


Controle de Estoque

Sistemas de Informação - Trabalho 1


Responsável: Rodrigo Guimarães - 140170740



Sumário


- 
- Problema;
 - Classes;
 - Interface Gráfica;
 - Aquisição do Arquivo *Listagem*;
 - Aquisição do Arquivo *Histórico*;
 - Modificação na Listagem;
 - Configuração LT, PP e FS;
 - Cálculo dos Estoques;
 - Relatório;
 - Documentação;
 - Repertório *Online*;
 - Referências.

Problema

- 
- ♦ Implementar um módulo de Sistema de Informação Gerencial (SIG) referente ao cálculo de estoque mínimo de um Sistema de Controle de Estoque (SCE).
 - ♦ Dados de entrada:
 - ♦ Listagem dos itens em estoque, com sua quantidade atual ;
 - ♦ Histórico de tais itens.




Classes

- 
- ♦ Controle:
 - ♦ ControladorEstoque;
 - ♦ Estoque;
 - ♦ Item.
 - ♦ GUI:
 - ♦ InterfaceGrafica;
 - ♦ FormatacaoPadrao;
 - ♦ FiltrosTxt.
 - ♦ Comunicação:
 - ♦ Comandos.



Interface Gráfica

- 
- ♦ Funcionalidades:
 - ♦ Possibilidade de abrir os arquivos de listagem e de histórico;
 - ♦ Exibir informações sobre o estoque: itens e histórico;
 - ♦ Editar itens, sem perder os antigos;
 - ♦ Configurar os fatores importantes: LT, PP, FS (indiretamente);
 - ♦ Habilitar itens para os cálculos de estoque (min e máx), além do cálculo, propriamente dito;
 - ♦ Gerar e salvar o relatório.



Interface Gráfica

Controlador de Estoque

Abrir Arquivos:

Listagem:

<nome_arquivo_listagem>

Abrir

Histórico:

<nome_arquivo_histórico>

Abrir

Configurações Globais:

Lead Time:

2

☐ Aplicar a todos

Período:

1

☐ Aplicar a todos

Fator Segurança:

95

☐ Aplicar a todos

Restaurar itens

Estoque:

Itens em estoque:

Estoque Min	Estoque Máx	Código	Descrição	Qty.
-------------	-------------	--------	-----------	------

Configurações:

Salvar

Histórico

Calcular

Gerar Relatório

Estoque Min:

Marcar todos

Desmarcar todos

Estoque Max:

Marcar todos

Desmarcar todos

Controle de Estoque

6 / 21

Aquisição do Arquivo *Listagem*

```
122- /**
123-  * Catalogação dos itens que constam no estoque
124-  * @param nomeArq Nome do arquivo que armazena os dados
125-  */
126- public void catalogar(String nomeArq) {
127-     try{
128-         BufferedReader leitura = new BufferedReader(new FileReader(nomeArq));
129-         String linha = leitura.readLine();
130-
131-         while(leitura.ready()){
132-             linha = leitura.readLine();
133-             String[] partes = linha.split("\\s+");
134-
135-             if (partes.length > 1){
136-                 int codigo = Integer.parseInt(partes[0]);
137-                 String descricao = partes[1];
138-                 int unidade = Integer.parseInt(partes[2]);
139-
140-                 addItem(codigo, descricao, unidade);
141-             }
142-         }
143-
144-         leitura.close();
145-
146-         ordena();
147-
148-         System.out.println("Catalogou e ordenou");
149-     } catch (IOException excessao){
150-         excessao.printStackTrace();
151-     }
152- }
```

Aquisição do Arquivo *Histórico*

```
154  /**
155   * Construção do histórico dos itens do estoque, conforme um
156   * arquivo
157   * @param nomeArq Nome do arquivo do Histórico
158   */
159  public void constroiHistorico (String nomeArq){
160      try{
161          BufferedReader leitura = new BufferedReader(new FileReader(nomeArq));
162          String linha = leitura.readLine();
163          Item item = new Item();
164
165          while(leitura.ready()){
166              linha = leitura.readLine();
167              String[] partes = linha.split("\\s+");
168
169              if (partes.length > 1){
170                  int codigo = Integer.parseInt(partes[0]);
171                  int demanda = Integer.parseInt(partes[2]);
172
173
174                  if ((codigo - 1) < itens.size())
175                      item = itens.get(codigo - 1);
176                  else
177                      item = itens.getLast();
178
179                  if (item.getCodigo() == codigo)
180                      item.addHistorico(demanda);
181
182              }
183          }
184
185          leitura.close();
186      }catch(IOException excessao){
187          excessao.printStackTrace();
188      }
189  }
```


Exibição da Tabela de Histórico

```
436  /**
437   * Construção de um painel responsável pela demonstração
438   * do histórico dos itens em estoque
439   * @param painelBase Painei antes da inserção
440   * @return Painei construído
441   */
442  private static JPanel montarPaineiHistorico (JPanel painelBase){
443      String[][] dados = {};
444      String cabecalho[] = {"Código", "Descrição", "Período", "Qnt."};
445      int larg[] = {60, 150, 60, 60};
446
447      listaS = criarPaineiTabelaS (dados, cabecalho, larg);
448
449      modeloTabelaDados2 = (DefaultTableModel) listaS.get(1);
450      tabelaDados2 = (JTable) listaS.get(2);
451      painelBase.add((JPanel) listaS.getFirst());
452      listaS.clear();
453
454      return painelBase;
455  }
```

Exibição da Tabela de Histórico

```
273  /**
274   * Atualização do conteúdo da tabela de histórico, conforme a condição atual
275   * do estoque
276   */
277  public static void atualizarTabelaHistorico () {
278      int qntItens = controle.ControladorEstoque.estoque.itens.size();
279
280      InterfaceGrafica.tabelaDados2.removeAll();
281
282      for (int lin = 0; lin < qntItens; lin++) {
283          Item itemAtual = controle.ControladorEstoque.estoque.itens.get(lin);
284
285          for (int periodo = 0; periodo < itemAtual.getHistorico().size(); periodo++) {
286              int qnt = itemAtual.getHistorico().get(periodo);
287
288              InterfaceGrafica.modeloTabelaDados2.addRow(new Object[] { itemAtual.getCodigo(),
289                                                                      itemAtual.getDescricao(),
290                                                                      periodo, qnt });
291          }
292
293          InterfaceGrafica.modeloTabelaDados2.addRow(new Object[] { null });
294      }
295  }
296  }
```

Modificação na *Listagem*

```
347- /**
348-  * Tratamento do nome do arquivo a ser salvo em HD
349-  * @param original Nome do arquivo original
350-  * @return Nome do arquivo tratado
351-  */
352- public static String tratarCaminho (String original){
353-     String tratada = "";
354-
355-     if(original.contains("_mod.txt")){
356-         System.out.println("primeiro caso");
357-
358-         int primeiraPos = original.lastIndexOf("_mod") + 4;
359-
360-         tratada = original.substring(0, primeiraPos) + "(1).txt";
361-
362-     }else if(original.contains("_mod(")){
363-         System.out.println("segundo caso");
364-         int primeiraPos = original.lastIndexOf("_mod(") + 5;
365-         int ultimaPos = original.indexOf(").txt");
366-         int valor = Integer.parseInt(original.substring(primeiraPos, ultimaPos)) + 1;
367-
368-         tratada = original.substring(0, primeiraPos);
369-         tratada += valor;
370-         tratada += ").txt";
371-
372-     }else{
373-         System.out.println("terceiro caso");
374-         tratada = original.replace(".txt", "_mod.txt");
375-     }
376-
377-     return tratada;
378- }
379- }
```

Configuração LT, PP e FS



```
280  /**
281   * Atribuição a todos os itens do estoque o mesmo período
282   * para o reabastecimento
283   * @param leadTime Tempo de reabastecimento a ser aplicado
284   */
285  public void aplicarTodosLeadTime (int leadTime){
286      for (int ind = 0; ind < itens.size(); ind++){
287          Item itemAvulso = itens.get(ind);
288
289          itemAvulso.setLeadTime(leadTime);
290      }
291  }
```



Cálculo dos Estoques



```
319  /**
320   * Cálculo do estoque mínimo e máximo para todos os itens
321   */
322  public void calcularEstoque (){
323      for (Item item : itens){
324          item.getEstoqueMin();
325          item.getEstoqueMax();
326      }
327  }
```



Cálculo dos Estoques - Mínimo



```
234  /**
235   * Valor atribuído ao nível de estoque mínimo
236   * @return Nível de estoque mínimo
237   */
238  public int getEstoqueMin () {
239      if (habilitadoEstoqueMin)
240          estoqueMin = calculaEstoqueMin ();
241      else
242          estoqueMin = 0;
243
244      return estoqueMin;
245  }
```



Cálculo dos Estoques - Máximo



```
431  /**
432   * Cálculo do nível máximo de estoque
433   * @return Nível mínimo calculado
434   */
435  private int calculaEstoqueMax() {
436      return ((getEstoqueMin() + getDemandaMedia()) * getLeadTime());
437  }
```



Gerar relatório

```
368 /**
369  * Construção do relatório do controle,
370  * disponibilizando para exibição da maneira escolhida
371  * @param data Data a ser incorporada ao relatório
372  * @return Relatório atualizado
373  */
374 public String gerarRelatorio (String data){
375     String relatorio = "";
376
377     relatorio += "Relatório de situação do estoque\n\n";
378     relatorio += "Data de expedição: " + data + "\n";
379     relatorio += "Período de referência: " + itens.getFirst().getPeriodo() + "\n\n";
380     relatorio += "-----\n";
381     relatorio += "Código\tDescrição\tQnt Existente\tDemanda Tot.\tEstoque Min\t\tEstoque Max\n";
382     relatorio += "-----\n";
383
384     for (int ind = 0; ind < itens.size(); ind++){
385         Item item = itens.get(ind);
386         relatorio +=    item.getCodigo() + "\t\t"
387                        + item.getDescricao() + "\t\t"
388                        + item.getQntExistente() + "\t\t\t\t"
389                        + item.getDemanda() + "\t\t\t\t"
390                        + item.getEstoqueMin() + "\t\t\t\t"
391                        + item.getEstoqueMax() + "\n";
392     }
393
394     relatorio += "-----\n";
395
396     return relatorio;
397 }
```


Armazenar Relatório

```
399  /**
400   * Escrita do relatório num arquivo .txt
401   */
402  public void escreverRelatorio () {
403      try {
404          DateFormat dateFormat = new SimpleDateFormat ("yyyy-MM-dd_HH:mm:ss");
405          Date date = new Date();
406          String data = dateFormat.format(date);
407
408          String relatorio = gerarRelatorio (data);
409          String caminho = "Relatórios/relatorio_" + data + ".txt";
410
411          BufferedWriter gravarArq = new BufferedWriter(new FileWriter (caminho));
412
413          gravarArq.append(relatorio + "\n");
414
415          gravarArq.close();
416
417      } catch (IOException e) {
418          e.printStackTrace();
419      }
420  }
```

Documentação

- ♦ **Index;**

All Classes

Packages

controle
gui

All Classes

Comandos
ControladorEstoque
Estoque
FiltroTxt
FormatacaoPadrao
InterfaceGrafica
Item

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Packages

Package	Description
controle	
gui	

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Repositório Online

- Todo o trabalho está disponível *online* no repositório gratuito *GitHub*, sob o link: <https://github.com/rodrigofegui/SI-2016.2-Trabalho1>;
- Neste repositório estão contidos:
 - Arquivos fontes necessários para o trabalho;
 - Documentação;
 - Arquivos *Listagem* e *Históricos* básicos, para teste;
 - Referências para o desenvolvimento;
 - Arquivos de controle.

Referências

- 
- ♦ **Oracle – Docs.** *Java™ Platform, Standard Edition 7 API Specification.*
Acessado em 2016. Disponível em:
<https://docs.oracle.com/javase/7/docs/api/>;

Executando



- ♦ O programa construído terá suas funcionalidades demonstradas agora, comentários são bem-vindos.

FIM

