**Laboratório 4**



4.8.1 Objetivo:

Além dos objetivos definidos nos laboratórios acima temos também a implementação da classe Vector, a utilização de arquivos no modo append, a utilização da classe StringBuffer e a utilização do método split da classe String.

4.8.2 Definição:

Fazer uma classe Java chamada Lab04Sistema no pacote view e copiar o conteúdo da classe Lab03Sistema. Alterar onde aparece Lab03Sistema para Lab04Sistema.

Fazer uma classe Java chamada Lab04Historico no pacote model com os atributos e métodos abaixo definidos.

|  |  |
| --- | --- |
| Atributos - privados | Descrição |
| numAge | Número da Agência – int |
| numConta | Número da Conta – int |
| dia | Dia do lançamento. Usar a data atual – int |
| mês | Mês do lançamento. Usar a data atual – int |
| ano | Ano do lançamento. Usar a data atual – int |
| hora | Hora do lançamento. Usar a hora atual – int |
| min | Minuto do lançamento. Usar a hora atual – int |
| seg | Segundo do lançamento. Usar a hora atual – int |
| codHist | Código do histórico. Usar 1 para saque e 2 para o depósito – int |
| valor | Valor do lançamento – double |
| vetOperacoes | Vetor que irá receber as transações gravadas no arquivo .hist. – Criar do tipo Vector. Este atributo será utilizado pelos métodos recuperar () e imprimir(). |

O histórico da conta deve ser gravado ao final do arquivo com o nome "AAA.NNNN.hist" no formato fixo descrito abaixo para facilitar a navegação sobre o mesmo. Onde AAA significa o número da agência e NNNN significa o numéro da conta.

A classe Lab04Historico terá os seguintes métodos definidos:

Lab04Historico (int numAge, int numConta) - Construtor que inicializa os atributos com valores recebidos como parâmetro.

void gravar (int codHist, double valor) - Grava um registro no arquivo de histórico com as informações constantes do layout. O histórico irá indicar quais as transações foram efetuadas na conta. Deve-se utilizar para o saque o código 1 e para o depósito o código 2.

void imprimir () - Imprime as informações da instância do histórico atual na tela de maneira formatada. Ver o layout abaixo da saída deste método. Este método deverá executar o método recuperarHistorico () já nas suas primeiras linhas.

void recuperarHistorico () – Recupera as operações da conta corrente e as adiciona em uma objeto da classe Vector.

Implementar nos métodos execSaque () e execDeposito () da classe Lab04Sistema uma chamada aos métodos da classe Lab04Historico, ou seja, para todas as vezes que seja efetuado um saque ou depósito com sucesso, deveremos criar um objeto da classe Lab04Historico chamando o construtor com os parâmetros código de agência e número da conta. Deve-se chamar em seguida o método gravar () com os parâmetros adequados para gravar o histórico no arquivo de históricos. Usa-se como código do histórico para o método gravar (), os códigos 1 para o saque e 2 para o depósito.

Implementar na classe Lab04Sistema a opção "5 - Extrato" que irá chamar o método execExtrato(). O método execExtrato () deve pedir o número da agência e conta e mostrar o extrato no layout especificado abaixo. Deve-se usar o método imprimir () da classe Lab04Historico para mostrar cada transação da conta corrente. Não esquecer que o método imprimir () da classe Lab04Historico deverá chamar o método recuperarHistorico () da própria classe que irá ler os movimentos da conta.

Implementar a opção na classe Lab04Sistema "8 – Remover Conta Corrente" que irá chamar o método execRemoverContaCorrente (). O método deve pedir o número da agência e conta que será removida. Em seguida instanciar um objeto da classe Lab03ContaCorrente e executar o método removerArquivo ().

Implementar o método removerArquivo () na classe Lab03ContaCorrente com a seguinte composição:

public boolean removerArquivo ()

{

File tArq1;

tArq1 = new File (numAge + "." + numConta + ".dat");

tArq1.delete();

tArq1 = new File(numAge + "." + numConta + ".hist");

tArq1.delete();

return true;

}

Este método será criado para permitir que no diagrama de classes tenhamos definido a associação de agregação composta.

LAYOUT MENU PRINCIPAL

1 - Cadastramento

2 - Saque

3 - Deposito

4 – Consulta

5 – Extrato

8 – Remove Conta Corrente

9 - Fim

Entre com a opcao desejada:

LAYOUT "ExecutaExtrato"

------------------------------------------

Agencia : 10

Parte impressa pelo método Imprime da classe Lab03ContaCorrente

Conta : 10

Nome : Francyele S. Freire David

Saldo : R$ 100,00

Parte impressa pelo método imprimir () da classe Lab04Histórico

0010 0000010 16/05/2008 - 07:09:57 - Saque caixa R$ 20,00

0010 0000010 16/05/2008 - 07:10:09 - Deposito dinheiro R$ 500,00

------------------------------------------

LAYOUT ARQUIVO

GGGG NNNNNNN DD MM AAAA HH MM SS XXX VVVVVVVVVVVVVV

Onde

GGGG - Código da Agência com 4 dígitos

NNNNNNN - Número da Conta com 7 dígitos

DD - Dia do Lançamento com 2 dígitos

MM - Mês do Lançamento com 2 dígitos

AAAA - Ano do Lançamento com 4 dígitos

HH - Hora do Lançamento com 2 dígitos

MM - Minuto do Lançamento com 2 dígitos

SS - Segundo do Lançamento com 2 dígitos

XXX - Código do Histórico, sendo 1 igual a saque e 2 igual a deposito. Escrever as literais ao invés do número.

VVVVVVVVVVVVVV - Valor do Lançamento

**OBS.: Todos os campos devem estar separados por um espaço. É importante manter desta forma para que o método split () da classe String possa separar estes valores em partes.**

4.8.2.1 Sugestões:

Para o método recuperarHistorico () utilizar o código da listagem de programa 30:

 Listagem de programa 30 – Método para realizar a recuperação do arquivo de histórico.

public void recuperarHistorico() {

FileReader tArq1;

BufferedReader tArq2;

String tLinha = null;

try {

// Operação I - Abrir o arquivo

tArq1 = new FileReader(numAge + "." + numConta + ".hist");

tArq2 = new BufferedReader(tArq1);

// Operação III - Ler atributo/valor e colocar na matriz

while (true) {

tLinha = tArq2.readLine();

if (tLinha == null)

break;

// Criar vetOperacoes como um atributo do tipo Vector

vetOperacoes.add(tLinha);

}

// Operação IV - Fechar o arquivo

tArq2.close();

}

catch (FileNotFoundException e)

{

System.*out*.println("\n Conta sem movimento \n\n");

}

catch (IOException tExcept)

{

tExcept.printStackTrace();

}

}

Para o método gravar () que irá gravar uma transação da conta utilizar o código da listagem de programa 31:

 Listagem de programa 31 – Método para realizar a gravação do arquivo de histórico.

public boolean gravar(int p\_hist, double p\_valor) {

FileWriter tArq1;

PrintWriter tArq2;

try {

// Operação I - Abrir o aquivo

tArq1 = new FileWriter(numAge + "." + numConta + ".hist",true);

tArq2 = new PrintWriter(tArq1);

Date hoje = new Date ();

Calendar cal = new GregorianCalendar();

cal.setTime(hoje);

dia = cal.get(Calendar.*DAY\_OF\_MONTH*);

// O mês em Java inicia com 0

mes = cal.get(Calendar.*MONTH*) + 1;

ano = cal.get(Calendar.*YEAR*);

hora = cal.get(Calendar.*HOUR*);

min = cal.get(Calendar.*MINUTE*);

seg = cal.get(Calendar.*SECOND*);

tArq2.print(numAge + " ");

tArq2.print(numConta + " ");

tArq2.print(dia + " ");

tArq2.print(mes + " ");

tArq2.print(ano + " ");

tArq2.print(hora + " ");

tArq2.print(min + " ");

tArq2.print(seg + " ");

tArq2.print(p\_hist + " ");

tArq2.println(p\_valor);

// Operação III - Fechar o arquivo

tArq2.close();

return true;

} catch (IOException tExcept) {

tExcept.printStackTrace();

return false;

}

}

Para o arquivo estar aberto no formato append passar true no segundo parâmetro do objeto do tipo FileWriter. Porém esta informação já esta presente no exemplo do método gravar (), apresentado anteriormente. Ex.: tArq1 = new FileWriter(numAge+ "." + numConta + ".hist",true); Para a formação da data do dia de hoje utilizar o código:

Date hoje = new Date ();

Calendar cal = new GregorianCalendar();

cal.setTime(hoje);

dia = cal.get(Calendar.DAY\_OF\_MONTH);

mes = cal.get(Calendar.MONTH);

ano = cal.get(Calendar.YEAR);

hora = cal.get(Calendar. HOUR\_OF\_DAY);

min = cal.get(Calendar.MINUTE);

seg = cal.get(Calendar.SECOND);

Porém esta informação já esta presente no exemplo do método gravar, apresentado anteriormente. Para imprimir o histórico de uma conta corrente, deve-se inicialmente instanciar a classe Lab04Historico na classe Lab04Sistema usando o construtor com 2 parâmetros, executar o método imprimir da classe Lab04Historico que por sua vez irá executar o método recuperarHistorico () também da classe Lab04Historico. O método recuperarHistorico () irá preencher um objeto da classe Vector com os movimentos da conta corrente apresentado no arquivo .hist. No método imprimir da classe Lab04Historico implementar um loop para que cada linha do movimento seja impresso na tela de acordo com o LAYOUT "ExecutaExtrato" já apresentado. Utilizar o método split da classe String para separar os valores de cada linha do objeto Vector.

Na apresentação do saldo devemos: Formatar o saldo da conta que esta sendo impresso no método imprimir () utilizando a classe NumberFormat. Vejamos o exemplo:

valor = Double.*parseDouble*(tSplit[9]);

NumberFormat formatter = DecimalFormat.getCurrencyInstance

(new Locale("pt","BR"));

formatter.setMinimumFractionDigits(2);

System.out.println(formatter.format(valor));

Para a formatação e apresentação da agencia, conta entre outros utilizar o seguinte exemplo:

NumberFormat formatter;

StringBuffer sb = new StringBuffer ();

formatter = new DecimalFormat("0000");

sb.append(String.*valueOf*(formatter.format (numAge)));

sb.append(" ");

formatter = new DecimalFormat("0000000");

sb.append(String.*valueOf*(formatter.format (numConta)));

sb.append(" ");