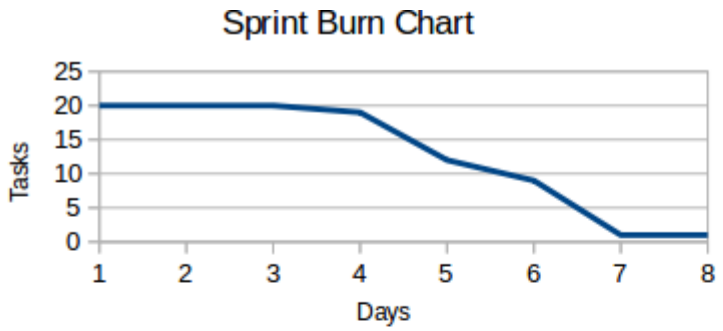
Sprint 2 Backlog

The Plan for the Suggested Solution

TBD

Remaining	Completed (this day)
20	
20	0
20	0
19	1
12	7
9	3
1	8
1	0



Feature ID	Assigned To	Description	Status	Notes
GUI		Create a main window (menu and tool bars)	Completed Day 3	Baseline from Nim
IGUI		Add item vectors to main window class	Completed Day 4	
IGUI		Add Mainwin::create_item dialog – select item type	Completed Day 4	
IGUI		Add Mainwin::create_item dialog – create item	Completed Day 4	
CS		Create Serving class	Completed Day 4	
CS		Create Mainwin::select_container() method	Completed Day 4	
CS		Create Mainwin::select_scoop() method	Completed Day 4	
CS		Create Mainwin::select_topping() method	Completed Day 4	
CS		Add Mainwin::create_serving() private method	Completed Day 5	
CS		Add operator<< for Item (use POLYMORPHISM)	Completed Day 5	
CS		Update regression tests to support operator<<	Completed Day 5	
CS		Create Mainwin::on_create_order_click() (but create serving)	Completed Day 6	
CS		Add operator<< for Serving	Completed Day 6	
CS		Show created serving temporarily on STDOUT	Completed Day 6	
CS		Add (temporary) Mainwin::_servings vector and push to it	Completed Day 6	
CS		Find icons	Completed Day 6	
CS		Find a logo	Completed Day 6	
CS		Create Help > About dialog to credit icon and logo authors	Completed Day 6	Baseline from CSE1325 Paint
CS		Add toolbar icons for Create Item and Create Order	Completed Day 6	
CS		Update class diagram		

The Main Window – mainwin.h

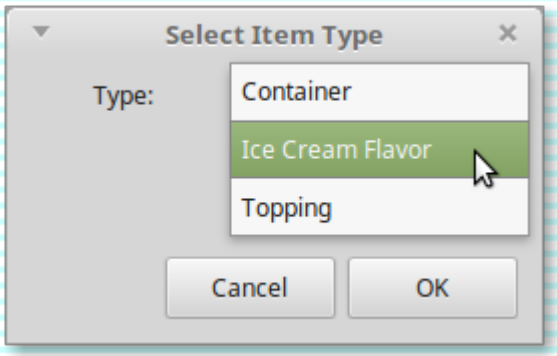
From a Nim baseline

```
#include "container.h"
#include "scoop.h"
#include "topping.h"
#include "serving.h"
#include <gtkmm.h>
#include <string>

class Mainwin : public Gtk::Window {
    public:
        Mainwin();
        virtual ~Mainwin();
    protected:
        void on_create_order_click();                   // Create a new order
        void on_create_item_click();                    // Create a new item
        void on_about_click();                          // Display About dialog
        void on_quit_click();                           // Exit the program
        void on_easteregg_click();                      // TODO: For test only
    private:
        Mice::Serving create_serving();                 // Create a new serving
        int select_container();                         // Select a container index
        int select_scoop();                             // Select a scoop index
        int select_topping();                           // Select a container index
        int select_from_vector (std::vector<std::string> names, std::string title);

        std::vector<Mice::Container> _containers;       // All defined containers
        std::vector<Mice::Scoop>     _scoops;           // All defined scoops
        std::vector<Mice::Topping>   _toppings;         // All defined toppings
        std::vector<Mice::Serving>   _servings;         // All defined servings
};
```

Mainwin::on_create_item_click()


```
void Mainwin::on_create_item_click() {

    Gtk::Dialog dialog_type{"Select Item Type", *this};
    const int WIDTH = 15;

    // Type
    Gtk::HBox b_type;

    Gtk::Label l_type{"Type:"};
    l_type.set_width_chars(WIDTH);
    b_type.pack_start(l_type, Gtk::PACK_SHRINK);

    // TODO: Replace this with 3 large, colorful buttons
    Gtk::ComboBoxText c_type;      c_type.set_size_request(WIDTH*10);
    const int CONTAINER = 0;       c_type.append("Container");
    const int SCOOP = 1;           c_type.append("Ice Cream Flavor");
    const int TOPPING = 2;         c_type.append("Topping");
    b_type.pack_start(c_type, Gtk::PACK_SHRINK);
    dialog_type.get_vbox()->pack_start(b_type, Gtk::PACK_SHRINK);

    // Show dialog_type
    dialog_type.add_button("Cancel", 0);
    dialog_type.add_button("OK", 1);
    dialog_type.show_all();
    if (dialog_type.run() != 1) {dialog_type.close(); return;}

    int type = c_type.get_active_row_number();

    dialog_type.close();
```

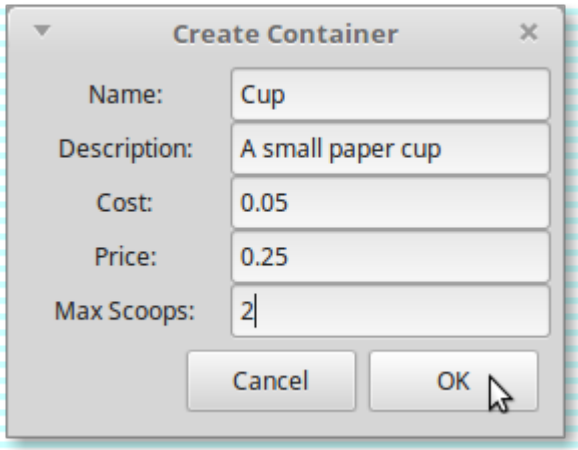
I chose to have a single “Create Item” command, and then ask the user what type of item to create.
It’s equally reasonable to have “Create Container”, “Create Flavor”, and “Create Topping” commands.



Mainwin::on_create_item_click()
2 of 4

```
Gtk::Dialog dialog;
if (type == CONTAINER) dialog.set_title("Create Container");
else if (type == SCOOP) dialog.set_title("Create Flavor");
else dialog.set_title("Create Topping");
dialog.set_transient_for(*this);

// Name (an Entry) ...
// Description (an Entry) ...
// Cost (an Entry) ...
// Price (an Entry) ...
// Max Scoops (an Entry for Container only)
Gtk::HBox b_max_scoops;

Gtk::Label l_max_scoops{"Max Scoops:"};
l_max_scoops.set_width_chars(WIDTH);
b_max_scoops.pack_start(l_max_scoops, Gtk::PACK_SHRINK);

Gtk::Entry e_max_scoops;
e_max_scoops.set_max_length(WIDTH*4);
b_max_scoops.pack_start(e_max_scoops, Gtk::PACK_SHRINK);
if (type == CONTAINER) {
    dialog.get_vbox()->pack_start(b_max_scoops, Gtk::PACK_SHRINK);
}

dialog.add_button("Cancel", 0);    dialog.add_button("OK", 1);
dialog.show_all();

bool valid_data = false;
double d_cost;    double d_price;    int i_max_scoops;
```

This is the usual dialog creation code, with the "if" packing Max Scoops only for the Container field. We'll run the dialog within a loop to on the next page to re-request bad data.



Mainwin::on_create_item_click()

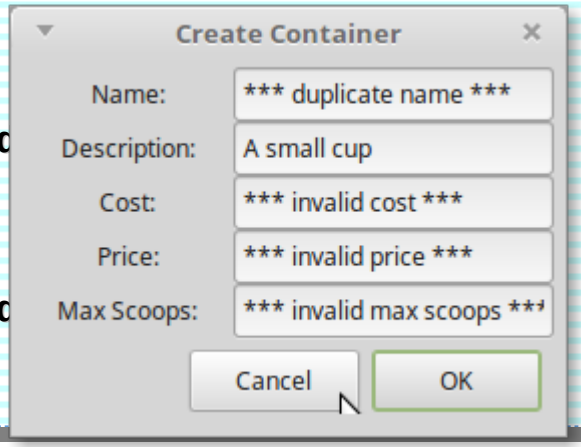
3 of 4

```
while(!valid_data) {valid_data = true;

    if (dialog.run() != 1) {dialog.close(); return;}

    try {d_cost = std::stod(e_cost.get_text());
    } catch(std::exception e) {e_cost.set_text("*** invalid cost ***");
                               valid_data = false;}
    try {d_price = std::stod(e_price.get_text());
    } catch(std::exception e) {e_price.set_text("*** invalid price ***");
                               valid_data = false;}
    if (type == CONTAINER) {
        try {i_max_scoops = std::stoi(e_max_scoops.get_text());
        } catch(std::exception e) {e_max_scoops.set_text("*** invalid max scoops ***");
            valid_data = false;}
    }
    for (Mice::Container c : _containers)
        if (c.name() == e_name.get_text()) {
            e_name.set_text("*** duplicate name ***"); valid_data = false;
        }
    for (Mice::Scoop s : _scoops)
        if (s.name() == e_name.get_text()) {
            e_name.set_text("*** duplicate name ***"); valid_d
        }
    for (Mice::Topping t : _toppings)
        if (t.name() == e_name.get_text()) {
            e_name.set_text("*** duplicate name ***"); valid_d
        }
}
```

Data Validation!

Catch invalid doubles / ints

Avoid duplicate names (confusing)



Mainwin::on_create_item_click()
4 of 4

```
    if (type == CONTAINER) {
        Mice::Container
            c{e_name.get_text(), e_desc.get_text(), d_cost, d_price, i_max_scoops};
        _containers.push_back(c);
        std::cout << c << std::endl;  // TODO: Temp (replace with dialog)

    } else if (type == SCOOP) {
        Mice::Scoop
            s{e_name.get_text(), e_desc.get_text(), d_cost, d_price};
        _scoops.push_back(s);
        std::cout << s << std::endl;  // TODO: Temp (replace with dialog)

    } else {
        Mice::Topping
            t{e_name.get_text(), e_desc.get_text(), d_cost, d_price, 0};
        _toppings.push_back(t);
        std::cout << t << std::endl;  // TODO: Temp (replace with dialog)
    }

    dialog.close();
}
```

Instance the new item

Serving class – serving.h

```
#include "container.h"
#include "scoop.h"
#include "topping.h"
#include <vector>

namespace Mice {
  class Serving {
    public:
      Serving(Container container);
      void add_scoop(Scoop scoop);
      void add_topping(Topping topping);
      Container container() const;
      std::vector<Scoop> scoops() const;
      std::vector<Topping> toppings() const;
      double cost() const;
      double price() const;
    private:
      Container _container;
      std::vector<Scoop> _scoops;
      std::vector<Topping> _toppings;
  };
}

std::ostream& operator<<(std::ostream& os, const Mice::Serving& serving);
```

Gtkmm has a Container class, too! Who knew? They have the Gtk namespace, so we created the Mice namespace to avoid naming confusion.

Construction

Getters

Calculations

Fields (Private Attributes)

Operator Overloading

Serving class – serving.cpp

```
namespace Mice {
    Serving::Serving(Container container) : _container{container} { }
    void Serving::add_scoop(Scoop scoop) {_scoops.push_back(scoop);}
    void Serving::add_topping(Topping topping) {_toppings.push_back(topping);}

    Container Serving::container() const {return _container;}
    std::vector<Scoop> Serving::scoops() const {return _scoops;}
    std::vector<Topping> Serving::toppings() const {return _toppings;}

    double Serving::cost() const {
        double total = _container.cost();
        for (Scoop scoop : _scoops) total += scoop.cost();
        for (Topping topping : _toppings) total += topping.cost();
        return total;
    }
    double Serving::price() const {
        double total = _container.price();
        for (Scoop scoop : _scoops) total += scoop.price();
        for (Topping topping : _toppings) total += topping.price();
        return total;
    }
}
std::ostream& operator<<(std::ostream& os, const Mice::Serving& serving) {
    os << serving.container();
    for (Mice::Scoop s : serving.scoops()) os << std::endl << s;
    for (Mice::Topping t : serving.toppings()) os << std::endl << t;
    return os;
}
```

Mainwin::create_serving()

```
Mice::Serving Mainwin::create_serving() {
    int container = select_container();
    if (container == -1) throw std::runtime_error("Canceled");

    Mice::Serving serving{_containers[container]};

    bool has_no_scoops = true;
    for (int i=0; i<_containers[container].max_scoops(); ++i) {
        int scoop = select_scoop();
        if (scoop == -1) break;
        serving.add_scoop(_scoops[scoop]);
        has_no_scoops = false;
    }
    if (has_no_scoops) throw std::runtime_error("Canceled");

    while (true) {
        int topping = select_topping();
        if (topping == -1) break;
        else serving.add_topping(_toppings[topping]);
    }

    return serving;
}
```

Pick a container and instance the serving

Select 1 or more scoops; stop on Cancel or when the container is full

Select 0 or more toppings; stop on Cancel

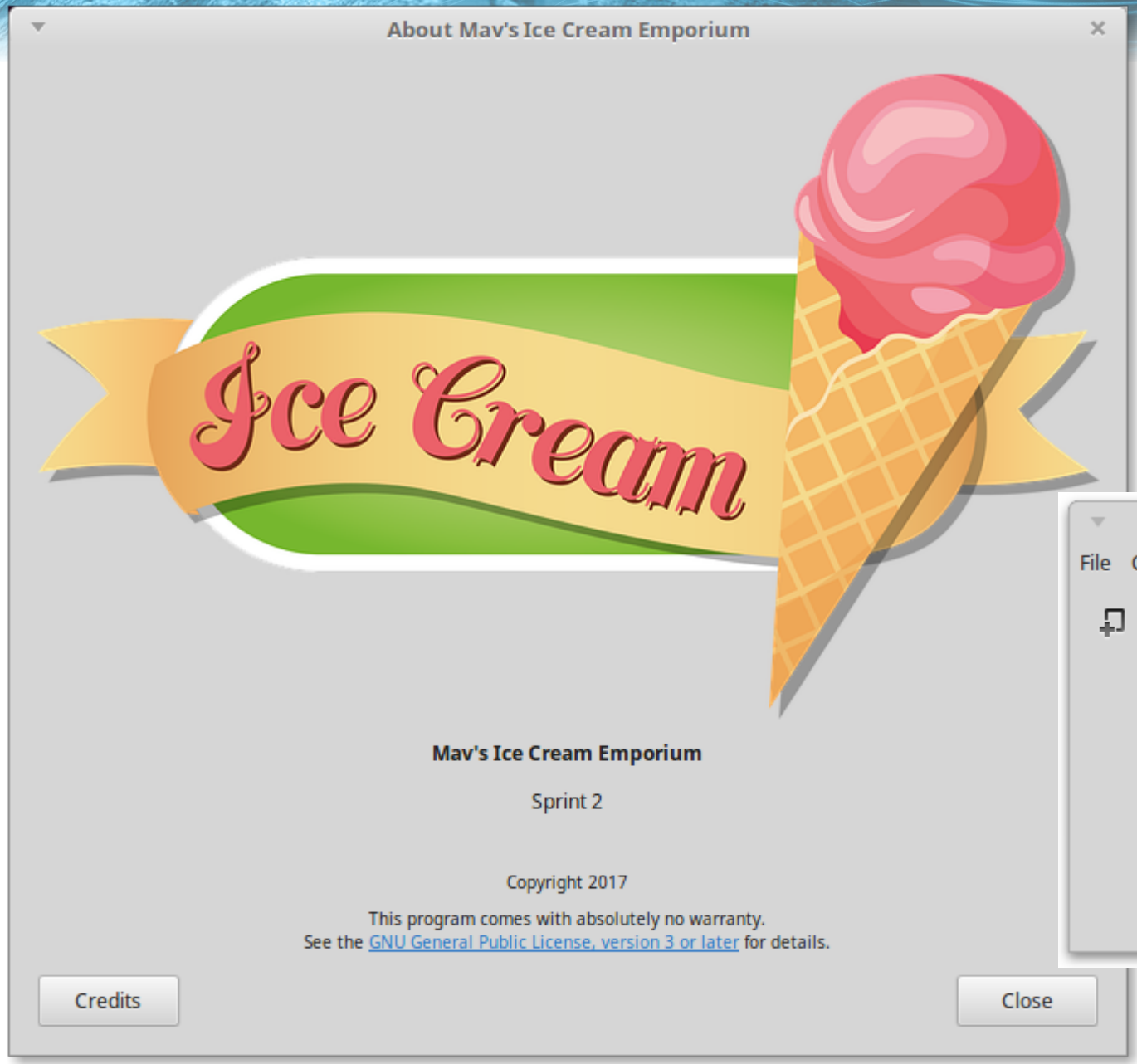
We note a minor problem: Once the user has selected at least one scoop, they can no longer cancel the order. Perhaps we should add a Done button? Should we also provide Back (and Forward)? Quantity of this serving?

Mainwin::on_create_order_click() and an Easter Egg

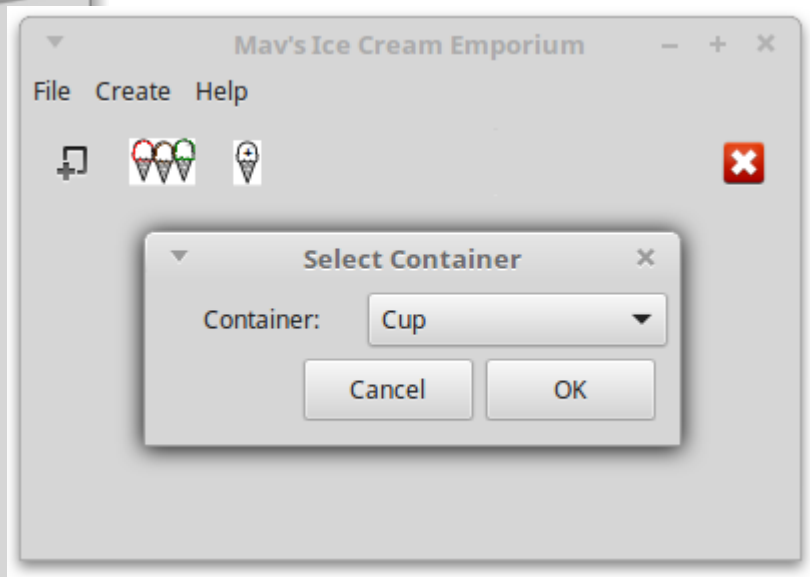
```
void Mainwin::on_create_order_click() {
    try {
        // Create a new serving (NOT an order yet!)
        Mice::Serving serving = create_serving();
        _servings.push_back(serving); // TODO: Temporary until _orders is created
        std::cout << serving << std::endl;  // TODO: Temporary - replace with dialog
    } catch(std::runtime_error e) { } // User canceled order
}
void Mainwin::on_easteregg_click() {
        _containers.push_back(
            Mice::Container{"Cone", "Crispy airy delight", 0.25, 0.50, 2});
        _containers.push_back(
            Mice::Container{"Waffle Cone", "Crunchy wrapped waffle cake", 0.35, 0.75, 3});
        _scoops.push_back(
            Mice::Scoop{"Chocolate", "Rich smooth chocolate ice cream", 0.20, 0.50});
        _scoops.push_back(
            Mice::Scoop{"Vanilla", "Real vanilla bean ice cream", 0.20, 0.50});
        _scoops.push_back(
            Mice::Scoop{"Strawberry", "Chunks of strawberry in vanilla", 0.20, 0.50});
        _toppings.push_back(
            Mice::Topping{"Cherry", "Classic marichino cherry", 0.1, 0.2, 0});
        _toppings.push_back(
            Mice::Topping{"Chocolate Sauce", "Rich thick chocolate", 0.1, 0.25, 0});
        _toppings.push_back(
            Mice::Topping{"Whipped Cream", "Sweet cream perfection", 0.1, 0.2, 0});
        // Display acknowledgement
        Std::cout << "Added 2 containers, 3 scoops, and 3 toppings" << std::endl;
}
```

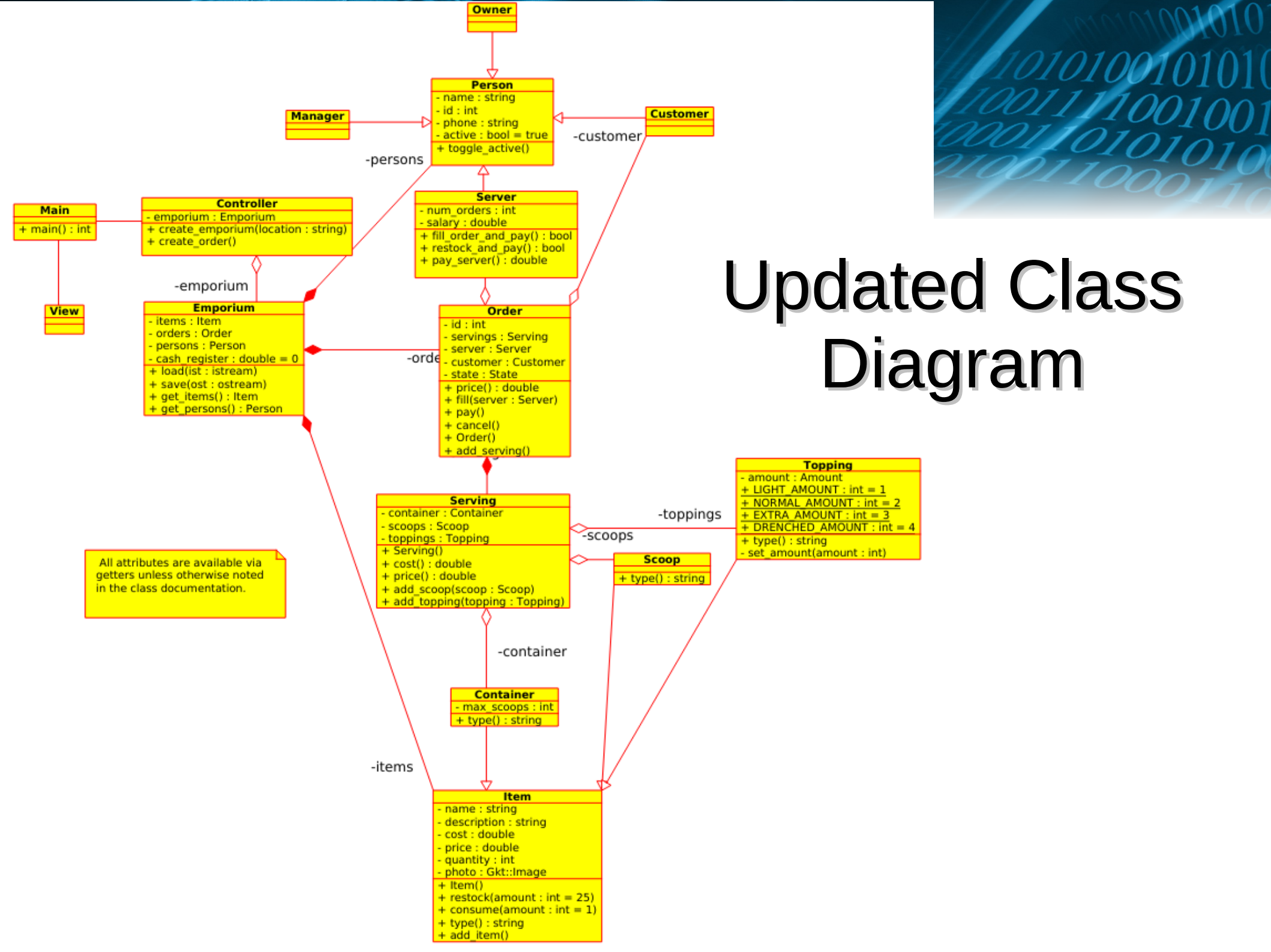
This just creates a serving for now

Miscellany




```
          Container: Cup $0.25
       Scoop: Strawberry $0.50
          Scoop: Vanilla $0.50
        Scoop: Chocolate $0.50
Topping: Chocolate Sauce $0.25
          Topping: Cherry $0.20
```





Updated Class Diagram