Avaliação 1 (Bot Trader)

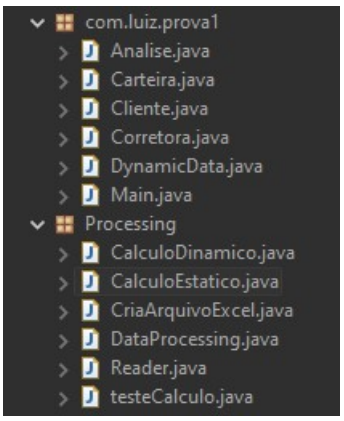
GAT 108 - Automação Avançada
Professor: Arthur de Miranda Neto

Aluno: Luiz Fernando Rodrigues - 201610292 - 22A

Lavras-MG
Agosto de 2022

1 Introdução

O projeto foi organizado de acordo com a imagem abaixo, e a primeira parte da avaliação, que se trata da extração e tratamento dos dados, foi trabalhada no pacote Processing.



2 Extração e Tratamento dos Dados

2.1 Leitura do Arquivo CSV

Inicialmente foi feita a coleta dos dados de 4 ativos de FOREX pelo MetaTrader 5 com um time frame de 10 minutos. Os ativos são : AUDUSD, EURUSD, NZDUSD, USDCAD. Foram selecionado dados de 20/05/2022 à 19/08/22, totalizando em um indíce de em 9505. Após fazer o download dos dados dos 4 ativos, foi feita a leitura do arquivo CSV no JAVA pela classe Reader e DataProcessing. Como pode ser visto a seguir:

```
1  package Processing;
2
3  /**
4  *   Classe responsavel pela leitura dos arquivos csv
5  *   retorna ArrayList de cada coluna
6  *
7  *@author Luiz Fernando
8  */
9
10 import java.io.FileReader;
11 import java.util.ArrayList;
12 import java.util.Arrays;
13 import java.util.List;
14
15 import com.opencsv.CSVParser;
16 import com.opencsv.CSVParserBuilder;
17 import com.opencsv.CSVReader;
18 import com.opencsv.CSVReaderBuilder;
19
```

```
20 public class Reader {
21     private String name;
22     private ArrayList<String> data;
23     private ArrayList<String> time;
24     private ArrayList<Double> open;
25     private ArrayList<Double> high;
26     private ArrayList<Double> low;
27     private ArrayList<Double> close;
28     private ArrayList<Double> tick;
29 
30     public Reader(String name, String path) {
31         this.name = name;
32         data = new ArrayList<String>();
33         time = new ArrayList<String>();
34         open = new ArrayList<Double>();
35         high = new ArrayList<Double>();
36         low = new ArrayList<Double>();
37         close = new ArrayList<Double>();
38         tick = new ArrayList<Double>();
39 
40         try {
41             FileReader filereader = new FileReader(path);
42             CSVParser parser = new CSVParserBuilder().withSeparator(';
                   ').build();
43             CSVReader csvReader = new CSVReaderBuilder(filereader).
                   withCSVParser(parser).build();
44 
45             // Read all data at once
46             List<List<String>> rows = new ArrayList<List<String>>();
47             String[] column = null;
48             csvReader.readNext();
49 
50             while ((column = csvReader.readNext()) != null) {
51                 rows.add(Arrays.asList(column));
52             }
53 
54             rows.forEach(cols -> {
55                 data.add(cols.get(0));
56                 time.add(cols.get(1));
57                 open.add(Double.parseDouble(cols.get(2)));
58                 high.add(Double.parseDouble(cols.get(3)));
59                 low.add(Double.parseDouble(cols.get(4)));
60                 close.add(Double.parseDouble(cols.get(5)));
```

```
                tick.add(Double.parseDouble(cols.get(6)));
            });

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public ArrayList<String> getData() {
        return data;
    }

    public ArrayList<String> getTime() {
        return time;
    }

    public ArrayList<Double> getOpen() {
        return open;
    }

    public ArrayList<Double> getHigh() {
        return high;
    }

    public ArrayList<Double> getLow() {
        return low;
    }

    public ArrayList<Double> getClose() {
        return close;
    }

    public ArrayList<Double> getTick() {
        return tick;
    }
```

```
104
105 }
```

A partir dessa classe foi possível criar a classe DataProcessing para receber os dados lidos de todos ativos.

```
1
2  package Processing;
3  /**
4  *   Classe responsavel por instanciar um ArrayList do tipo
5  *   Reader para os 4 ativos para retornar um getAssets
6  *
7  *@author Luiz Fernando
8  */
9  import java.util.ArrayList;
10
11 public class DataProcessing {
12
13     private ArrayList<Reader> assets;
14     private Reader Asset_1;
15     private Reader Asset_2;
16     private Reader Asset_3;
17     private Reader Asset_4;
18
19     public DataProcessing() {
20
21         assets = new ArrayList<Reader>();
22
23         String path_1 = "C:\\Projetos_java\\prova1\\Dados\\AUDUSD_M10.
               csv";
24         String path_2 = "C:\\Projetos_java\\prova1\\Dados\\EURUSD_M10.
               csv";
25         String path_3 = "C:\\Projetos_java\\prova1\\Dados\\NZDUSD_M10.
               csv";
26         String path_4 = "C:\\Projetos_java\\prova1\\Dados\\USDCAD_M10.
               csv";
27
28         Asset_1 = new Reader("Asset1", path_1);
29         Asset_2 = new Reader("Asset2", path_2);
30         Asset_3 = new Reader("Asset3", path_3);
31         Asset_4 = new Reader("Asset4", path_4);
32
33         assets.add(Asset_1);
```

```
34          assets.add(Asset_2);
35          assets.add(Asset_3);
36          assets.add(Asset_4);
37
38      }
39
40      public ArrayList<Reader> getAssets() {
41          return assets;
42      }
43
44  }
```

O metodo getAssets retorna a lista com todos os dados de todos ativos. A partir disso podemos instanciar um atributo do tipo DataProcessing nas outras classes para receber os dados e fazer as operações necessarias.

2.2 Calculos

Como criterio de análise das tendencias, os traders utilizam de parametros estatísticos para tomadas de decisão. Para isso utilizaremos os valores de fechamento recebido no arquivo lido para calcular a média movel de curta, media e longa duração, calcular a media movel exponencia e o desvio padrão.

Nesta sessão trataremos do conjunto completo dos dados, ou seja calcularemos esses parametros enviando todos os dados de uma vez.

```
1   package Processing;
2
3   import java.util.ArrayList;
4
5   public class CalculoEstatico {
6
7       private DataProcessing Process;
8       protected ArrayList<Reader> assets;
9
10      private double[] mediaAritmetica;
11      private double[] mediaExponencial;
12      private double[] desvioPadrao;
13      private int n;
14
15      public CalculoEstatico() {
16          Process = new DataProcessing();
17          assets = Process.getAssets();
18      }
```

```
19
20      public double[] mediaMovel(double intervalo, int nAsset) {
21          n = assets.get(nAsset).getClose().size();
22          mediaAritmetica = new double[n];
23
24          for (int i = 0; i < n; i++) {
25              if (i >= intervalo) {
26                  double sum = 0;
27                  for (int j = i; j > i - intervalo; j--) {
28                      sum += assets.get(nAsset).getClose().get(j - 1);
29                  }
30                  mediaAritmetica[i - 1] = (sum / intervalo);
31              }
32          }
33          return mediaAritmetica;
34      }
35
36      public double[] mediaExpo(double intervalo, int nAsset) {
37          double mult = 2 / (1 + intervalo);
38          int cont = (int) intervalo;
39          double close = 0;
40          double[] Aux = mediaMovel(intervalo, nAsset);
41
42          n = assets.get(nAsset).getClose().size();
43          mediaExponencial = new double[n];
44          mediaExponencial[cont - 1] = Aux[cont - 1];
45
46          for (int j = cont; j < n; j++) {
47
48              close = assets.get(nAsset).getClose().get(j);
49              mediaExponencial[j] = (mult * (close - Aux[cont - 1])) +
                    Aux[cont - 1];
50              Aux[cont - 1] = mediaExponencial[j];
51
52          }
53          return mediaExponencial;
54      }
55
56      public double[] desvioPadrao(double intervalo, int nAsset) {
57          n = assets.get(nAsset).getClose().size();
58          double somediaAritmetica = 0;
59          int aux = 0;
60
```

```
61          desvioPadrao = new double[n];
62          mediaAritmetica = mediaMovel(intervalo, nAsset);
63
64          for (int i = (int) intervalo; i < mediaAritmetica.length; i++)
                {
65              somediaAritmetica = 0;
66              for (int j = aux; j < i; j++) {
67                  somediaAritmetica += (double) Math.pow(assets.get(
                        nAsset).getClose().get(j) - mediaAritmetica[i], 2);
68              }
69
70              desvioPadrao[i] = Math.sqrt(somediaAritmetica / intervalo)
                    ;
71              aux++;
72          }
73          return desvioPadrao;
74      }
75
76      public double[] MMcurta(int nAsset) {
77          return mediaMovel(5, nAsset);
78      }
79
80      public double[] MMinter(int nAsset) {
81          return mediaMovel(10, nAsset);
82      }
83
84      public double[] MMlonga(int nAsset) {
85          return mediaMovel(20, nAsset);
86      }
87
88  }
```

Dentro do pacote Processing foi criado uma classe main chamada TesteCalculo para receber os calculos e chamar outra classe CriaArquivoExcel. Dessa forma os dados da primeira parte da prova foram salvos.

```
1  package Processing;
2
3  import java.util.ArrayList;
4
5  public class testeCalculo {
6
7      public static void main(String[] args) {
```

```
8
9           CalculoEstatico c = new CalculoEstatico();
10
11          // Media Movel exponencial
12          ArrayList<double[]> MME = new ArrayList<>();
13
14          // Media Movel Simples
15          ArrayList<double[]> MMcurta = new ArrayList<>();
16          ArrayList<double[]> MMinter = new ArrayList<>();
17          ArrayList<double[]> MMlonga = new ArrayList<>();
18          ArrayList<double[]> DP = new ArrayList<>();
19
20          // Estrutura q envia pre o de fechamento de todos ativos
21          ArrayList<ArrayList<Double>> assets = new ArrayList<ArrayList<
                Double>>();
22
23          for (int i = 0; i < 4; i++) {
24              assets.add(c.assets.get(i).getClose());
25              MME.add(c.mediaExpo(5, i));
26              MMcurta.add(c.MMcurta(i));
27              MMinter.add(c.MMinter(i));
28              MMlonga.add(c.MMlonga(i));
29              DP.add(c.desvioPadrao(5, i));
30          }
31
32          ArrayList<double[]> calculo0 = new ArrayList<>();
33          calculo0.add(MMcurta.get(0));
34          calculo0.add(MMinter.get(0));
35          calculo0.add(MMlonga.get(0));
36          calculo0.add(MME.get(0));
37          calculo0.add(DP.get(0));
38          CriaArquivoExcel.criarArquivo("Ativo0.xlsx", assets.get(0),
                calculo0);
39
40          ArrayList<double[]> calculo1 = new ArrayList<>();
41          calculo1.add(MMcurta.get(1));
42          calculo1.add(MMinter.get(1));
43          calculo1.add(MMlonga.get(1));
44          calculo1.add(MME.get(1));
45          calculo1.add(DP.get(1));
46          CriaArquivoExcel.criarArquivo("Ativo1.xlsx", assets.get(1),
                calculo1);
47
```

```
48          ArrayList<double[]> calculo2 = new ArrayList<>();
49          calculo2.add(MMcurta.get(2));
50          calculo2.add(MMinter.get(2));
51          calculo2.add(MMlonga.get(2));
52          calculo2.add(MME.get(2));
53          calculo2.add(DP.get(2));
54          CriaArquivoExcel.criarArquivo("Ativo2.xlsx", assets.get(2),
                calculo2);
55
56          ArrayList<double[]> calculo3 = new ArrayList<>();
57          calculo3.add(MMcurta.get(3));
58          calculo3.add(MMinter.get(3));
59          calculo3.add(MMlonga.get(3));
60          calculo3.add(MME.get(3));
61          calculo3.add(DP.get(3));
62          CriaArquivoExcel.criarArquivo("Ativo3.xlsx", assets.get(3),
                calculo3);
63
64      }
65
66  }
```

Como pode ser visto a classe testeCalculo chama CriaArquivoExcel.criarArquivo(); passando os paramtros a serem salvos. A seguir esta explícito o codigo fonte da classe CriaArquivoExcel.

```
1  package Processing;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.util.ArrayList;
7
8  import org.apache.poi.ss.usermodel.Cell;
9  import org.apache.poi.ss.usermodel.Row;
10 import org.apache.poi.xssf.usermodel.XSSFSheet;
11 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
12
13 import lombok.extern.slf4j.Slf4j;
14
15 @Slf4j
16 public class CriaArquivoExcel {
17
```

```
18      public static void criarArquivo(String nomeArquivo, ArrayList<
            Double> close, ArrayList<double[]> calculo) {
19
20          System.out.println("Gerando o arquivo {}" + nomeArquivo);
21
22          try (var workbook = new XSSFWorkbook(); var outputStream = new
                FileOutputStream(nomeArquivo)) {
23              var planilha = workbook.createSheet("Calculos");
24              int numeroDaLinha = 0;
25
26              adicionarCabecalho(planilha, numeroDaLinha++);
27
28              for (int i = 0; i < close.size(); i++) {
29
30                  var linha = planilha.createRow(numeroDaLinha++);
31                  adicionarCelula(linha, 0, close.get(i));
32                  adicionarCelula(linha, 1, calculo.get(0)[i]);
33                  adicionarCelula(linha, 2, calculo.get(1)[i]);
34                  adicionarCelula(linha, 3, calculo.get(2)[i]);
35                  adicionarCelula(linha, 4, calculo.get(3)[i]);
36                  adicionarCelula(linha, 5, calculo.get(4)[i]);
37              }
38
39              workbook.write(outputStream);
40          } catch (FileNotFoundException e) {
41              System.out.println("Arquivo n o encontrado: {}" +
                    nomeArquivo);
42          } catch (IOException e) {
43              System.out.println("Erro ao processar o arquivo: {} " +
                    nomeArquivo);
44          }
45          System.out.println("Arquivo gerado com sucesso!");
46      }
47
48      private static void adicionarCabecalho(XSSFSheet planilha, int
            numeroLinha) {
49          var linha = planilha.createRow(numeroLinha);
50          adicionarCelula(linha, 0, "Close");
51          adicionarCelula(linha, 1, "MM curta");
52          adicionarCelula(linha, 2, "MM inter");
53          adicionarCelula(linha, 3, "MM longa");
54          adicionarCelula(linha, 4, "MME");
55          adicionarCelula(linha, 5, "DP");
```

```
56      }
57
58      private static void adicionarCelula(Row linha, int coluna, Double
            close) {
59          Cell cell = linha.createCell(coluna);
60          cell.setCellValue(close);
61      }
62
63      private static void adicionarCelula(Row linha, int coluna, double
            valor) {
64          Cell cell = linha.createCell(coluna);
65          cell.setCellValue(valor);
66      }
67
68      private static void adicionarCelula(Row linha, int coluna, String
            valor) {
69          Cell cell = linha.createCell(coluna);
70          cell.setCellValue(valor);
71      }
72  }
```

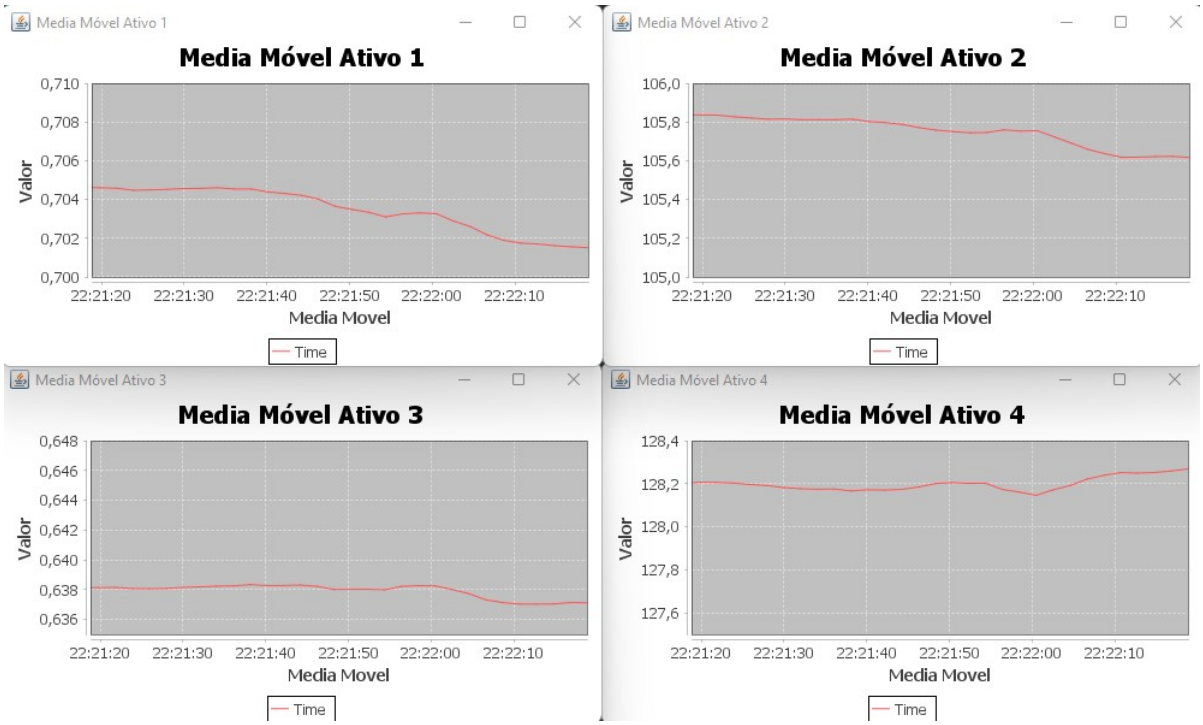
Dessa forma foi criado 4 arquivos com extensão xlsx, cada um com seus respectivos dados dos ativos. Como pode ser visto a seguir.

	A	B	C	D	E	F
1	Close	MM curta	MM inter	MM longa	MME	DP
2	0,70435	0	0	0	0	0
3	0,70456	0	0	0	0	0
4	0,70483	0	0	0	0	0
5	0,70444	0	0	0	0	0
6	0,7045	0,704536	0	0	0,704536	0
7	0,70467	0,7046	0	0	0,704581	0,000175
8	0,70491	0,70467	0	0	0,70469	0,000155
9	0,70448	0,7046	0	0	0,70462	0,000195
10	0,70454	0,70462	0	0	0,704594	0,000175
11	0,70443	0,704606	0,704571	0	0,704539	0,00016
12	0,70453	0,704578	0,704589	0	0,704536	0,000174
13	0,70439	0,704474	0,704572	0	0,704487	0,0002
14	0,70457	0,704492	0,704546	0	0,704515	6,02E-05
15	0,70469	0,704522	0,704571	0	0,704573	7,56E-05
16	0,70464	0,704564	0,704585	0	0,704596	0,000114
17	0,70457	0,704572	0,704575	0	0,704587	0,000103
18	0,70457	0,704608	0,704541	0	0,704581	0,000108
19	0,70423	0,70454	0,704516	0	0,704464	8,39E-05
20	0,70469	0,70454	0,704531	0	0,704539	0,000161
21	0,70388	0,704388	0,704476	0,704524	0,70432	0,000222
22	0,70418	0,70431	0,704441	0,704515	0,704273	0,000307
23	0,70409	0,704214	0,704411	0,704492	0,704212	0,000305
24	0,70324	0,704016	0,704278	0,704412	0,703888	0,000332
25	0,70283	0,703644	0,704092	0,704332	0,703535	0,0006
26	0,70308	0,703484	0,703936	0,704261	0,703384	0,000547
27	0,70345	0,703338	0,703824	0,7042	0,703406	0,000567

Calculos

2.3 Gráficos

A partir dos calculos das medias moveis, foi gerado um grafico para cada ativo, em tempo de execução.



3 Bot Trader

Nesta sessão trataremos do desenvolvimento do sistema como um todo, atendendo todos os requisitos de funcionamento imposto no arquivo base da avaliação. Inicialmente foi dedenvolvida a classe corretora (Thread), que tem por objetivo principal publicar os valores dos ativo periodicamente e registrar as operações feita pelos clientes. Que por sua vez também é uma classe que herda uma Thread, e tem por objetivo instanciar um objeto do tipo analise para cada ativo publicado pela corretora. A classe cliente tambem implementa metodos para tomada de decisão de acordo com as tendências da classe analise. A classe analise tambem é uma Thread, que aplica os calculos de médias moveis como critérios para tendências de compra, venda e dradown. O controle do saldo e da quantidade de ativo de cada cliente foi implementado na classe Carteira.

3.1 Main

A classe main apenas instancia uma corretora, e dez clientes par a mesma corretora.

```
1 package com.luiz.prova1;
2
3 public class Main {
4
5     public static void main(String[] args) {
```

```
6
7           Corretora corretora = new Corretora();
8           corretora.start();
9
10          Cliente c1 = new Cliente("Luiz", corretora, 1000.00);
11          c1.start();
12          Cliente c2 = new Cliente("Pedro", corretora, 1000.00);
13          c2.start();
14          Cliente c3 = new Cliente("Algusto", corretora,1000.00);
15          c3.start();
16          Cliente c4 = new Cliente("Amanda", corretora, 1000.00);
17          c4.start();
18          Cliente c5 = new Cliente("Julia", corretora, 1000.00);
19          c5.start();
20          Cliente c6 = new Cliente("Joao", corretora, 1000.00);
21          c6.start();
22          Cliente c7 = new Cliente("Fabio", corretora, 1000.00);
23          c7.start();
24          Cliente c8 = new Cliente("Rafael", corretora, 1000.00);
25          c8.start();
26          Cliente c9 = new Cliente("Julio", corretora, 1000.00);
27          c9.start();
28          Cliente c10 = new Cliente("Henrique", corretora, 1000.00);
29          c10.start();
30      }
31
32  }
```

3.2 Corretora

```
1  package com.luiz.prova1;
2
3  import java.io.FileNotFoundException;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.util.ArrayList;
7  import java.util.Hashtable;
8  import java.util.concurrent.Semaphore;
9  import java.util.concurrent.atomic.AtomicInteger;
10
11 import org.apache.poi.ss.usermodel.Cell;
```

```java
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.jfree.ui.RefineryUtilities;

//import java.util.concurrent.Semaphore;
//import java.util.concurrent.atomic.AtomicInteger;
import Processing.DataProcessing;
import Processing.Reader;

public class Corretora extends Thread {

    // leitura dos ativos
    private DataProcessing Process;
    private static ArrayList<Reader> Assets = new ArrayList<>();

    // parte divulgada pela corretora
    public static ArrayList<Double> Asset1 = new ArrayList<>();
    public static ArrayList<Double> Asset2 = new ArrayList<>();
    public static ArrayList<Double> Asset3 = new ArrayList<>();
    public static ArrayList<Double> Asset4 = new ArrayList<>();

    // Lista de 4 ativos que serao divulgados publicamente pela
       corretora
    public static ArrayList<ArrayList<Double>> publicPRICE = new
       ArrayList<ArrayList<Double>>();

    public static int amostra; // amostra em que percorre o vetor de
       ativos
    private static AtomicInteger cont; // numero de opera  es
    private Semaphore semaforo = new Semaphore(2);// semaforo para os
       caixas 1 e 2

    //Atributo para o grafico
    final DynamicData chart1;
    final DynamicData chart2;
    final DynamicData chart3;
    final DynamicData chart4;

    //Atributo para salvar os dados em excel
    private static ArrayList<Hashtable<String, String>> caixaGeral;

    public Corretora() {
```

```
51
52          // Faz leitura dos dados e retorna uma lista de ativos (Assets
                )
53          Process = new DataProcessing();
54          Assets = Process.getAssets();
55          amostra = 0;
56          cont = new AtomicInteger(1);
57
58          //Instancia dos graficos
59          chart1 = new DynamicData("Media M vel Ativo 1", 0.70, 0.71);
60          chart1.pack();
61          RefineryUtilities.centerFrameOnScreen(chart1);
62          chart1.setVisible(true);
63
64          chart2 = new DynamicData("Media M vel Ativo 2", 105, 106);
65          chart2.pack();
66          RefineryUtilities.centerFrameOnScreen(chart2);
67          chart2.setVisible(true);
68
69          chart3 = new DynamicData("Media M vel Ativo 3", 0.635, 0.648)
                ;
70          chart3.pack();
71          RefineryUtilities.centerFrameOnScreen(chart3);
72          chart3.setVisible(true);
73
74          chart4 = new DynamicData("Media M vel Ativo 4", 127.5, 128.4)
                ;
75          chart4.pack();
76          RefineryUtilities.centerFrameOnScreen(chart4);
77          chart4.setVisible(true);
78
79          caixaGeral = new ArrayList<Hashtable<String, String>>();
80      }
81
82      //Metodo para la o de repeticao dentro da excecucao das Threads
83      public static boolean adicionando() {
84          if ((Asset1.size() != Assets.get(0).getClose().size())) {
85              return true;
86          }
87          return false;
88      }
89
90      private void addPrice() {
```

```
91          Asset1.add(Assets.get(0).getClose().get(amostra));
92          Asset2.add(Assets.get(1).getClose().get(amostra));
93          Asset3.add(Assets.get(2).getClose().get(amostra));
94          Asset4.add(Assets.get(3).getClose().get(amostra));
95
96          publicPRICE.add(0, Asset1);
97          publicPRICE.add(1, Asset2);
98          publicPRICE.add(2, Asset3);
99          publicPRICE.add(3, Asset4);
100
101         chart1.incrementValue(0);
102         chart2.incrementValue(1);
103         chart3.incrementValue(2);
104         chart4.incrementValue(3);
105     }
106
107     @Override
108     public void run() {
109         while (adicionando()) {
110             try {
111                 addPrice();
112                 criarArquivo();
113                 amostra++;
114                 Thread.sleep(2000);
115             } catch (InterruptedException e) {
116                 e.printStackTrace();
117             }
118         }
119         System.out.println("fim");
120
121     }
122
123     public boolean registrarOperacao(String operacao, String cliente,
        int ativo, double valorAtual, int posicao,
124              double saldo) {
125         try {
126             if (cont.get() == 1001) {
127                 System.out.println("NUMERO DE OPERACOES ESGOTADO");
128                 return false;
129             } else {
130                 // Semaforo para CAIXA 1 e CAIXA 2; permitido somente
                       dois ciientes
131                 semaforo.acquire();
```

```
132                 // System.out.println("-- ENTROU CLIENTE: " + cliente
                        + "--");
133
134                 String data = Assets.get(ativo).getData().get(posicao)
                        ;
135                 String hora = Assets.get(ativo).getTime().get(posicao)
                        ;
136
137                 //--------Estrutura para salvar os parametros no excel
                        ----------
138                 Hashtable<String, String> registro = new Hashtable<
                        String, String>();
139
140                 registro.put("Cliente", String.valueOf(cliente));
141                 registro.put("Data", data);
142                 registro.put("Hora", hora);
143                 registro.put("Operacao", operacao);
144                 registro.put("Ativo", String.valueOf(ativo));
145                 registro.put("Valor", String.valueOf(valorAtual));
146                 registro.put("Saldo", String.valueOf(saldo));
147
148                 caixaGeral.add(registro); //ArrayList com os dados
                        ser salvo
149
150                 System.out.println("Op: " + cont + " -> Cliente: " +
                        cliente + ", efetuou " + operacao
151                         + " do ativo: " + ativo + " Saldo Atual: " +
                            saldo);
152                 cont.incrementAndGet();
153                 // System.out.println("-- SAIU CLIENTE: " + cliente +
                        "--");
154                 return true;
155             }
156
157         } catch (InterruptedException e) {
158             e.printStackTrace();
159         } finally {
160             semaforo.release();
161         }
162         return false;
163     }
164     //Metodo para criar o arquivo e salvar os dados
165     public synchronized static void criarArquivo() {
```

```
166
167         try (var workbook = new XSSFWorkbook(); var outputStream = new
                FileOutputStream("RegistroCaixaGeral.xlsx")) {
168             var planilha = workbook.createSheet("CaixaGeral");
169             int numeroDaLinha = 0;
170
171             adicionarCabecalho(planilha, numeroDaLinha++);
172
173             for (int i = 0; i < caixaGeral.size(); i++) {
174                 var linha = planilha.createRow(numeroDaLinha++);
175                 adicionarCelula(linha, 0, caixaGeral.get(i).get("
                       Cliente"));
176                 adicionarCelula(linha, 1, caixaGeral.get(i).get("Data"
                       ));
177                 adicionarCelula(linha, 2, caixaGeral.get(i).get("Hora"
                       ));
178                 adicionarCelula(linha, 3, caixaGeral.get(i).get("
                       Operacao"));
179                 adicionarCelula(linha, 4, caixaGeral.get(i).get("Ativo
                       "));
180                 adicionarCelula(linha, 5, caixaGeral.get(i).get("Valor
                       "));
181                 adicionarCelula(linha, 6, caixaGeral.get(i).get("Saldo
                       "));
182             }
183             workbook.write(outputStream);
184         } catch (FileNotFoundException e) {
185             System.out.println("Arquivo nao encontrado:
                    RegistroCaixaGeral");
186         } catch (IOException e) {
187             System.out.println("Erro ao processar o arquivo:
                    RegistroCaixaGeral ");
188         }
189         System.out.println("Arquivo gerado com sucesso!");
190     }
191
192     private static void adicionarCabecalho(XSSFSheet planilha, int
           numeroLinha) {
193         var linha = planilha.createRow(numeroLinha);
194         adicionarCelula(linha, 0, "Cliente");
195         adicionarCelula(linha, 1, "Data");
196         adicionarCelula(linha, 2, "Hora");
197         adicionarCelula(linha, 3, "Opera  o");
```

```
198             adicionarCelula(linha, 4, "Ativo");
199             adicionarCelula(linha, 5, "Valor");
200             adicionarCelula(linha, 6, "Saldo atual");
201         }
202 
203         private static void adicionarCelula(Row linha, int coluna, String
                valor) {
204             Cell cell = linha.createCell(coluna);
205             cell.setCellValue(valor);
206         }
207 
208 }
```

3.3 Cliente

```
1  package com.luiz.prova1;
2  
3  public class Cliente extends Thread {
4  
5      // Array para analise de cada ativo
6      private Analise[] analises = new Analise[4];
7  
8      // Tendencia -> true (comprado) false (vendido)
9      private boolean[] trend = new boolean[4];
10 
11     // Criterios de stoploss
12     private boolean[] draw = new boolean[4];
13 
14     // Dados Carteira
15     private Corretora corretora;
16     private Carteira carteira;
17 
18     private int lastAsset;
19     private boolean lastTipeOp;
20 
21     public Cliente(String name, Corretora corretora, double saldo) {
22         super(name);
23         this.corretora = corretora;
24         carteira = new Carteira(saldo);
25         lastAsset = 10;
26 
```

```
27          // Analise de ativos individuais
28          for (int i = 0; i < analises.length; i++) {
29              analises[i] = new Analise(Corretora.publicPRICE.get(i));
30              analises[i].start();
31          }
32
33      }
34
35      public void run() {
36          System.out.println("Cliente " + this.getName() + " criado com
               saldo de : R$ " + carteira.getSaldo());
37
38          while (Corretora.adicionando()) {
39              try {
40                  for (int i = 0; i < 4; i++) {
41                      trend[i] = analises[i].tendenciaGeral(); // 0 se
                           compra,1 se vende, 2 se for drawdowm
42                      draw[i] = analises[i].drawDown();
43                      verifica(i, trend[i], draw[i]);
44                  }
45                  Thread.sleep(2000);
46              } catch (InterruptedException e) {
47                  e.printStackTrace();
48              }
49          }
50      }
51
52      public void verifica(int Ativo, boolean tendencia, boolean
           drawdown) {
53          // Verifica se as duas opera  es sao iguais e se a
               opera  o    de compra ou venda
54          if ((lastAsset != Ativo) || (lastTipeOp != tendencia)) {
55
56              lastTipeOp = tendencia;
57              lastAsset = Ativo;
58
59              // Verifica se    comprado, vendido ou drawdown
60              if (tendencia && (drawdown == false)) {
61                  System.out.println("Tendecia de ALTA p/ ativo " +
                       Ativo + " do cliente " + this.getName());
62                  comprar(Ativo, tendencia);
63              } else if (!tendencia && (drawdown == false)) {
```

```
64                  System.out.println("Tendecia de BAIXA p/ ativo " +
                        Ativo + " do cliente " + this.getName());
65                  vender(Ativo, 5, tendencia);
66              } else if (drawdown == true) {
67                  System.out.println("DRAWDOWN para o ativo " + Ativo +
                        " do cliente " + this.getName());
68                  vender(Ativo, 10, tendencia);
69              }
70
71          } else {
72              // System.out.println("Nao podem realizar duas operacoes
                    iguais seguidas com o
73              // mesmo ativo!");
74          }
75
76      }
77
78      public void comprar(int ativo, boolean tendencia) {
79          try {
80              double valorAtual = 0;
81              int posicao = Corretora.publicPRICE.get(ativo).size() - 1;
82              valorAtual = Corretora.publicPRICE.get(ativo).get(posicao)
                    ;
83
84              if (carteira.getSaldo() > valorAtual) {
85                  boolean valida = this.corretora.registrarOperacao("
                        Compra", this.getName(), ativo, valorAtual, posicao
                        ,
86                                carteira.getSaldo());
87                  if (valida) {
88                      carteira.registrarCarteira(ativo, valorAtual,
                            tendencia);
89                  }
90                  Thread.sleep(500); // Dps q opera espera 500ms
91              } else {
92                  System.out.println(this.getName() + " nao possui saldo
                        para comprar o ativo " + ativo);
93              }
94          } catch (InterruptedException e) {
95              e.printStackTrace();
96          }
97
98      }
```

```
99
100     public void vender(int ativo, int prioridade, boolean tendencia) {
101         try {
102             if (carteira.temAtivo(ativo)) {
103                 this.setPriority(prioridade);
104                 boolean valida = false;
105                 double valorAtual = 0;
106                 int posicao = Corretora.publicPRICE.get(ativo).size()
                        - 1;
107                 valorAtual = Corretora.publicPRICE.get(ativo).get(
                        posicao);
108
109                 if (prioridade == 10) {
110                     valida = this.corretora.registrarOperacao("
                            DRAWDOWN", this.getName(), ativo, valorAtual,
                            posicao,
111                                 carteira.getSaldo());
112                 } else {
113                     valida = this.corretora.registrarOperacao("Venda",
                             this.getName(), ativo, valorAtual, posicao,
114                                 carteira.getSaldo());
115                 }
116
117                 if (valida) {
118                     carteira.registrarCarteira(ativo, valorAtual,
                            tendencia);
119                 }
120
121                 this.setPriority(5);
122                 Thread.sleep(500); // Dps q opera espera 500ms
123             } else {
124                 // System.out.println(this.getName() + " nao possui o
                        ativo " + ativo + " para
125                 // venda");
126             }
127         } catch (InterruptedException e) {
128             e.printStackTrace();
129         }
130     }
131
132 }
```

3.4 Carteira

```java
package com.luiz.prova1;

public class Carteira {

    private double saldo;
    public int[] qtdAtivo;

    public Carteira(double _saldo) {
        this.saldo = _saldo;
        qtdAtivo = new int[4];
    }

    public synchronized double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public boolean temAtivo(int ativo) {
        if (qtdAtivo[ativo] < 1) {
            return false;
        }
        return true;
    }

    public void saque(double valor) {
        setSaldo(getSaldo() - valor);
    }

    public void deposito(double valor) {
        setSaldo(getSaldo() + valor);
    }

    public synchronized void registrarCarteira(int ativo, double valor
       , boolean tipoOperacao) {

        if (tipoOperacao) {
            saque(valor);
            qtdAtivo[ativo]++;
```

```
41
42          } else if (!tipoOperacao) {
43              deposito(valor);
44              qtdAtivo[ativo]--;
45          }
46      }
47  }
```

3.5 Analise

```
1  package com.luiz.prova1;
2
3  import java.util.ArrayList;
4  import Processing.CalculoDinamico;
5
6  public class Analise extends Thread {
7
8      public static ArrayList<Double> mediaCurta;
9      public static ArrayList<Double> mediaInter;
10     public static ArrayList<Double> mediaLonga;
11
12     // Ativo a ser analisado
13     public ArrayList<Double> AssetList = new ArrayList<>();
14
15     //Variaveis para calculo do Drawdown
16     private Double calculo;
17     private Double max;
18     private Double min;
19
20     // CONSTRUTOR
21     public Analise(ArrayList<Double> Asset) {
22         this.AssetList = Asset;
23         mediaCurta = new ArrayList<Double>();
24         mediaInter = new ArrayList<Double>();
25         mediaLonga = new ArrayList<Double>();
26
27         calculo = 0.0;
28         max = 0.0;
29         min = 0.0;
30     }
31
```

```
32      public void run() {
33
34          while (Corretora.adicionando()) {
35              try {
36                  CalculoDinamico.MMcurta(AssetList, mediaCurta);
37                  CalculoDinamico.MMinter(AssetList, mediaInter);
38                  CalculoDinamico.MMlonga(AssetList, mediaLonga);
39
40                  Thread.sleep(2000);
41              } catch (InterruptedException ex) {
42                  ex.printStackTrace();
43              }
44          }
45      }
46
47      public boolean tendenciaGeral() {
48
49          if (mediaLonga.size() >= 2) {
50              if (mediaCurta.get(mediaCurta.size() - 1) > mediaLonga.get
                    (mediaLonga.size() - 1)) {
51                  return true;
52              }
53          }
54          return false;
55      }
56
57      public boolean drawDown() {
58          if (AssetList.get(AssetList.size() - 1) > max) {
59              max = AssetList.get(AssetList.size() - 1);
60              min = AssetList.get(AssetList.size() - 1);
61          } else if (AssetList.get(AssetList.size() - 1) < min) {
62              min = AssetList.get(AssetList.size() - 1);
63          }
64          calculo = ((max - min) / max) * 100.00;
65          if (calculo > 15.0) { // Se a queda for maior q 15% vende tudo
66              return true;
67          }
68          return false;
69      }
70  }
```

4 Resultados e conclusão

As operações foram registradas no caixa geral respeitando o limite de uso apenas para dois caixas. Os dados salvos em excel pode ser visto a seguir.

```
Tendecia de ALTA p/ ativo 2 do cliente Henrique
Op: 992 -> Cliente: Henrique, efetuou Compra do ativo: 2 Saldo Atual: 525.8031499999998
Tendecia de ALTA p/ ativo 2 do cliente Algusto
Op: 993 -> Cliente: Algusto, efetuou Compra do ativo: 2 Saldo Atual: 527.1292699999999
Tendecia de ALTA p/ ativo 3 do cliente Joao
Tendecia de ALTA p/ ativo 3 do cliente Julio
Op: 995 -> Cliente: Julio, efetuou Compra do ativo: 3 Saldo Atual: 525.2381599999999
Tendecia de ALTA p/ ativo 1 do cliente Rafael
Op: 994 -> Cliente: Joao, efetuou Compra do ativo: 3 Saldo Atual: 522.6763799999998
Op: 996 -> Cliente: Rafael, efetuou Compra do ativo: 1 Saldo Atual: 548.50987
Tendecia de ALTA p/ ativo 1 do cliente Fabio
Op: 997 -> Cliente: Fabio, efetuou Compra do ativo: 1 Saldo Atual: 547.09657
Tendecia de ALTA p/ ativo 3 do cliente Julia
Tendecia de ALTA p/ ativo 3 do cliente Luiz
Op: 999 -> Cliente: Luiz, efetuou Compra do ativo: 3 Saldo Atual: 526.5175599999999
Tendecia de ALTA p/ ativo 3 do cliente Amanda
Op: 998 -> Cliente: Julia, efetuou Compra do ativo: 3 Saldo Atual: 526.5175599999999
Op: 1000 -> Cliente: Amanda, efetuou Compra do ativo: 3 Saldo Atual: 526.5175599999999
Tendecia de ALTA p/ ativo 1 do cliente Pedro
NUMERO DE OPERACOES ESGOTADO
Tendecia de ALTA p/ ativo 3 do cliente Henrique
NUMERO DE OPERACOES ESGOTADO
Tendecia de ALTA p/ ativo 3 do cliente Algusto
NUMERO DE OPERACOES ESGOTADO
Tendecia de ALTA p/ ativo 2 do cliente Rafael
NUMERO DE OPERACOES ESGOTADO
Tendecia de ALTA p/ ativo 2 do cliente Fabio
```

	A	B	C	D	E	F	G	H
1	Cliente	Data	Hora	Operação	Ativo	Valor	Saldo atual	
2	Luiz	2022.05.2(	03:20:00	Compra	1	105.784	1000.0	
3	Rafael	2022.05.2(	03:20:00	Compra	1	105.784	1000.0	
4	Fabio	2022.05.2(	03:20:00	Compra	0	0.70418	1000.0	
5	Luiz	2022.05.2(	03:20:00	Compra	2	0.63832	894.216	
6	Rafael	2022.05.2(	03:20:00	Compra	2	0.63832	894.216	
7	Fabio	2022.05.2(	03:20:00	Compra	1	105.784	999.29582	
8	Luiz	2022.05.2(	03:20:00	Compra	3	128.178	893.57768	
9	Rafael	2022.05.2(	03:20:00	Compra	3	128.178	893.57768	
10	Joao	2022.05.2(	03:20:00	Compra	3	128.178	893.57768	
11	Fabio	2022.05.2(	03:20:00	Compra	2	0.63832	893.5118200000001	
12	Fabio	2022.05.2(	03:30:00	Compra	3	128.171	892.8735	
13	Amanda	2022.05.2(	03:30:00	Compra	0	0.70409	1000.0	
14	Henrique	2022.05.2(	03:30:00	Compra	0	0.70409	1000.0	
15	Pedro	2022.05.2(	03:30:00	Compra	0	0.70409	1000.0	
16	Amanda	2022.05.2(	03:30:00	Compra	1	105.759	999.29591	
17	Henrique	2022.05.2(	03:30:00	Compra	1	105.759	999.29591	
18	Pedro	2022.05.2(	03:30:00	Compra	1	105.759	999.29591	
19	Amanda	2022.05.2(	03:30:00	Compra	2	0.63841	893.53691	
20	Henrique	2022.05.2(	03:30:00	Compra	2	0.63841	893.53691	
21	Pedro	2022.05.2(	03:30:00	Compra	2	0.63841	893.53691	

CaixaGeral

Como critério de operar comprado ou vendido, foi utilizado apenas a analise da media movel curta cruzar a longa, se cruza para cima opera comprado se cruza para baixo, opera vendido. Porém, o estudo desta análise é vasto e possui um pertinente espaço para aprimoramento.