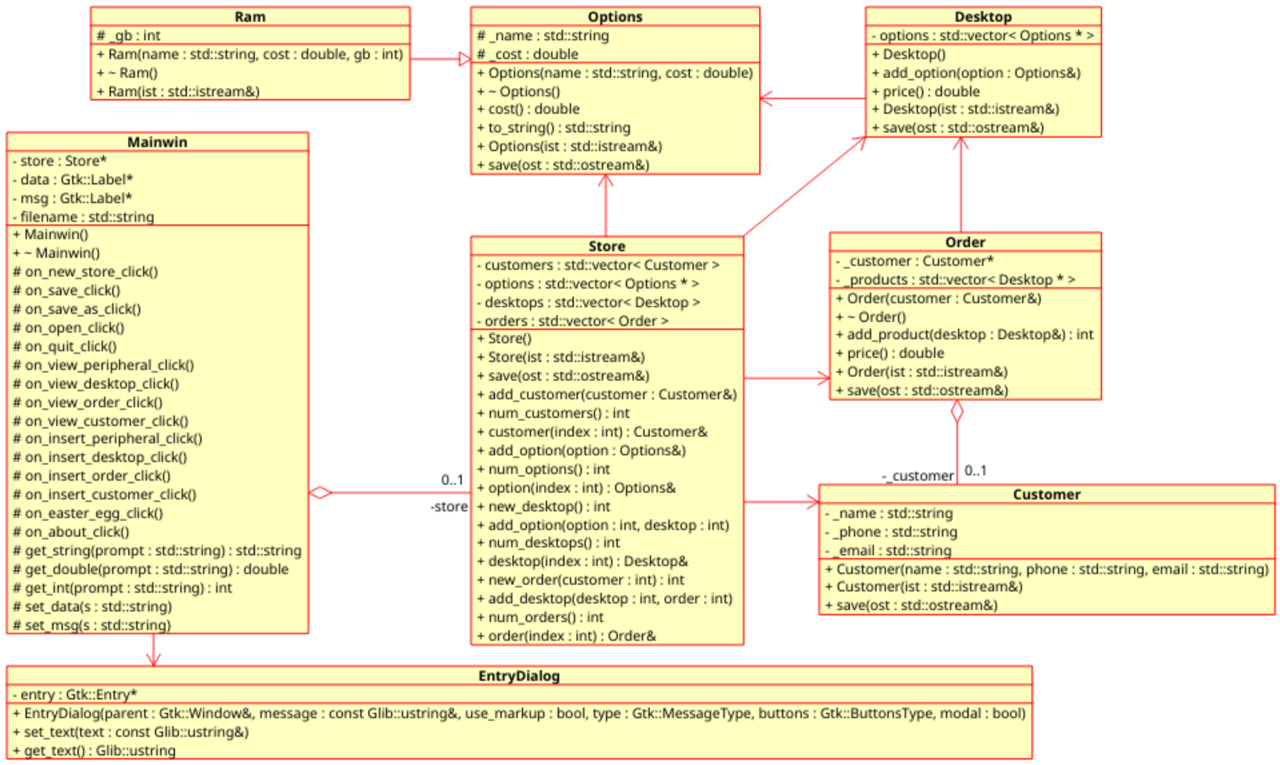
Sprint 4 Guidance

CSE 1325 – Spring 2020 – ELSA

IMPORTANT: Do NOT use full_credit, bonus, or extreme_bonus subdirectories for the final project. Keep all files to be graded in the cse1325/elsa directory on GitHub for the duration of the final project – source code in cse1325/elsa/src, and the spreadsheet in cse1325/elsa/doc.

For Sprint 4, you will add a custom dialog, a derived Options type, and any bonus features that you choose and for which you may receive extra credit. All bonus features will be graded at the conclusion of *Sprint 4*. This is it! No more sprints follow, ELSA will be *complete*!

This class diagram is for sprint 4 non-bonus features only. You must design any bonus features you choose to implement.


Ram
# _gb : int
+ Ram(name : std::string, cost : double, gb : int)
+ ~ Ram()
+ Ram(ist : std::istream&)
Options
# _name : std::string
# _cost : double
+ Options(name : std::string, cost : double)
+ ~ Options()
+ cost() : double
+ to_string() : std::string
+ Options(ist : std::istream&)
+ save(ost : std::ostream&)
Desktop
- options : std::vector< Options * >
+ Desktop()
+ add_option(option : Options&)
+ price() : double
+ Desktop(ist : std::istream&)
+ save(ost : std::ostream&)
Mainwin
- store : Store*
- data : Gtk::Label*
- msg : Gtk::Label*
- filename : std::string
+ Mainwin()
+ ~ Mainwin()
# on_new_store_click()
# on_save_click()
# on_save_as_click()
# on_open_click()
# on_quit_click()
# on_view_peripheral_click()
# on_view_desktop_click()
# on_view_order_click()
# on_view_customer_click()
# on_insert_peripheral_click()
# on_insert_desktop_click()
# on_insert_order_click()
# on_insert_customer_click()
# on_easter_egg_click()
# on_about_click()
# get_string(prompt : std::string) : std::string
# get_double(prompt : std::string) : double
# get_int(prompt : std::string) : int
# set_data(s : std::string)
# set_msg(s : std::string)
0..1
-store
Store
- customers : std::vector< Customer >
- options : std::vector< Options * >
- desktops : std::vector< Desktop >
- orders : std::vector< Order >
+ Store()
+ Store(ist : std::istream&)
+ save(ost : std::ostream&)
+ add_customer(customer : Customer&)
+ num_customers() : int
+ customer(index : int) : Customer&
+ add_option(option : Options&)
+ num_options() : int
+ option(index : int) : Options&
+ new_desktop() : int
+ add_option(option : int, desktop : int)
+ num_desktops() : int
+ desktop(index : int) : Desktop&
+ new_order(customer : int) : int
+ add_desktop(desktop : int, order : int)
+ num_orders() : int
+ order(index : int) : Order&
Order
- _customer : Customer*
- _products : std::vector< Desktop * >
+ Order(customer : Customer&)
+ ~ Order()
+ add_product(desktop : Desktop&) : int
+ price() : double
+ Order(ist : std::istream&)
+ save(ost : std::ostream&)
-_customer
0..1
Customer
- _name : std::string
- _phone : std::string
- _email : std::string
+ Customer(name : std::string, phone : std::string, email : std::string)
+ Customer(ist : std::istream&)
+ save(ost : std::ostream&)
EntryDialog
- entry : Gtk::Entry*
+ EntryDialog(parent : Gtk::Window&, message : const Glib::ustring&, use_markup : bool, type : Gtk::MessageType, buttons : Gtk::ButtonsType, modal : bool)
+ set_text(text : const Glib::ustring&)
+ get_text() : Glib::ustring

This sprint adds fewer lines of code than was needed for the previous 3 sprints, but it covers two unrelated concepts because we lost a sprint due to the extended spring break.

Scrum Spreadsheet

Allocate the features you intend to complete this sprint on your Product Backlog tab. You should at least allocate through those features with a "4" in the Required column, as these are the features that contribute toward your full credit grade at the end of this sprint.

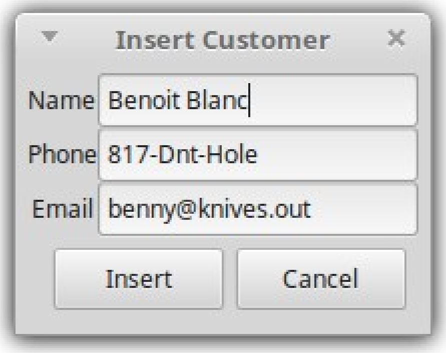
Then read through the suggestions below, and create your anticipated tasks on the Sprint 04 Backlog tab for each feature, setting the Feature ID of the feature that each task helps implement.

IMPORTANT: You may NOT simply baseline the complete Sprint 3 Suggested Solution for this sprint! If you have issues with the previous sprint's code, you may examine the Suggested Solution to help you bring your code up to where we are – just add tasks for this in your Sprint 04 Backlog.

Once ready, proceed with Sprint 4, marking each task as complete on the day you finish it.

Unified Dialog

Previously, when creating a new Customer object, you likely used 3 separate EntryDialog instances – one for name, one for phone number, and one for email. But users expect to have all of the data collected via a single form or dialog, something similar to what is shown to the right.



Implementing this dialog should only affect one method, Mainwin::on_insert_customer_click(). **Note that you are NOT required to replicate this dialog!** As long as the dialog follows the Principle of Least Astonishment and is able to create any valid Customer, your design is eligible to receive full marks.

The dialog shown uses a Gtk::Dialog with 3 label / entry pairs, laid out in a grid, with Insert and Cancel buttons via the add_button method. Data validation occurs when Insert is clicked, with error messages replacing the invalid data.

However, feel free to explore the available methods. For example, you might insert default data for the user via Gtk::Entry::set_text. You might show formats using Gtk::Entry::set_placeholder_text. You might use icons to visually identify the fields, with Gtk::Image or the Gtk::Entry::set_icon_from_ family of methods. Make this dialog your own!

Polymorphism

More significant to your existing code is to add one or more classes derived from Options. Ram (random access memory) is show, with an added _gb attribute to store the number of gigabytes. However, other options such as Cpu (with a _freq attribute) or Drive (with a _tb attribute, or perhaps hybrid with flash and rotating sizes) are perfectly acceptable. **The key point for full marks is to use polymorphism with an Options class hierarchy of your choice.**

Ram

The new Ram (or Cpu or Drive or whatever) class(es) inherits from the Options class, adding the _gb attribute (or other attibute(s) if you select a different peripheral).

You'll need to override the to_string method (to replace the string representation of the combined attributes of Options and Ram).

You'll also need to override the save(std::ostream&) method[1], which should delegate to Options::save(std::ostream&) and then also save _gb or other additional attributes. And, since

1 Note that Options::save was *not* originally defined as virtual, but you can't override a non-virtual method. Might want to add a task to your Sprint Backlog right now!

constructors do not inherit, you’ll need a Ram::Ram(std::ostream&) that also delegates to Options::Options(std::ostream&) and then load _gb et. al.

Do NOT write an operator<< overload for classes derived from Options. Instead, ensure that your to_string() method overrides Options::to_string(), and that your operator<< for Options calls to_string *polymorphically*.

Mainwin

When the user selects Insert > Peripheral, you will now need to determine *which type* of peripheral to create. You might accomplish this with a simple Gtk::Dialog with buttons – RAM, Other, and Cancel, for example. Or you might prefer a ComboBoxText complemented with OK and Cancel buttons. Or perhaps you’d prefer a flyout menu, so the user directly selects Insert > Peripheral > RAM or whatever. It’s your program – just make it usable!

Store

A more subtle issue is that your new file format will no longer be readable by your previous ELSA program versions – they can’t load Ram objects! How will you handle this?

Bonus Points

Bonus Features

To earn additional credit, you may implement additional features beyond those required. For example, adding other derived classes for disks or CPUs, replacing the other sequences of dialogs with custom dialogs, making the data area in the main window scrollable, adding product deprecation, and so on, are all listed as *approved* bonus features.

You will only receive bonus points for implementing bonus features actually listed in your Scrum spreadsheet and approved by the professor. A few pre-approved bonus features are provided in the Sprint 2 Scrum spreadsheet, with the *maximum* bonus points available for an *excellent* implementation in the Bonus column (less excellent implementations will receive appropriate partial credit). **You may suggest unique features** that you would like to implement, and if appropriate and in line with our educational objectives, I will approve them. **This allows you to earn credit for taking the project in your own unique direction.**

Our project is complete after Sprint 4 – **you many NOT implement additional bonus features for credit after Sprint 4!**

Class Presentation (via Video)

You will have one additional opportunity to earn extra credit after Sprint 4, however, by presenting your project to the class via video. This is usually a live presentation, but we must adapt with the times this semester. A *quality* presentation that provides a clear overview of the *unique* features of your ELSA implementation (we know the basics – we wrote one, too!), delivered to me via Canvas prior to Exam #3, can earn you up to 25 bonus points for your homework grade.

Note that Canvas limits video submissions to 500 MB, so you may need to use low resolution and keep your presentation brief and too the point!

I create the lecture videos[2] using free-as-in-liberty software – Open Broadcaster Software (OBS) and the OpenShot video editor. These are well-supported on Windows, Mac, and Linux, but you are welcome to use other video editing software to prepare and render your presentation as you please. Remember – quality and professionalism count! Please avoid giving me motion sickness.

What's Left

Our last assignment will be a **stand alone multi-threaded app**, and then we're on to **Exam #3** and wrapping it all up with a bow on top!

2 Yes, I'm an engineer, not an artist. Desperate times call for desperate measures!