified Scrum process you practiced with our Library Management System project to manage a 6-week development of a simple management system for an ice cream emporium.

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. For reasons not stated, they really want a mouse-driven application.

Your job is to win this project (with associated profit and glory) from Mav's Ice Cream Emporium by producing a proposal package, including a prototype that wows and other creative and persuasive artifacts that prove you know your stuff. You have 6 sprints over the next 9 weeks (with a few 1-week intervening assignments) to deliver your emporium manager prototype v1.0 with your proposal package to the Owner, at first featuring a glorious text menu or command line interface (CLI) but later a thoroughly compelling Graphical User Interface (GUI).

> **IMPORTANT:** An earlier C++ semester project was also named Mav's Ice Cream Emporium, but the requirements this semester are somewhat different. Nothing screams lack of integrity and a strong desire to repeat this class louder than submitting a poorly implemented Java tranlation of the suggested solution from a similar C++ project that meets a previous semester's requirements instead of the new requirements. I wish I could say no student has ever tried, but I won't lie. Don't be That Student.

The Owner's Monologue

Here's what the Owner of Mav's Ice Cream Emporium (MICE) says they need.

MICE offers a revolving selection of ice cream flavors (sold by the Scoop), ice cream Containers (each of which can hold a maximum number of Scoops), and Toppings (which can be added on top of the ice cream in several quantities).

Each of these Items has a name, description, wholesale cost, retail price, stock remaining, and picture with which to entice the customer.

- An ice cream Scoop includes no additional information. Examples include vanilla, chocolate, cookies and cream, and raspberry swirl.
- An ice cream Container also includes the number of scoops of ice cream it can hold. Examples include bowls, cups, and sugar and waffle cones.
- An ice cream Topping can be requested as light, normal, extra, and drenched.

The emporium employs Servers, who have a name, uniquely assigned employee number, number of orders filled during their career, and their hourly salary.

The emporium also keeps track of Customers by name, phone number, and uniquely assigned customer number. Any Server can create a new Customer.

A Server or a Customer may select a Container, one or more scoops of ice cream Flavors (not exceeding the maximum for that container), and zero or more Toppings each from light to drenched to create a new Serving. The retail price of the Serving is the sum of the retail prices for the Container, scoops of ice cream Flavors, and Toppings.

The Server or Customer further assembles one or more Servings into an Order. The Order associates the Server (eventually – a Customer-created Order doesn't have one until it's filled!), the Customer, and one or more Servings, along with a unique order number (for this store), the total price (which is the sum of the price of each Serving contained therein, ignoring sales tax), and the state of the order: Unfilled, Filled, Paid, or Canceled.

- Once created, an order is initially Unfilled.
- The Customer for that order may transition an order from Unfilled to Canceled, which is a final state. Nothing else changes for a canceled order.
- Any Server may change the state from Unfilled to Filled, at which time the stock remaining of each Item in each Serving in the Order is reduced by the amount consumed. In addition, for every tenth order filled, the Server's hourly salary is deducted from the store's Cash Register.
- Any Server may change the state from Filled to Paid, which is a final state. When an Order is paid, the retail price of the Order is added to the store's Cash Register.

Any Server may replace items in the store – Scoops, Containers, and Toppings. The wholesale price of the items replaced is deducted from the Cash Register. Restocking is equivalent to filling 2 orders, regardless of the number of Items restocked, for purposes of paying the Server their hourly wage.

Manager may add new Items to the store, hire new Servers, and view reports on the store's operation.

- A Server Report, showing all Server data including how much they have earned total.
- A Customer Report, showing all Customer data.
- An Inventory Report, showing every Item and how many of each are currently available.
- An Order Report, showing every active Order, its current state, its wholesale cost and retail price, and the profit for that Order. An option also includes Filled and Canceled orders.
- A Profit and Loss Statement, showing income from all Orders, the cost of Server salaries and of Items added to stock, and calculating the difference as net profit.

To support future franchise operations, all of the above should be able to be managed independently for more than one Emporium. That is, a running instance of the program should be able to load the data for an Emporium, modify and save it, then load the data for another Emporium.

These features have associated priorities, but the Owner hasn't decided on them yet. The prototype need not implement all or even most of them, of course. Lots of additional features are also needed, which the Owner will inevitably think of as we go along. Welcome to reality.

Software Development Process

Scrum

Each Sprint concludes at (you guessed it) 8 a.m. each Tuesday with delivery to GitHub of your updated Scrum Spreadsheet, code, and other artifacts implementing that Sprint's features.

The Scrum spreadsheet lists each Feature that will be graded in each Sprint in the Required column of the Product Backlog tab. It also lists a selection of Bonus features along with their value in grade points in the Bonus Points column. You may implement features allocated to future Sprints early, but they will only be graded at the end of the Sprint to which they are assigned. Bonus features may be implemented at any time, and will be graded at the conclusion of the project.

The first week will focus on implementing a basic Model, along with either a simple command line interface for interactive testing OR a set of regression tests (or both for bonus points). We will begin implementing a Graphical User Interface in Sprint 2.

Every class documented in the standard Java 17 Class Library is available for your use on this project. You may also consider third party libraries that have been approved *in writing* by the Product Owner (that's the Professor, not the TAs!), as long as you comply with all license agreements. You will also need to obtain your own graphics and other needed resources, again in compliance with all applicable license agreements as documented in your About dialog and README.md file.

You must also of course use git with GitHub to version control and deliver your work. We'll stick with the P05 directory structure, although **the full_credit / bonus / extreme_bonus subdirectories are no longer required**. Instead, you'll (eventually) have your Java *packages* as subdirectories.

Do NOT write all of your code for the entire project and then ask us for help getting it to compile. That's not how we teach software development, nor is it how professional developers operate in the real world.

Baby steps!

Code a little, test a lot, backup frequently, version control always. See Lectures 00 and 09.

Changes to the Product Backlog

The list of Features, their priority, and (for Bonus Features) their bonus point value may change at the *start* of any Sprint. This is a standard feature of Scrum.

If you have already implemented Features for a Sprint early and will be impacted by a change, please contact the Professor. I'll mitigate any impacts to your plan to the extent possible.

Tempo

At the Start of each Sprint

- **Select Features:** Select the Features you'll implement for the next Sprint in the Planned column of the Product Backlog tab of the Scrum Spreadsheet. This should include *at a minimum* those Features that will be graded next Tuesday that you haven't yet implemented (shown in the Required column), along with any future or Bonus features you believe you can fit in. No worries - no points are deducted if you don't complete future or Bonus features you had hoped to complete. We only grade the Features listed in the Required column, *not* the Planned column, and we grade the Scrum Spreadsheet on its accurate and complete reflection of *reality* rather than the imaginary "perfect project".
- **Plan Tasks:** Plan the Tasks to implement each Feature on the Sprint xx Backlog tab (where xx is the current Sprint). We expect at least 3 to 5 Tasks per Feature, although more is perfectly fine. Be sure to link each Task to the Feature it supports via the Feature ID column (we call this "traceability", that is: *Why* are you planning this task?).

During each Sprint

- **Fix Bugs:** Bugs may be tracked on GitHub's Issues tab (recommended - this is how professionals track bugs), but bugs to be fixed on in future Sprint xx should *certainly* be listed as a Task on the associated Sprint xx Backlog tab in the Scrum spreadsheet. This ensures we don't overlook a bug fix.
- **Perform Tasks:** When you complete a Task, update its Status on the Sprint xx Backlog tab in the Scrum spreadsheet to the day on which it was completed. For example, if you complete a Task on Thursday (day 3), set its Status to Completed Day 3. The Sprint Burn Chart will update automatically.
- **Complete Features:** When you complete a Feature, update its Status on the Product Backlog tab in the Scrum spreadsheet to the current Sprint. For example, in Sprint 1, set the Status for each Feature you *complete* to "Finished in Sprint 1". The Product Backlog Burn Chart will update automatically. (If you fail to complete a Feature during the Sprint in which it is graded, we *expect* your spreadsheet to accurately reflect this! An *inaccurate* spreadsheet is wrong, not a truthful one.)
- **Push in Discrete Commits to GitHub:** Implement your plan by the following Tuesday at 8 a.m., **pushing each Feature to GitHub in a separate commit** (additional interim commits are *highly recommended*). Each commit message should indicate at a minimum single the Feature ID and Task ID it supports, and / or the GitHub Issue or Task ID (the Bug) it fixes. (Always use *separate* commits for each Bug fix and for each Feature implementation.)

At the End of each Sprint

- **Check Code and Resources:** Ensure all code has been committed to GitHub (at a minimum with a separate commit for each Feature), just as with previous assignments.
- **Push an Updated Scrum Spreadsheet:** Ensure that your *updated* Scrum spreadsheet is committed to GitHub in the P05 (or later) directory. Yes, this is graded, for completeness not the imaginary "perfect project". These are easy points - don't miss them!

Suggested Solutions

- **Review For Ideas:** I *will* post suggested solutions each week to give you insight into how your project should be proceeding (although your project may and *should* be uniquely yours and thus perhaps significantly different than mine). You may *reference* my code to enhance your own without attribution, just as novel authors will consult The Bard at times to improve their plots and inspire their own prose. Similarly, you may look at similar projects on GitHub or StackOverflow seeking techniques, idioms, and licensable resources to try out on your project.
- **(Optional) Adopt as a Baseline:** Copying an entire *method* or more from a suggested solution requires attribution in your About dialog and README.md file as well as licensing your own code under the Gnu General Public License 3.0 (under which my own code is released). You can find the attribution requirements at https://www.gnu.org/licenses/gpl-3.0.html#howto. Yes, we scan and verify attribution! But if you make a good faith effort to comply with licenses, you need not worry.

At the End of the Project

- **(Optional) Add Sales Material:** Additionally push to GitHub any sales material you believe will increase the probability of winning the contract (e.g., PowerPoint slides, PDF brochures, websites, YouTube or TicToc videos) to GitHub with the final Sprint. This will garner additional bonus credit at the Professor's sole discretion (but I'm easily impressed).
- **Present Your Project (If Selected):** The creators of the best projects (at the discretion of the professor and TAs) will be invited to present their projects to the class (3-5 minutes only). If time permits, volunteer presenters will also be accepted. Each presenter will receive *substantial* bonus points. Programmers, indeed *all* engineers, need to communicate effectively to succeed! Will there be snacks during the presentations? Do dogs bark?
- **Win the Lucrative Contract™:** In addition, after the presentations the class will vote by secret ballot on which developer should be awarded the lucrative Mav's Ice Cream Emporium (MICE) development contract, which includes even *more* substantial bonus points. (Those presentation materials should prove useful here.) "Go confront the problem. Fight! Win! And call me when you get back, darling. I enjoy our visits." -- Edna Mode

The Bonus Sprint

Unlike previous homework, your opportunity for bonus credit is primarily rooted in completing additional Features within the 6 Sprint project limit. Thus, it is to your advantage to complete Features ahead of the Sprint schedule to allow time for Bonus Features to be added by the end of the scheduled project.

Sprint 6 is the "Bonus Sprint", with no assigned Features, and concludes on the day of the last exam (and thus ends *after* the project presentations). It is specifically created to give you one last chance to enhance your grade for the semester. You are NOT required to submit anything for the last Sprint.

The value of each Bonus Feature listed on the Product Backlog of the Scrum Spreadsheet is listed in the Bonus Points column.

Teaming

Teaming is not permitted this semester. Better professors than I may give you opportunities to work in groups, but not I.