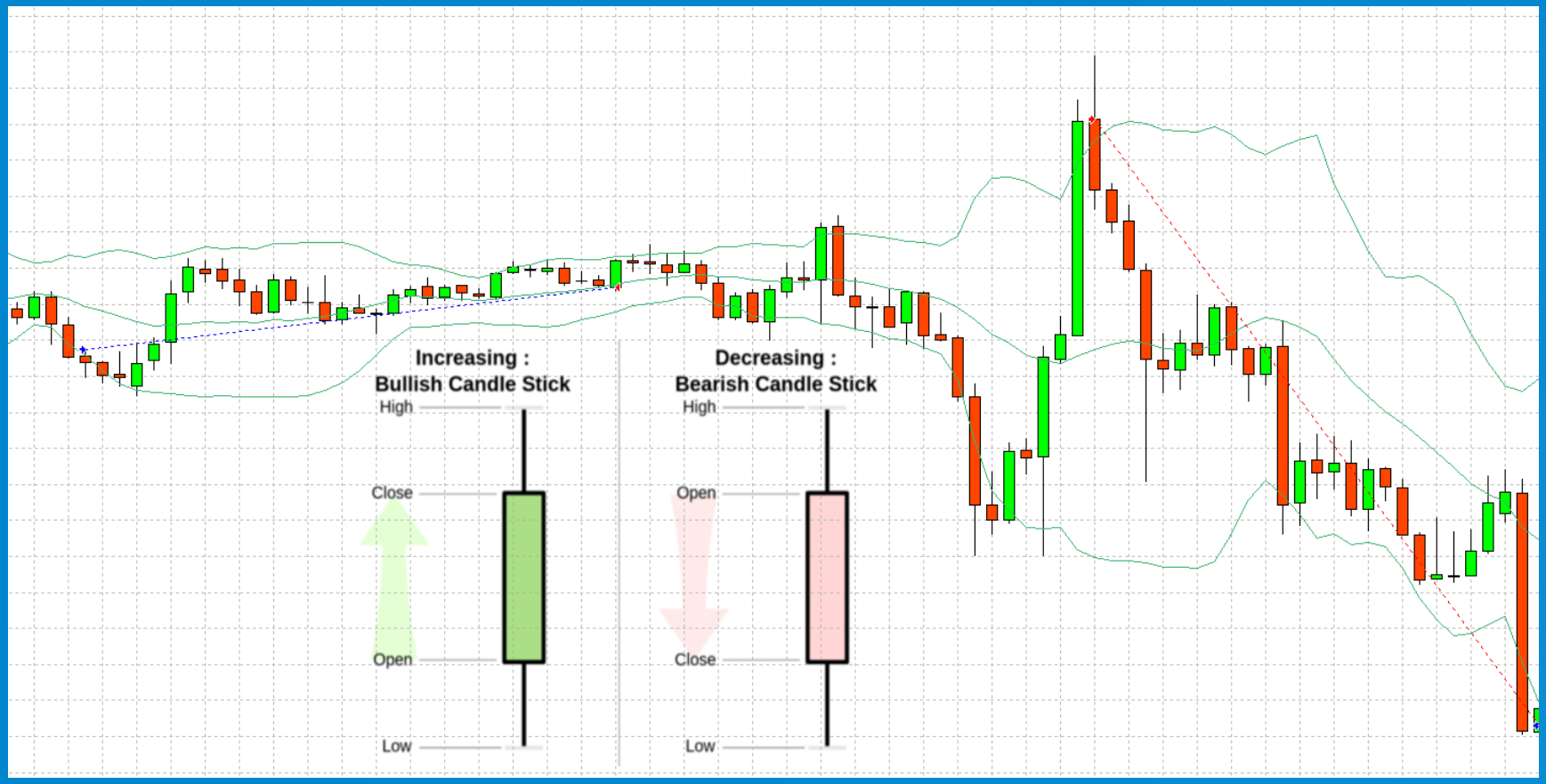
Lightning Talk:

Creating a composable class

With C++14 variadic templates

Ray Burgemeestre

Candle Stick / OHLC chart



Green lines are a Bollinger Bands indicator

Prices

```
#include <cmath>
#include <iostream>

int main() {
  for (time_t time = 1; time <= 1000; ++time) {
    double bid = fabs(sin(time / 30.0));
    double ask = bid + 0.005;
    std::cout << time << " - " << bid << ", " << ask
              << "\n";
  }
}
```

Prices

```
$ g++ --std=c++14 main.cpp -o main && ./main
1 - 0.841471, 0.846471
2 - 0.909297, 0.914297
3 - 0.14112, 0.14612
4 - 0.756802, 0.761802
5 - 0.958924, 0.963924
6 - 0.279415, 0.284415
7 - 0.656987, 0.661987
8 - 0.989358, 0.994358
9 - 0.412118, 0.417118
10 - 0.544021, 0.549021
11 - 0.99999, 1.00499
12 - 0.536573, 0.541573
13 - 0.420167, 0.425167
14 - 0.990607, 0.995607
15 - 0.650288, 0.655288
16 - 0.287903, 0.292903 ...
```

Desired interface 1/3

```
candlesticks chart;
for (...) {
  ...
  chart.add(time, bid, ask);
  if (!chart.initialized()) continue;

  // do something
  auto current_close = chart.get(0).close;
  auto previous_close = chart.get(0).close;
  auto predicted_close = chart.get(1).forecast;
  auto current_prediction = chart.get(0).forecast;
  ...
}
```

Desired interface 2/3

```
// OHLC
chart.get(shift).open
chart.get(shift).high
chart.get(shift).low
chart.get(shift).close

// moving average
chart.get(shift).ma

// bollinger bands
chart.get(shift).upper
chart.get(shift).main
chart.get(shift).lower

// some complicated model
chart.get(shift).predicition
...
```

Desired interface 3/3

```
candlesticks<ohlc, bollinger> chart;
...
```

Step 1

```
class ohlc {
public:
  double open;
  double high;
  double low;
  double close;
};

class bollinger {
public:
  double main;
  double upper;
  double lower;
};

class candlestick : public ohlc, public bollinger {};
```

```
class candlestick : public ohlc, public bollinger {};

candlestick test;
```

```
class candlestick : public ohlc, public bollinger {};

candlestick test;
```

```
template<class... T>
class candlestick : public T... {};

candlestick<ohlc, bollinger> test;
```

```
class candlestick : public ohlc, public bollinger {};

candlestick test;
```

```
template<class... T>
class candlestick : public T... {};

candlestick<ohlc, bollinger> test;
```

```
template<class... T>
class candlesticks {
public:
  bool initialized();
  void add(time_t time, double bid, double ask);
  candlestick<T...> get(size_t shift);
private:
  void create_candlestick_per_interval(time_t time);
  std::deque<candlestick<T...>> data_;
  time_t current_period_ = -1;
};
```

initialized function

```
template<class... T>
bool candlesticks<T...>::initialized() {
  return data_.size() >= 10;
}
```

get function

```
template <class... T>
candlestick<T...> candlesticks<T...>::get(size_t shift) {
  return data_[(data_.size() - 1) - (shift % 10)];
}
```

add function

```
template <class... T>
void candlesticks<T...>::add(time_t time,
                             double bid,
                             double ask) {
  // ???
}
```

```
class ohlc {
public:
  double open;
  double high;
  double low;
  double close;
};
```

add function

```
template <class... T>
void candlesticks<T...>::add(time_t time,
                             double bid,
                             double ask) {
  // call update here on all base classes
}
```

```
class ohlc {
public:
  double open;
  double high;
  double low;
  double close;

  void update(double bid, double ask);
};
```

ohlc update

```
class ohlc {
public:
  double open = 0;
  double high = std::numeric_limits<double>::min();
  double low  = std::numeric_limits<double>::max();
  double close = 0;

  void update(double bid, double ask);
};

void ohlc::update(double bid, double ask) {
  if (open == 0) open = bid;
  high = std::max(bid, high);
  low = std::min(bid, low);
  close = bid;
}
```

bollinger update

```
class bollinger {
public:
  double main;
  double upper;
  double lower;

  void update(double bid, double ask);
};

void bollinger::update(double bid, double ask) {
  // TODO:
  // main = moving average past 10 close prices
  // upper = main +  2 times standard deviation
  // lower = main -  2 times standard deviation
}
```

add function

```
template <class... T>
void candlesticks<T...>::add(time_t time,
                             double bid,
                             double ask) {
  create_candlestick_per_interval(time);
  data_.back().update(bid, ask);  // ambiguous!
}
```

add function

```
template <class... T>
void candlesticks<T...>::add(time_t time,
                             double bid,
                             double ask) {
  create_candlestick_per_interval(time);
  data_.back().update(bid, ask);  // ambiguous!
}
template <class... T>
void candlesticks<T...>::create_candlestick_per_interval(
    time_t time) {
  time_t period = time / 15;
  if (current_period_ != period) {
    data_.push_back({});
    if (data_.size() > 10) {
      data_.pop_front();
    }
    current_period_ = period;
```

candlestick update function

```
template<class... T> class candlestick : public T... {
public:
  void update(double bid, double ask)
  {
      // ???
  }
};
```

candlestick update function

```
template<class... T> class candlestick : public T... {
public:
  void update(double bid, double ask)
  {
    update<T...>(bid, ask);
  }

  template <class head, class... tail>
  void update(double bid, double ask)
  {
    static_cast<head *>(this)->update(bid, ask);
    update<tail...>(bid, ask);
  }
};
```

```
main.cpp: In instantiation of ‘void candlestick<T>::update
main.cpp:59:20:   required from ‘void candlestick<T>::upda
main.cpp:46:17:   required from ‘void candlestick<T>::upda
main.cpp:90:3:   required from ‘void candlesticks<T>::add(
main.cpp:99:29:   required from here
main.cpp:59:20: error: no matching function for call to ‘c
     update<tail...>(bid, ask);
     ~~~~~~~~~~~~~~~^~~~~~~~~~
main.cpp:56:8: note: candidate: template<class head, class
   void update(double bid, double ask)
        ^~~~~~
main.cpp:56:8: note:   template argument deduction/substit
main.cpp:59:20: note:   couldn't deduce template parameter
     update<tail...>(bid, ask);
     ~~~~~~~~~~~~~~~^~~~~~~~~~
```

```
template<class... T> class candlestick : public T... {
public:
  void update(double bid, double ask)
  {
    update<T...>(bid, ask);
  }

  template <class head>
  void update (double bid, double ask) {
    static_cast<head *>(this)->update(bid, ask);
  }

  template <class head, class... tail>
  void update(double bid, double ask)
  {
    static_cast<head *>(this)->update(bid, ask);
    update<tail...>(bid, ask);
  }
};
```

```
main.cpp: In instantiation of ‘void candlestick<T>::update
main.cpp:46:17:   required from ‘void candlestick<T>::upda
main.cpp:90:3:   required from ‘void candlesticks<T>::add(
main.cpp:99:29:   required from here
main.cpp:59:20: error: call of overloaded ‘update(double&,
     update<tail...>(bid, ask);
     ~~~~~~~~~~~~~~~~~^~~~~~~~~~
main.cpp:50:8: note: candidate: void candlestick<T>::updat
   void update (double bid, double ask) {
        ^~~~~~
main.cpp:56:8: note: candidate: void candlestick<T>::updat
   void update(double bid, double ask)
        ^~~~~~
```

```
template <class... T>
class candlestick : public T... {
public:
  void update(double bid, double ask) {
    update<T...>(bid, ask);
  }

  template <class head>
  void update(double bid, double ask) {
    static_cast<head *>(this)->update(bid, ask);
  }

  template <class head, class... tail>
  typename std::enable_if<0 != sizeof...(tail)>::type
  update(double bid, double ask) {
    static_cast<head *>(this)->update(bid, ask);
    update<tail...>(bid, ask);
  }
};
```

The result

```
$ g++ --std=c++14 main.cpp -o main && ./main
268 Buy at: 0.4769
308 Close at: 0.745902
```

```
if (chart.get(1).close > 0.75 &&
    chart.get(0).close < 0.50 && !in_trade
){
  std::cout << time << " Buy at: " << ask << "\n";
  in_trade = true;
} else if (ask > 0.75 && in_trade) {
  std::cout << time << " Close at: " << bid << "\n";
  break;
}
```

Resources

- Book: 12.4.2 - Where Can Pack Expansions Occur?
- [C++Now 2018: Odin Holmes “C++ Mixins: Customization Through Compile Time Composition”]

Link to source code

- https://bitbucket.org/rayburgemeestre/lightning-talk2/src