Robbie Robot Shop in Pictures!

CSE 1325 – Fall 2016 – Homework #6
Due Tuesday, November 22 at 8:00 am
with Sprint Reviews due 8:00 am on October 27 and November 3, 10, and 17

Command Line Interface (CLI) applications are still around and in use, especially by more technical folk, but most desktop applications today offer a Graphical User Interface (GUI) based either on the native hardware via frameworks such as FLTK or on HTML 5 standards. For our final homework project, we will continue our weekly sprints with an updated Feature Backlog that is focused on adding a native GUI for our existing features as well as any additional features we have time to implement.

Introduction

Having been selected in the first round of competition to continue to the second and final round, Robbie Robots is now asking that your shift your focus from the business logic (the Model) to the user interface (the Controller and View). Rather than a basic CLI interface, the prototype for phase 2 will be GUI-based.

You're job is to win this phase and theproject from RRS by producing an updated proposal package, including a prototype GUI front-end that wows and other creative and persuasive artifacts that prove you know your stuff. You have a little under 5 weeks to deliver RRS Manager (Prototype) v2.0 (“now with pictures!”) with your proposal package, featuring a slick new GUI front-end.

Background

(From Homework #5) Robbie Robot Shop assembles their robots from 5 different components – a torso, head, arm(s), locomotor, and battery(ies). A Product Manager (PM) defines new robot models from these components, assigns a product name and price, writes up a brief sales description, and approves the result. The robot then appears at their Point Of Sale (POS) systems so that their Sales Associates (SA) can arrange for their Beloved Customers (BC) to order them, as well as on their web store so that BCs can order them direct. The Pointy-haired Boss (PB) will need to track profit margins, sales, and such from the robot product lines to ensure the business is profitable.

Requirements

(From Homework #5) The Product Owner has identified the needs of each of the actors that will interact with the system, and has assembled these into a prioritized Product Backlog in the Scrum spreadsheet, which is included in this set of Requirements by reference.

Product Manager (PM)

(From Homework #5) The PM needs to be able to create new instances of robot components in the system. For each component, they need to specify a name, part number, type (torso, head, arm, locomotor, or battery), weight, cost, and a brief description. In addition, some types of components need additional data: each torso may have from 1 to 3 battery compartments, each locomotor should specify a maximum speed (in MPH) and power consumed when operating (in watts), each arm should specify power consumed when operating (in watts), and each battery should specify the energy it contains (in kilowatt hours).

(From Homework #5) The PM needs to be able to define new robot models by selecting one or more of each type of robot component. Only one component of each type may be added, except that up to 2 arms and as many batteries as will fit into the selected torso may be attached. For each robot model, the PM also needs to define a model name, model number, and price. They will want to know the total cost of all components when setting the price to ensure adequate profit for each model.

Beloved Customer (BC)

(From Homework #5) A BC needs to be able to browse the catalog of robots, with pictures (eventually – not expected on the command line version), prices, shipping costs, and description at the POS terminal and (eventualy) on the web store. Each customer will also need to view their orders and their outstanding bill.

Sales Associate (SA)

(From Homework #5) A Sales Associate (SA) needs to order robot models for Beloved Customers and present a bill of sale listing the order number, date of sale, customer name, robot(s) ordered, and price (subtotal, shipping, tax, and total). Each Sales Associate would like a sales report for the sales they personally completed with which to lobby for a raise.

Pointed-haired Boss (PB)

(From Homework #5) The Boss needs to know overall shop metrics, such as the profit margin on each robot and how many were sold, a complete list of orders during a specified period, and sales for each Sales Associate with which to approve or deny requests for raises.

Process

Each developer will continue to follow a simplified Scrum process as identical to Homework #5, with an expanded Product (or Feature) Backlog list that now includes explicit GUI requirements. **We will number the sprints for Homework #6 as 4, 5, 6, 7 and 8** to recognize that we are continuing the same Scrum development effort.

As before, the developer is free to negotiate changes to the Product Backlog features and priority with the Product Owner (the Professor or TA). An approved change order is still required, an email from the Product Owner clearly stating the the change in the Product Backlog is acceptable.

You must provide your own *updated* UML models and design your own menus – that's part of this assignment. **Update your UML design before you start coding** – that's just good OO! **Neither the professor nor TA will discuss your project with you *at all* unless you bring your own UML models.**

Every class documented in the standard C++ library remains available for your use on this project, however, **you must use FLTK 1.3.x to implement your GUI**. You may also consider third party libraries that have been approved in advance by the Product Owner in writing, as long as you **comply with all license agreements.**

Students are expected to remain on their team if they teamed or to continue to work individually based on their choice for Homework #5, and to use the same delivery method for their work products (GitHub or git archive file). **Any teaming or collaboration changes must be approved in advance by the Professor.**

Regardless of whether the project is individual or team-based, **the product baseline should be built frequently – at least once per day, or better yet after every patch set is added – and all of the automated tests run** to identify any breakage that needs to be added to the Sprint Backlog of tasks, prioritized, and addressed. Build as small a set of functionality as can be tested, commit it to git, compile it, and test it. Note any bugs found in the Sprint Backlog (task list) for prioritization and eventual resolution (though not necessarily before the end of that sprint). Then move on. **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. **Code a little, test a lot, backup frequently, version control always.**

In lieu of weekly sprint demos, students will submit “Evidence of Compliance” via Blackboard at the end of each sprint – each Thursday at 8 am. The product itself will not be graded, but rather in sprints 5 through 7 the evidence that the student or team has (1) worked the highest priority backlog features, (2) delivered a tested, working prototype suitable for delivery if the Product Owner decided it's needed in production, and (3) is making suitable progress toward completion of a prototype by the 5th sprint. (Since Exam #2 coincides with the end of the sprint 4, no progress due to studying is acceptable for that sprint only – *though you must still submit your weekly sprint review material*.)

The project ends with Thanksgiving holidays, and thus sprint 8 (the final sprint) is only 5 days long. **Final work products are due in Blackboard by TUESDAY, November 22 at 8 a.m.** This is the Tuesday just prior to Thanksgiving. No extensions will be given. Finish your project and then go enjoy the holidays.

Questions? Concerns? Puzzlement beyond all reason? Contact your Product Owner ASAP! You have only 5 remaining one-week sprints to complete your initial proposal package, so don't waste time waffling.

Deliverables

Each student or team is required to use the git version management system, and may optionally use GitHub. **An archive of the current, stable git repository with commit history or a GitHub link, or a link with access granted to the Product Owner, representing that sprint's deliverable-ready milestone must be delivered to Blackboard *every week*.** {10 points per sprint, 50 total}

Each student or team is required to maintain and work from the Product Backlog and (for each sprint) the Sprint Backlog, using the Scrum spreadsheet provided in Homework #5 with an updated Product Backlog. Each Sprint Backlog item (task) must be implemented by the single team member with whose initials it is associated. This will enable the graders to estimate the portion of work contributed by each team member. **An updated copy of the Scrum spreadsheet must be delivered to Blackboard *every week*.** {10 points per sprint, 50 total}

Each student or team is required to maintain and work from a UML design, updated as the project progresses. These will include *at a minimum* a class diagram and use case diagram. **Each student or team will deliver an updated set of UML diagrams representing progress toward the final architecture and design *every week*.** These may be in Umbrello format or as PNG, JPG, or GIF images. Use of a computer tool rather than photographs of hand-drawn is *strongly recommended*. Note that if you ask for help with your project, **the Professor or TA will LALWAYS start with your UML diagrams – and if you haven't any with you, that's your answer!** {20 points per sprint, 100 total}

The student or team will deliver a final proposal package to Blackboard that presents their capabilities in the best possible light by the homework due date. The proposal package will include:

1. A working prototype of the system in source code form;
2. The git source code repository with the complete commit history of the project;
3. Supporting UML diagrams including a use case diagram and a class diagram that accurately reflects the code, with additional consideration given for other useful supporting diagrams);
4. A brief user manual (a few paragraphs); and
5. Any sales material deemed to increase the probability of winning the contract (e.g., PowerPoint slides, PDF brochures, YouTube videos).

The average individual student is expected to complete the top **12** Product Backlog items[1] by the end of the 8th sprint, including a working GUI interface and reasonably complete automated (non-interactive) test code. The Product Backlog is color coded by team size – grey for work that should have been completed in Sprints 1 – 3, black for individual expectation, green for 2-person team expectation, blue for 3-person team expectation, and red for 4-person team expectation.

Partial work may be submitted *for Sprint 8 only* for grading consideration. Completing additional Features from the backlog will be treated as extra credit work similar to Bonus levels. Persistence is worth up to 30 bonus points, and each additional Feature is worth up to 15 bonus points. {180 points plus bonuses}

Each student or team should be prepared to provide a final Product Demo to the class, if selected and scheduled by the Professor after completion of the 3th and final sprint. The number of Product Demos, if any, is dependent on availability of class time. More info to follow. {Up to 50 points bonus}

Grading

Homework #6 is graded on a 500 point scale (since this is a 5 week project), with opportunity for substantial additional credit.

Sprint #	4	5	6	7	8	Total
Git Repository	10	10	10	10	10	50
Scrum Spreadsheet	10	10	10	10	10	50
UML Diagrams	20	20	20	20	20	100
Final Package					300	300
Total	40	40	40	40	340	500

Implementation Advice

The FLTK manual is at http://fltk.org/doc-1.3/. A good overview of the FLTK widgets available for your use is at http://fltk.org/doc-1.3/common.html, with pre-defined dialogs at http://fltk.org/doc-1.3/group__group__comdlg.html. The FLTK installation also includes a LOT of example code and applications documented at http://fltk.org/doc-1.3/examples.html, and you'll find some excellent additional examples on Erco's FLTK Cheat Page at http://seriss.com/people/erco/fltk/ (kudos and props to Erco for providing this!).

It is best to implement incrementally, starting with easy GUI implementations to ensure you end up with a working product and then refactoring to add more sophisticated GUI features for a better grade. Thus, your first few GUI implementations may be very text-centric (CLI-in-windows, as it were).

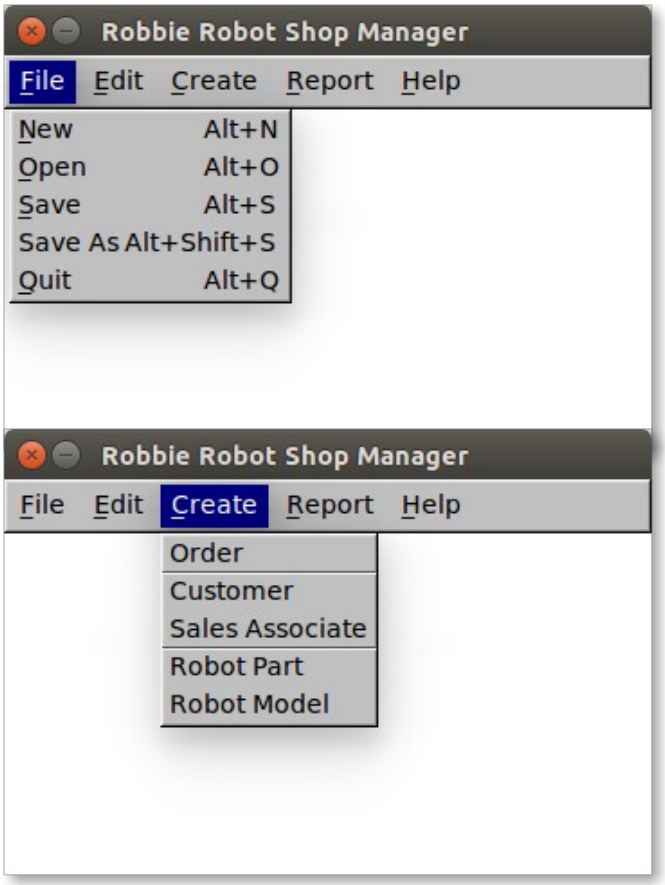
---

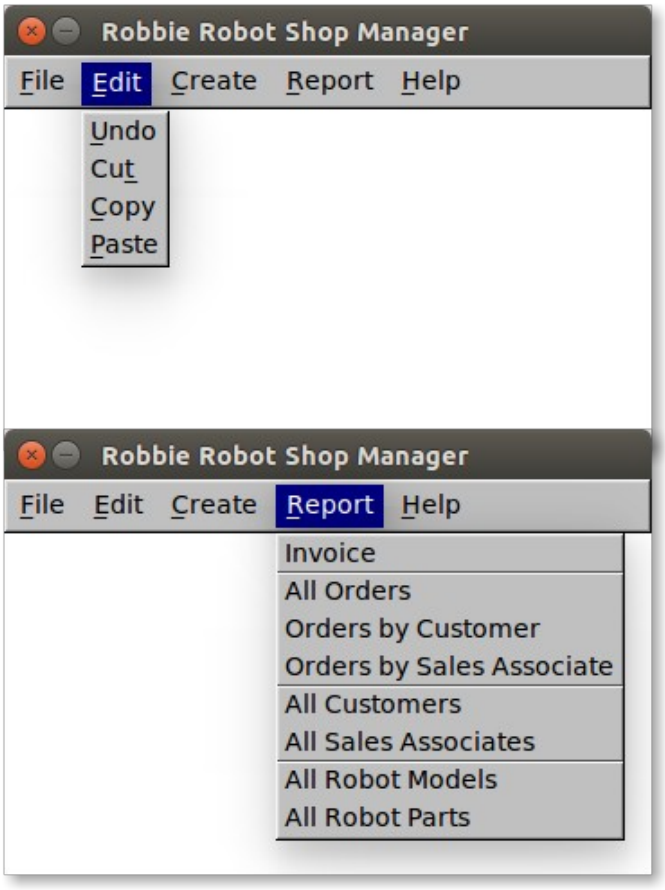
1 Add 6 Features per additional student for teams – 18 for 2-person teams, 24 for 3-person teams, etc. These numbers include Features implemented in the first 3 sprints for Homework #5.

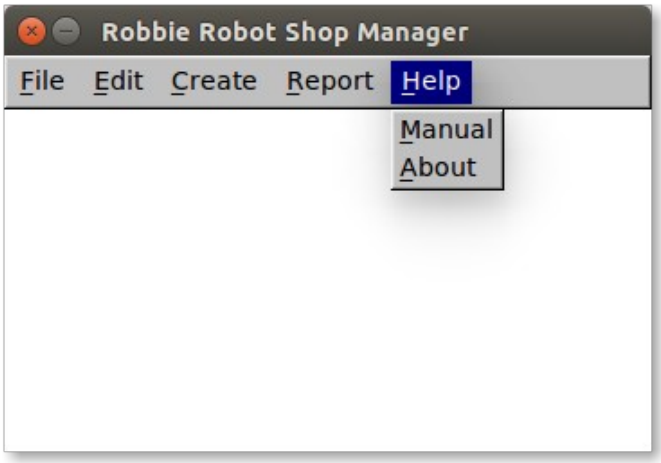
FLTK offers several candidate widgets to handle text-centric GUIs – consider http://fltk.org/doc-1.3/common.html#common_text, for example. For displaying text reports such as your initial list of Robot Models, Fl_Text_Display or Fl_Multiline_Output looks promising. For your initial implementation of creating Robot Parts and Robot Models, a series of Fl_Input dialogs would probably work with your existing code – just swap with the getline or cin calls.

In later sprints, we'd like to see a couple of custom dialogs added, for creating a Robot Part (perhaps with the field custom to each type – arm, head, etc. - in a 5-tabbed pane at the bottom) and a Robot Model. More visually appealing reports can be displayed in HTML by replacing Fl_Multiline_Output with Fl_Help_View.

Here is a possible menu design. **You should do your own menu design and UML modeling!**







The Fine Print

Robot images are used under license from Graphic Stock. Students may not under any circumstances use graphics provided along with this project for any other purpose, as such use would constitute copyright infringement. Remember, *we talked about this*!