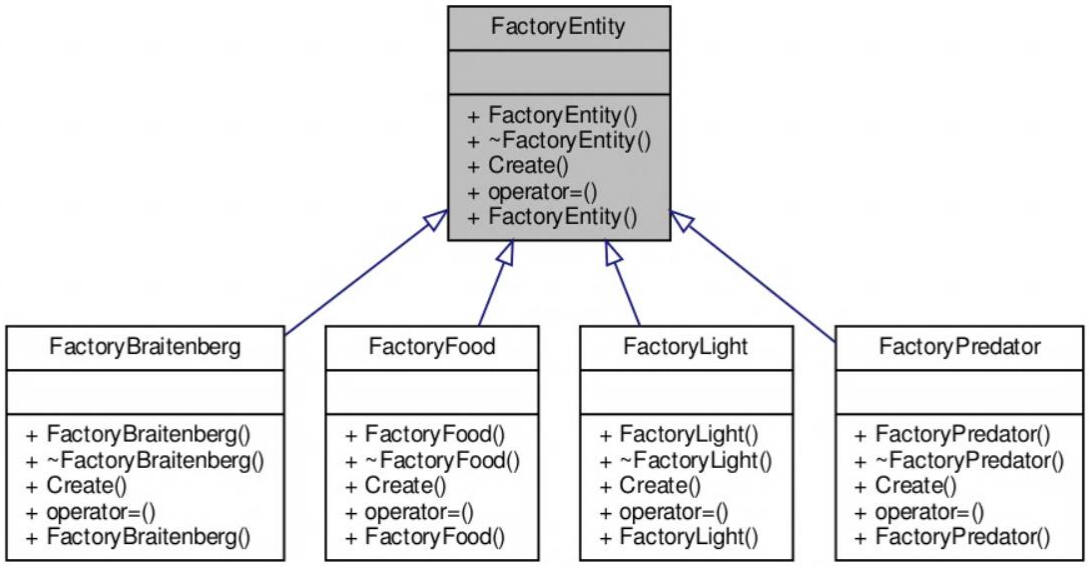
Implementation of Predator Class:



***I was unable to correctly implement the observer pattern, but I did get a start on it.

graphics_arena_viewer.cc

```
/************* BEGIN TEXTBOX GRAPHICS **************/
 robotWidgets.push_back(new nanogui::Label(
   panel, "Wheel Velocities", "sans-bold"));
 nanogui::Widget* grid = new nanogui::Widget(panel);
 // A grid with 3 columns
 grid->setLayout(
   new nanogui::GridLayout(nanogui::Orientation::Horizontal, 3,
   nanogui::Alignment::Middle, /*int margin = */0, /*int spacing = */0));
 robotWidgets.push_back(grid);

 // Columns Headers Row
 // Notice that it is assigning these items to grid locations row by row
 new nanogui::Label(grid, "", "sans-bold");
 new nanogui::Label(grid, "Left", "sans-bold");
 new nanogui::Label(grid, "Right", "sans-bold");

 // Next Row for wheel velocities from light behavior
 new nanogui::Label(grid, "Light", "sans-bold");
 light_value_left_ = new nanogui::TextBox(grid, "0.0");
 light_value_left_->setFixedWidth(75);
 light_value_right_ = new nanogui::TextBox(grid, "0.0");
 light_value_right_->setFixedWidth(75);
 new nanogui::Label(grid, "Food", "sans-bold");
 food_value_left_ = new nanogui::TextBox(grid, "0.0");
```

```
 food_value_left_->setFixedWidth(75);
 food_value_right_ = new nanogui::TextBox(grid, "0.0");
 food_value_right_->setFixedWidth(75);
 new nanogui::Label(grid, "Robot", "sans-bold");
 bv_value_left_ = new nanogui::TextBox(grid, "0.0");
 bv_value_left_->setFixedWidth(75);
 bv_value_right_ = new nanogui::TextBox(grid, "0.0");
 bv_value_right_->setFixedWidth(75);
 // Save these text boxes so they can be filled with values in real-time
 // Using the observer pattern
/* my_velocity_container_light_.StoreBoxes(
   light_value_left_, light_value_right_);*/

 entitySelect->setCallback(
   [this, isMobile, robotWidgets, lightBehaviorSelect,
   foodBehaviorSelect, BVBehaviorSelect](int index) {
     if (index > 0/* Already observing a robot */) {
       // Unsubscribe from that one
       // ArenaEntity* entity = arena_get_entities()[index];
      // bv_value_left_->setValue("1.0");
     }
     // ...
   /*  if (entity->get_type() == kBraitenberg) {
       // ...
       // Subscribe to observe this one
     } */
 });
/************** END TEXTBOX GRAPHICS ***************/



// EXAMPLE of setting the value to be displayed in the velocity grid
void GraphicsArenaViewer::SomeFunction(WheelVelocity* light_wv_ptr,
  WheelVelocity* food_wv_ptr, WheelVelocity* bv_wv_ptr) {
int i = light_wv_ptr->left;
  std::string out_string;
  std::stringstream ss;
  ss << i;
  out_string = ss.str();
  light_value_left_->setValue(out_string);
  i = light_wv_ptr->right;
  ss <<i;
  out_string == ss.str();
  light_value_right_->setValue(out_string);
  i = food_wv_ptr->left;
  ss << i;
  out_string = ss.str();
  food_value_left_->setValue(out_string);
```

```
 i = food_wv_ptr->right;
 ss <<i;
 out_string == ss.str();
 food_value_right_->setValue(out_string);
 i = bv_wv_ptr->left;
 ss << i;
 out_string = ss.str();
 bv_value_left_->setValue(out_string);
 i = bv_wv_ptr->right;
 ss <<i;
 out_string == ss.str();
 bv_value_right_->setValue(out_string);
    screen()->performLayout();

 /*my_velocity_container_light_.left_->
  setValue(formatValue(wv.left_))*/
}
```

braitenberg_vehicle.cc

```
/*void BraitenbergVehicle::Notify(GraphicsArenaViewer* gav_observer,
WheelVelocity* light_wv_ptr, WheelVelocity* food_wv_ptr,
WheelVelocity* bv_wv_ptr){
 gav_observer->SomeFunction(light_wv_ptr, food_wv_ptr, bv_wv_ptr);
} */
void BraitenbergVehicle::Update() {
 WheelVelocity* light_wv_ptr = new WheelVelocity();
 WheelVelocity* food_wv_ptr = new WheelVelocity();
 WheelVelocity* bv_wv_ptr = new WheelVelocity();
 food_behavior_ptr_->getWheelVelocity(
  get_sensor_reading_left(closest_food_entity_),
  get_sensor_reading_right(closest_food_entity_),
  defaultSpeed_, food_wv_ptr);

 light_behavior_ptr_->getWheelVelocity(
  get_sensor_reading_left(closest_light_entity_),
  get_sensor_reading_right(closest_light_entity_),
  defaultSpeed_, light_wv_ptr);

 bv_behavior_ptr_->getWheelVelocity(
  get_sensor_reading_left(closest_bv_entity_),
  get_sensor_reading_right(closest_bv_entity_),
  defaultSpeed_, bv_wv_ptr);
//  need this to be conditional
//  BraitenbergVehicle::Notify(gav_observer, light_wv_ptr,
//   food_wv_ptr, bv_wv_ptr);
...
```