Trade Stat Generator

Yecheng Li

Jun 12, 2015

0. Overview

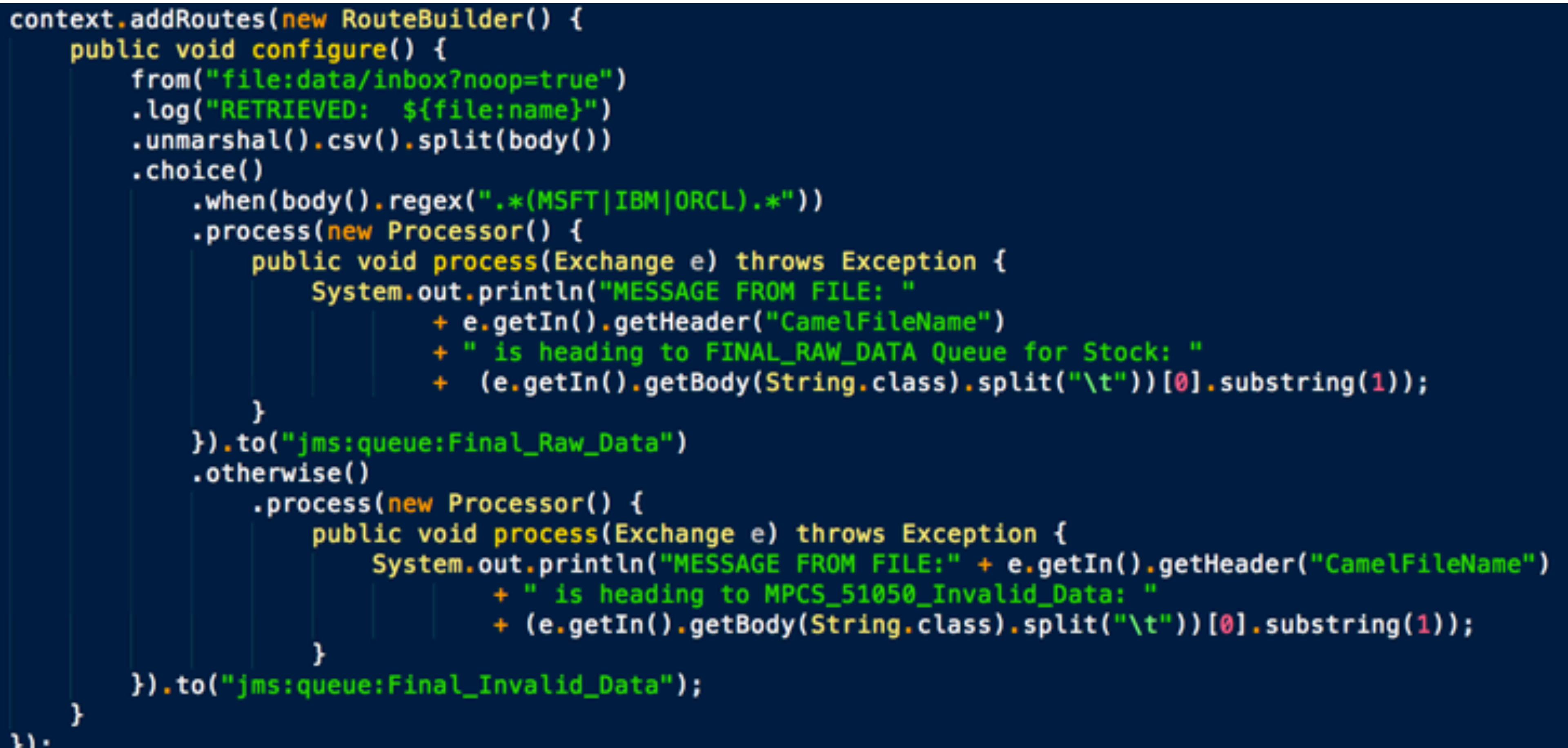
- Design patters
  - Singleton (ReportEngine)
  - Component (StockStat)
  - Composite (Portfolio)
  - Iterator (StockIterator)
  - Strategy (Strategy, MeanStat, StdDeviationStat, VarianceStat)

0. Overview

- Enterprise Integration Pattern
  - Message (Producer, Consumer, Subscriber)
  - Message Broker (Producer, Consumer, Subscriber - ActiveMQ)
  - Selective Consumer (Consumer)
  - Datatype Channel (Producer, Consumer, Subscriber – jms:queue/topic)
    - Publish-Subscribe channel
  - Endpoint (Producer, Consumer, Subscriber – from/to)
  - Message Translator (Producer, Consumer, Subscriber - Processor)

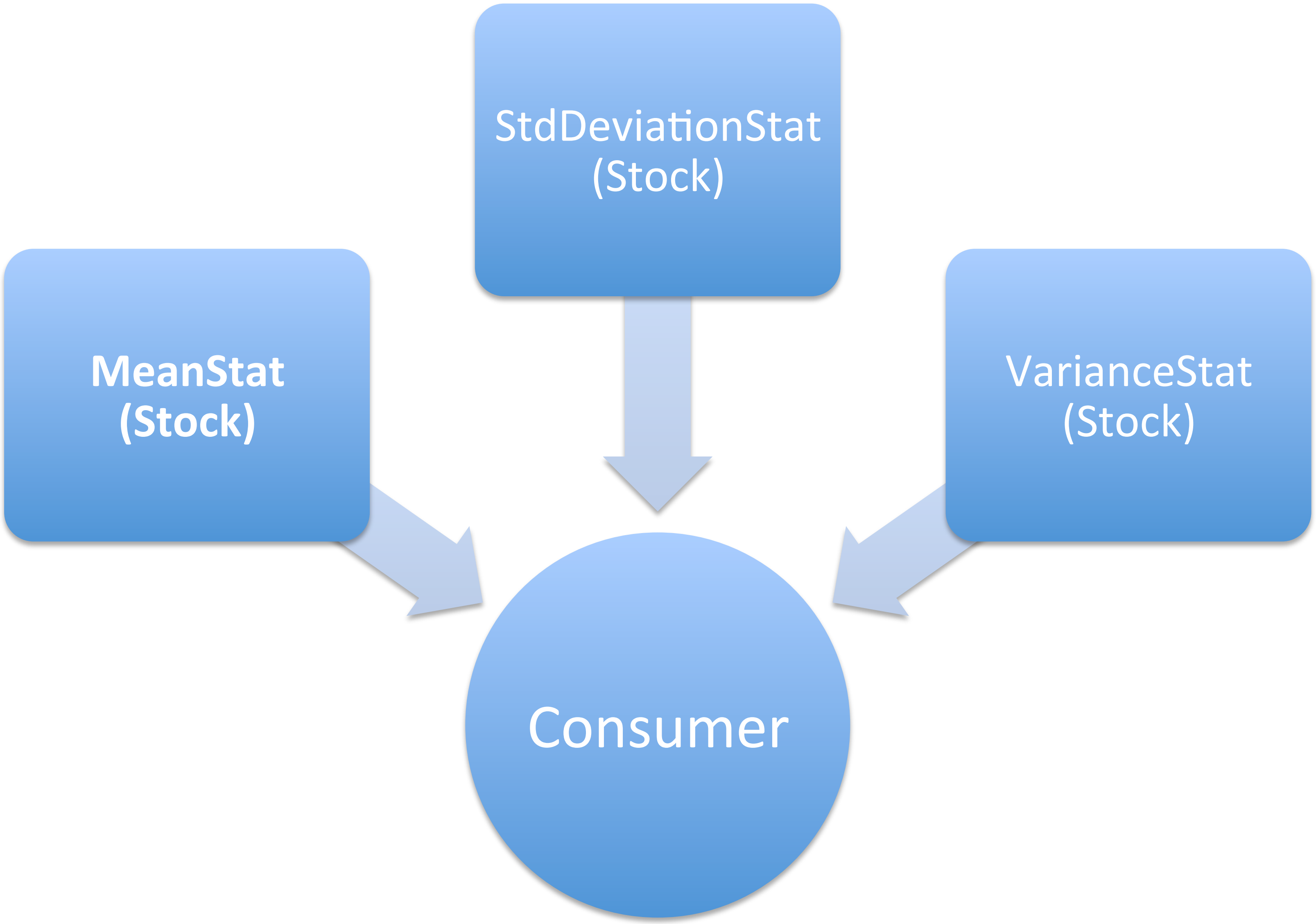
1. Producer - Overview

- If valid, add to "Final_Raw_Data"
- Otherwise, add to "Final_Invalid_Data" (invalid data added to test this functionality)

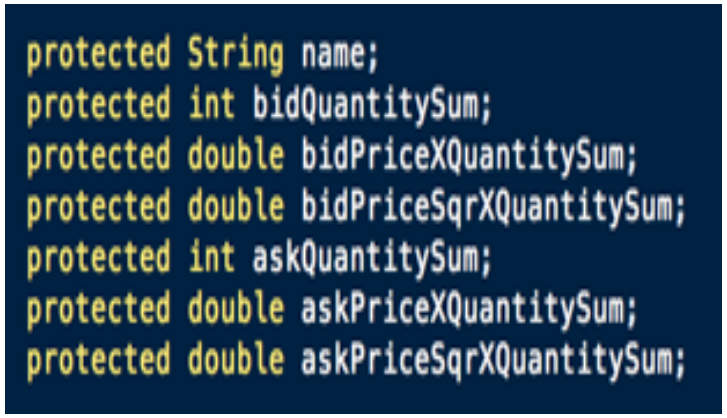


```
context.addRoutes(new RouteBuilder() {
    public void configure() {
        from("file:data/inbox?noop=true")
        .log("RETRIEVED:  ${file:name}")
        .unmarshal().csv().split(body())
        .choice()
            .when(body().regex(".*(MSFT|IBM|ORCL).*"))
            .process(new Processor() {
                public void process(Exchange e) throws Exception {
                    System.out.println("MESSAGE FROM FILE: "
                            + e.getIn().getHeader("CamelFileName")
                            + " is heading to FINAL_RAW_DATA Queue for Stock: "
                            +  (e.getIn().getBody(String.class).split("\t"))[0].substring(1));
                }
            }).to("jms:queue:Final_Raw_Data")
            .otherwise()
                .process(new Processor() {
                    public void process(Exchange e) throws Exception {
                        System.out.println("MESSAGE FROM FILE:" + e.getIn().getHeader("CamelFileName")
                                + " is heading to MPCS_51050_Invalid_Data: "
                                + (e.getIn().getBody(String.class).split("\t"))[0].substring(1));
                    }
        }).to("jms:queue:Final_Invalid_Data");
    }
});
```

2. Consumer - Overview



2. Consumer - Stock



```
protected String name;
protected int bidQuantitySum;
protected double bidPriceXQuantitySum;
protected double bidPriceSqrXQuantitySum;
protected int askQuantitySum;
protected double askPriceXQuantitySum;
protected double askPriceSqrXQuantitySum;
```

addTick(String message)

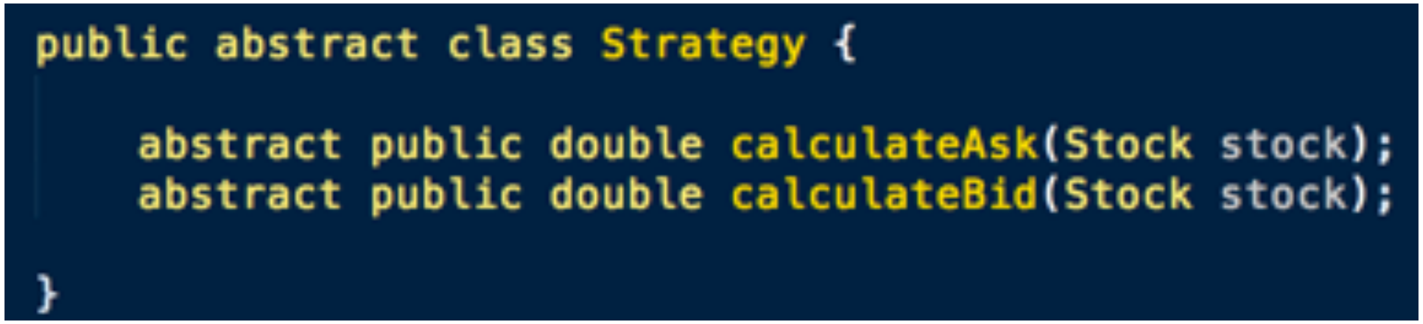
- Parse messages retrieved from Final Raw_Data
- Update the properties when a new tick is added (increase quantity sum, PriceXQuantity Sum, PriceSqrXQuantity Sum)

getAsk(Strategy index)

getBid(Strategy index)

- Getter method for ask/bid related statistical indices

2. Consumer - Strategy



```
public abstract class Strategy {

    abstract public double calculateAsk(Stock stock);
    abstract public double calculateBid(Stock stock);

}
```

MeanStat

- calculateAsk()
- calculateBid()

StdDeviationStat

- calculateAsk()
- calculateBid()

VarianceStat

- calculateAsk()
- calculateBid()

2. Consumer - Consumer

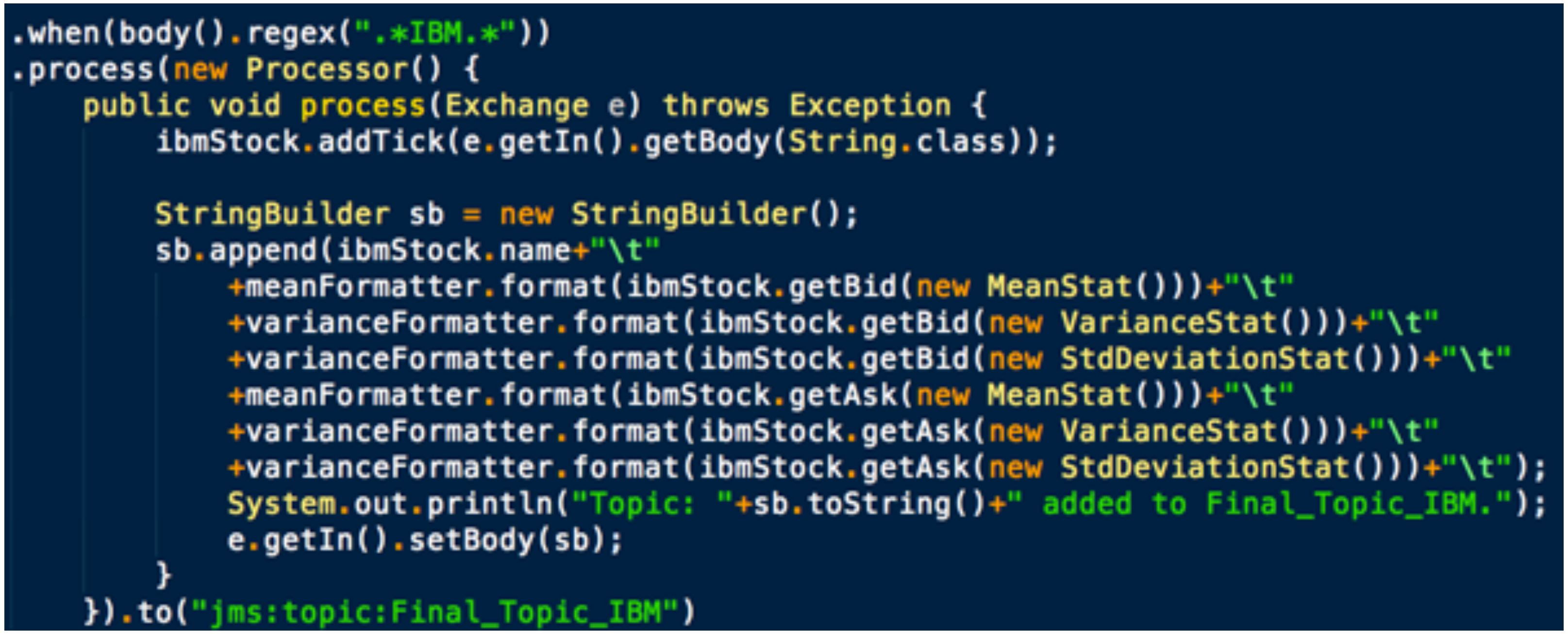
1. Retrieve message from Final_Raw_Data

2. Call corresponding strategies to calculate 6 statistic indices

calculation details encapsulated in Strategy

3. Form new message (eg: "**IBM 123 1.23 0.3124 1.24 0.4**")

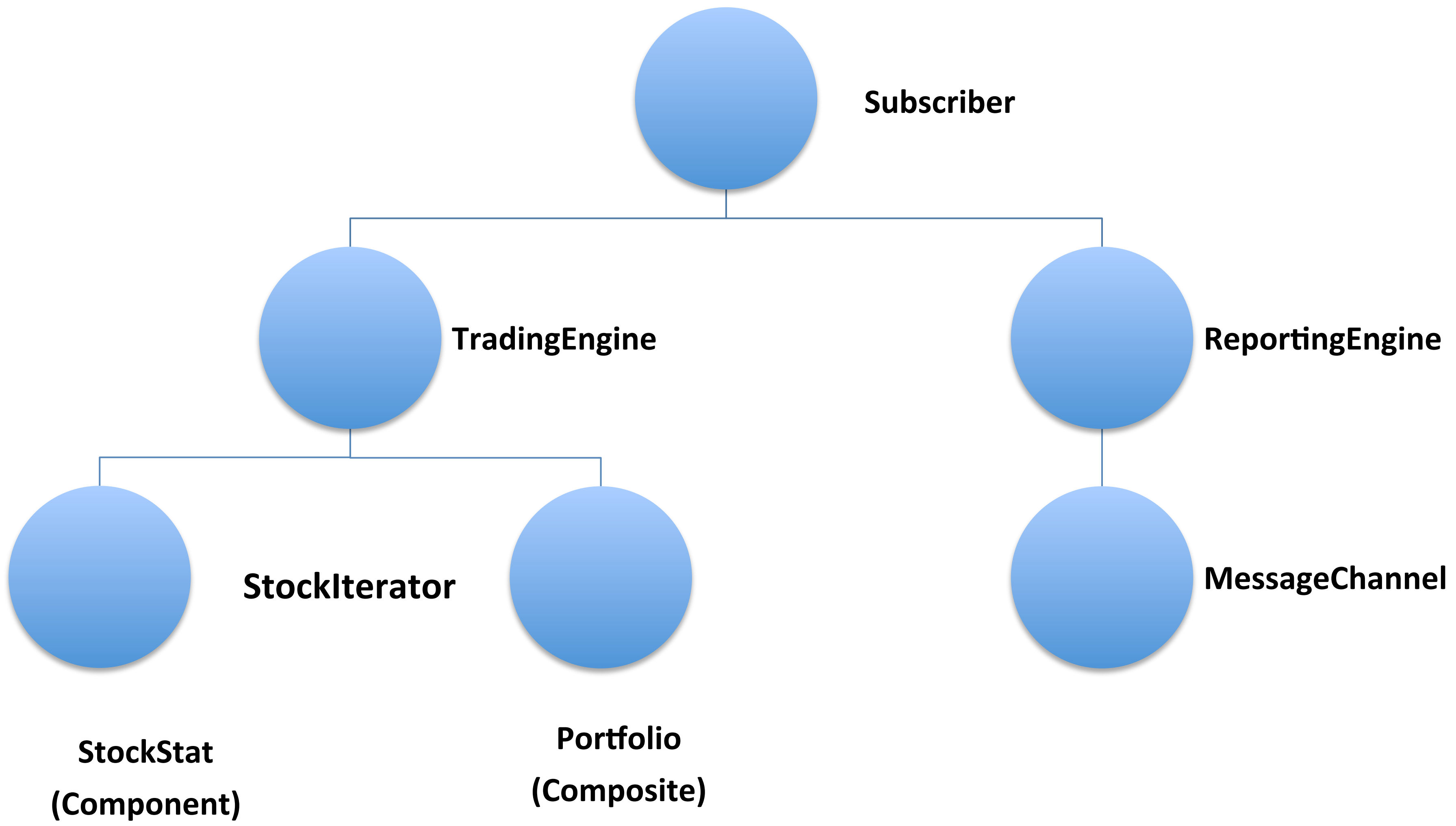
4. Add new messages to corresponding topic (eg: **Final_Topic_ORCL**)



```
.when(body().regex(".*IBM.*"))
.process(new Processor() {
    public void process(Exchange e) throws Exception {
        ibmStock.addTick(e.getIn().getBody(String.class));

        StringBuilder sb = new StringBuilder();
        sb.append(ibmStock.name+"\t"
            +meanFormatter.format(ibmStock.getBid(new MeanStat()))+"\t"
            +varianceFormatter.format(ibmStock.getBid(new VarianceStat()))+"\t"
            +varianceFormatter.format(ibmStock.getBid(new StdDeviationStat()))+"\t"
            +meanFormatter.format(ibmStock.getAsk(new MeanStat()))+"\t"
            +varianceFormatter.format(ibmStock.getAsk(new VarianceStat()))+"\t"
            +varianceFormatter.format(ibmStock.getAsk(new StdDeviationStat()))+"\t");
            System.out.println("Topic: "+sb.toString()+" added to Final_Topic_IBM.");
            e.getIn().setBody(sb);
        }
    }).to("jms:topic:Final_Topic_IBM")
```

3. Subscriber - Overview



3. Subscriber – Component & Composite

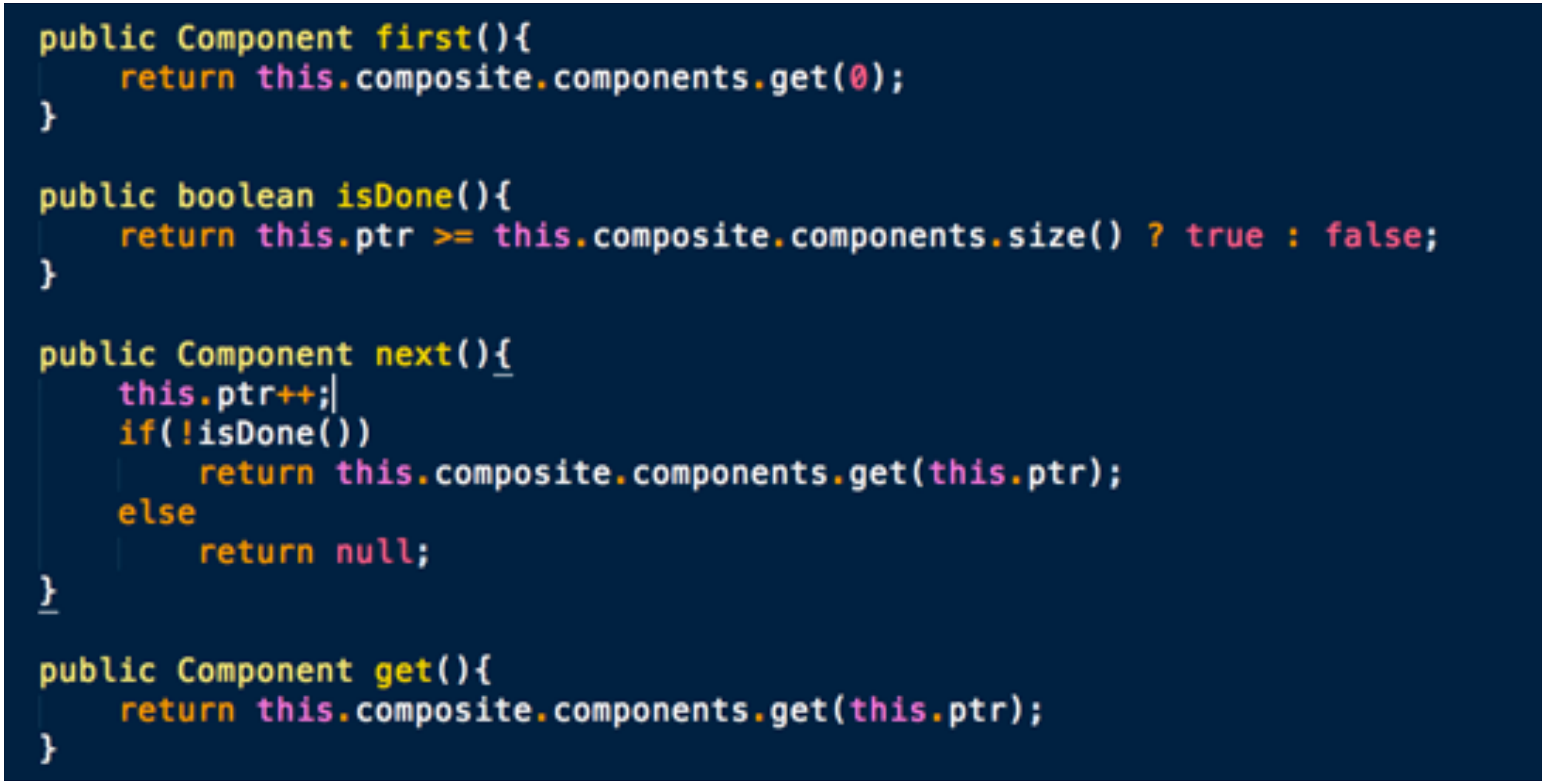
Component

- **StockStat**
  - add(), remove(), **getName()**
  - **update()** set value of stat index (eg: bidMean) to new value (Tick)
  - **reportStat()** form report message (eg: "MSFT-bidMean: 39.807")

Composite

- **Portfolio**
  - **add(), remove(), getName()**
  - **update()** iterate through an array of StockStat and update them
  - reportStat() iterate through an array of StockStat, and recursively form report message, finally return the overall return message

3. Subscriber - StockIterator



```
public Component first(){
    return this.composite.components.get(0);
}

public boolean isDone(){
    return this.ptr >= this.composite.components.size() ? true : false;
}

public Component next(){
    this.ptr++;
    if(!isDone())
        return this.composite.components.get(this.ptr);
    else
        return null;
}

public Component get(){
    return this.composite.components.get(this.ptr);
}
```

Referred to sample code

3. Subscriber - Engines

TradingEngine

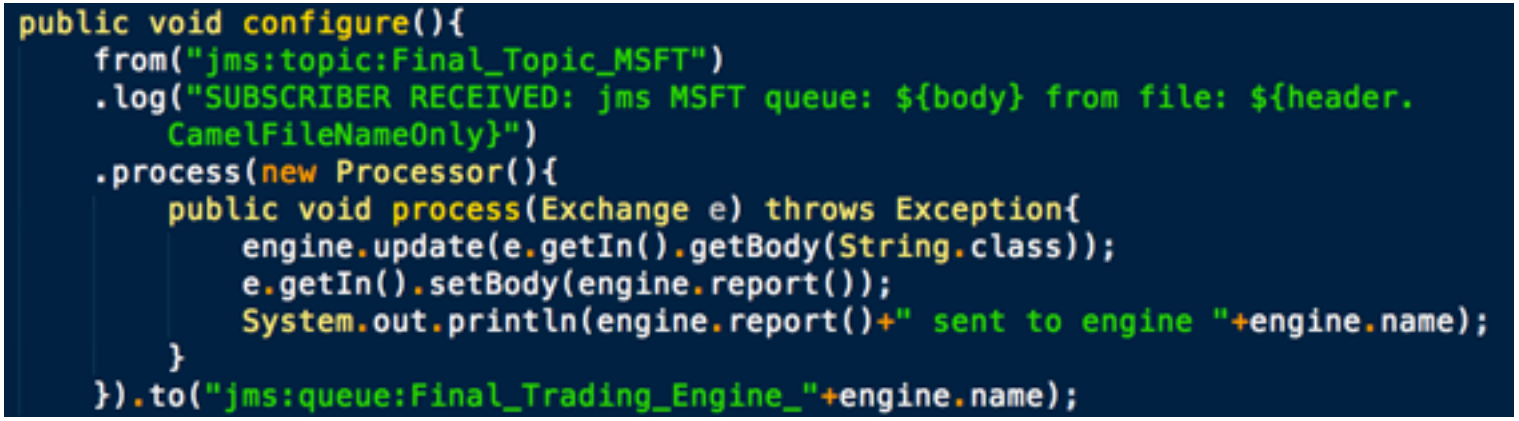
- **Portfolio**
  - Stores portfolios
- **ReportEngine**
  - Has an ReportEngine that generates report message
- **update()**
  - Iterate through all the portfolios and update them
- **report()**
  - Create an instance of ReportEngine and call its report() method

ReportEngine

- **Singleton**
- **report()**
  - Generate the full message (ready to be posted to queues)

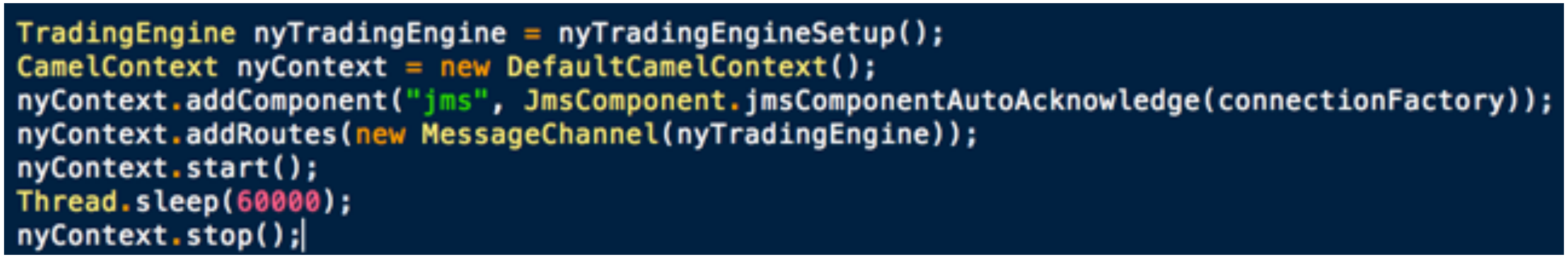
3. Subscriber – MessageChannel & Subscriber

- MessageChannel Configures the endpoint



```java
public void configure(){
    from("jms:topic:Final_Topic_MSFT")
    .log("SUBSCRIBER RECEIVED: jms MSFT queue: ${body} from file: ${header.
        CamelFileNameOnly}")
    .process(new Processor(){
        public void process(Exchange e) throws Exception{
            engine.update(e.getIn().getBody(String.class));
            e.getIn().setBody(engine.report());
            System.out.println(engine.report()+" sent to engine "+engine.name);
        }
    }).to("jms:queue:Final_Trading_Engine_"+engine.name);
```

- Subscriber connect to **ActiveMQ** JMS, setup NewYork, London, Tokyo trading engines
- Create 3 CamelContext and connect to ActiveMQ JMS **broker** listening localhost
- Start the route and let it do its work
- Stop the CamelContext



```java
TradingEngine nyTradingEngine = nyTradingEngineSetup();
CamelContext nyContext = new DefaultCamelContext();
nyContext.addComponent("jms", JmsComponent.jmsComponentAutoAcknowledge(connectionFactory));
nyContext.addRoutes(new MessageChannel(nyTradingEngine));
nyContext.start();
Thread.sleep(60000);
nyContext.stop();
```