Código Java:

TrabalhoEDfinal - package pacote.de.classes;

==========

```
package pacote.de.classes;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class ProgramaEDJava {
	public static void main (String args[]) throws NumberFormatException, IOException{
		int filialInicial, filialFinal, dataInicial, dataFinal;
		Filiais arvore = new Filiais();
		Scanner teclado = new Scanner(System.in);

		//Chama carregarestrutura, e carrega o arquivo texto na estrutura.
		long tempoInicial = System.currentTimeMillis();
		carregaestrutura("dados.txt", arvore);
		System.out.println("O metodo carregaEstrutura executou em " +
(System.currentTimeMillis() - tempoInicial) +
				" milisegundos");

		System.out.println("Trabalho de Implementação - Estrutura de Dados");
		System.out.println("Ricardo George Bhering");

	System.out.println("==================================================
==============");
```

```
		System.out.println("=========================Menu de Opções==============================");


		//Cria o menu de opções do programa.
		while(true){
			System.out.println("Opção 1: Total de vendas das filiais por inervalo de Filiais.");
			System.out.println("Opção 2: Total de vendas das filiais por inervalo de Filiais e");
			System.out.println("intervalo de perído. ");
			System.out.println("Opção 3: Total de vendas de todas as filiais por intervalo de período.");


			int opcao;
			System.out.println("Entre com a opção desejada: \n " );
			opcao = teclado.nextInt();


			switch (opcao){



			case 1:
				System.out.println("Entre com o valor da (Filial Inicial)");
				filialInicial = teclado.nextInt();
				System.out.println("Entre com o valor da (Filial Final");
				filialFinal = teclado.nextInt();
				tempoInicial = System.currentTimeMillis();
				System.out.println("Saída de dados da opção 1: Exercicio 1\n");
				float total = arvore.relatorioVendas1(filialInicial, filialFinal);
				System.out.println("O metodo relatorioVendas1 executou em " + (System.currentTimeMillis() - tempoInicial) +
							" milissegundos");

	System.out.println("=================================================================\n");
```

```
                        case 2:
                                System.out.println("Entre com o valor da (Filial Inicial)");
                                filialInicial = teclado.nextInt();
                                System.out.println("Entre com o valor da (Filial Final");
                                filialFinal = teclado.nextInt();
                                System.out.println("Formato Ex: 201701 (AnoMes)");
                                System.out.println("Entre com o valor da (Data Inicial)");
                                dataInicial = teclado.nextInt();
                                System.out.println("Entre com o valor da (Data Final");
                                dataFinal = teclado.nextInt();
                                tempoInicial = System.currentTimeMillis();
                                System.out.println("Exercicio 2\n");
                                arvore.relatorioVendas2(filialInicial, filialFinal, dataInicial,
dataFinal);
                                System.out.println("O metodo relatorioVendas2 executou em "
+ (System.currentTimeMillis() - tempoInicial) +
                                                " milissegundos");

        System.out.println("==================================================
==============\n");
                        case 3:
                                System.out.println("Formato Ex: 201701 (AnoMes)");
                                System.out.println("Entre com o valor da (Data Inicial)");
                                dataInicial = teclado.nextInt();
                                System.out.println("Entre com o valor da (Data Final)");
                                dataFinal = teclado.nextInt();
                                tempoInicial = System.currentTimeMillis();
                                System.out.println("Exercicio 3\n");
                                arvore.relatorioVendas3(dataInicial, dataFinal);
                                System.out.println("O metodo relatorioVendas3 executou em "
+ (System.currentTimeMillis() - tempoInicial) +
                                                " milissegundos");
```

```
		System.out.println("=================================================
===============\n");


				default:

					System.out.println("\nVocê digitou uma operação
inválida.\n");

				}

			}


		}//Lê o arquivo informado e chama inserir.

		static void carregaestrutura(String arquivo, Filiais filiais) throws
NumberFormatException, IOException {

			BufferedReader ler = new BufferedReader(new FileReader(new File(arquivo)));

			String linha;

			while((linha = ler.readLine()) != null){

				Integer filial  = Integer.valueOf(linha.split(";")[0]);

				Integer ano_mes  = Integer.valueOf(linha.split(";")[1]);

				Integer vendedor  = Integer.valueOf(linha.split(";")[2]);

				float valorVenda  = Float.valueOf(linha.split(";")[3]);

				filiais.inserir(filial,ano_mes, vendedor, valorVenda);

			}

			ler.close();

		}

}
```

```
package pacote.de.classes;


public class Filiais {

		private NoFilial raiz = null;
```

```
	//Retorna a altura da árvore
	private static int altura( NoFilial no ) {
		return no == null ? -1 : no.altura;
	}

	//Retorna o maior valor ente altura-esquerda e altura-direita..
	private static int maximo( int altura_esq, int altura_dir ) {
		return altura_esq > altura_dir ? altura_esq : altura_dir;
	}
	//Retorna o fator de balanceamento da árvore
	private int getFatorBalanceamento (NoFilial no) {
		return altura( no.esquerda ) - altura( no.direita );
	}//Recebe os parametros para inclusão a partir do carrega estrutura.
	public boolean inserir (int filial, int ano_mes, int cod_funcionario, float venda) {
		//raiz do conteiner
		raiz = inserir (filial, ano_mes, cod_funcionario, venda, raiz);
		return true;
	}//Faz os testes para inserir.
	private NoFilial inserir (int filial, int ano_mes, int cod_funcionario, float venda, NoFilial
no) {

		if( no == null ) no = new NoFilial( filial, ano_mes, cod_funcionario, venda, null,
null );
		else if( filial<no.chave ) no.esquerda = inserir( filial, ano_mes, cod_funcionario,
venda, no.esquerda );
		else if( filial>no.chave ) no.direita = inserir( filial, ano_mes, cod_funcionario,
venda, no.direita );

		else if( no.chave == filial){
```

```
                    no.arvore_ano_mes.inserir(ano_mes, cod_funcionario, venda);
            } //Finalizada a inserção testa o balanceamento da árvore.
            no = balanceia (no);
            return no;
        }//Recebe de retorno do método (getFatorBalanceamento) e executa as rotações para balanceamento.
        public NoFilial balanceia (NoFilial no) {
            if ( getFatorBalanceamento(no) == 2 ) {
                if (getFatorBalanceamento (no.esquerda)>0) no = rotacionarDireita( no );
                else no = duplaRotacaoDireita( no );
            }

            else if ( getFatorBalanceamento(no) == -2 ) {
                if ( getFatorBalanceamento(no.direita)<0 ) no = rotacionarEsquerda( no );
                else no = duplaRotacaoEsquerda( no );
            }

            no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
            return no;
        }
        //Faz Rotação simples a direita
        private static NoFilial rotacionarDireita( NoFilial no ) {
            NoFilial aux = no.esquerda;
            no.esquerda = aux.direita;
            aux.direita = no;
            no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
            aux.altura = maximo( altura( aux.esquerda ), no.altura ) + 1;
            return aux;
        }
        //Rotação simples à esquerda
```

```
private static NoFilial rotacionarEsquerda( NoFilial no ) {
        NoFilial aux = no.direita;
        no.direita = aux.esquerda;
        aux.esquerda = no;
        no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
        aux.altura = maximo( altura( aux.direita ), no.altura ) + 1;
        return aux;
}
//Rotação dupla à direita
private static NoFilial duplaRotacaoDireita( NoFilial no ) {
        no.esquerda = rotacionarEsquerda( no.esquerda );
        return rotacionarDireita( no );
}
//Rotação dupla à esquerda
private static NoFilial duplaRotacaoEsquerda( NoFilial no ) {
        no.direita = rotacionarDireita( no.direita );
        return rotacionarEsquerda( no );
}
public NoFilial busca(int valor) {
        return busca(raiz,valor);
}
protected NoFilial busca(NoFilial no, int valor) {
        while (no != null) {
                if (valor==no.chave) return no;
                else if (valor<no.chave) no = no.esquerda;
                else no = no.direita;
        }

        return null;
}
```

```
	//Verifica se os valores informados estão ordenados para executar a busca.
	public float relatorioVendas1(int filial1, int filial2){
		if(maximo(filial1, filial2)==filial1){
			int aux = filial1;
			filial1 = filial2;
			filial2 = aux;
		}

		return vendasPorIntervaloFilial(this.raiz, filial1, filial2);
	}
	//Verifica se os valores informados estão ordenados para executar a busca.
	public void relatorioVendas2(int filial1, int filial2, int data1, int data2){
		if(maximo(filial1, filial2)==filial1){
			int aux = filial1;
			filial1 = filial2;
			filial2 = aux;
		}
		vendasPorDataFilial(this.raiz, filial1, filial2, data1, data2);
	}
	public void relatorioVendas3(int data1, int data2){
		vendasPorData(this.raiz, data1, data2);
	}//Buscas
	private float vendasPorIntervaloFilial(NoFilial no, int filial1, int filial2){
		if (no == null) return 0;
		float totalzudo = 0;
		if (no.chave>=filial1 && no.chave<=filial2){//ano_mes = chave.

			System.out.println("Filial "+no.chave+":
"+no.arvore_ano_mes.getTotalzudo());
			totalzudo += no.arvore_ano_mes.getTotalzudo();
		}
```

```
            if (no.chave>=filial1) totalzudo += vendasPorIntervaloFilial(no.esquerda, filial1,
filial2);
            if (no.chave<=filial2) totalzudo += vendasPorIntervaloFilial(no.direita, filial1,
filial2);
            if (no == this.raiz) System.out.println("\nTotal geral de vendas:
"+totalzudo+"\n\n");
            return totalzudo;
        }

        private float vendasPorDataFilial(NoFilial no, int filial1, int filial2, int data1, int data2){
            if (no == null) return 0;
            float totalzudo = 0;
            if (no.chave>=filial1 && no.chave<=filial2){
                System.out.println("Filial "+no.chave+": ");
                totalzudo += no.arvore_ano_mes.vendasPorIntervaloAno_mes(data1,
data2);
            }
            if (no.chave>=filial1) totalzudo += vendasPorDataFilial(no.esquerda, filial1,
filial2, data1, data2);
            if (no.chave<=filial2) totalzudo += vendasPorDataFilial(no.direita, filial1, filial2,
data1, data2);
            if (no == this.raiz) System.out.println("\nTotal geral de vendas: "+totalzudo);
            return totalzudo;
        }
        private float vendasPorData(NoFilial no, int data1, int data2){
            if (no == null) return 0;
            float totalzudo = 0;
            System.out.println("Filial "+no.chave+": ");
            totalzudo += no.arvore_ano_mes.vendasPorIntervaloAno_mes(data1, data2);
            totalzudo += vendasPorData(no.esquerda, data1, data2);
            totalzudo += vendasPorData(no.direita, data1, data2);
            if (no == this.raiz) System.out.println("\nTotal geral de vendas por data:
"+totalzudo);
            return totalzudo;
```

```
	}
}
```

```
package pacote.de.classes;

public class NoFilial {

	protected int altura;
	protected int chave;//filial

	protected NoFilial esquerda, direita;
	protected AnoMes arvore_ano_mes;
	//Construtor - inicializa
	public NoFilial ( int filial, int ano_mes, int cod_funcionario, float venda ) {
		this( filial, ano_mes, cod_funcionario, venda, null, null );
	}
	public NoFilial ( int filial, int ano_mes, int cod_funcionario, float venda, NoFilial esq,
NoFilial dir ) {
		chave = filial;

		esquerda = esq;
		direita = dir;
		altura   = 0;
		arvore_ano_mes = new AnoMes();
		arvore_ano_mes.inserir(ano_mes, cod_funcionario, venda);
	}
}
```

```
package pacote.de.classes;

public class AnoMes {
	private NoAnoMes raiz = null;
```

```
	//Retorna a altura da árvore
	private static int altura( NoAnoMes no ) {
		return no == null ? -1 : no.altura;
	}

	//Retorna o maior valor ente altura-esquerda e altura-direita.
	private static int maximo( int altura_esq, int altura_dir ) {
		return altura_esq > altura_dir ? altura_esq : altura_dir;
	}
	//Retorna o fator de balanceamento da árvore
	private int getFatorBalanceamento (NoAnoMes no) {
		return altura( no.esquerda ) - altura( no.direita );
	}
	public boolean inserir (int ano_mes, int cod_funcionario, float venda) {
		raiz = inserir (ano_mes, cod_funcionario, venda, raiz);
		return true;
	}
	private NoAnoMes inserir (int ano_mes, int cod_funcionario, float venda, NoAnoMes
no) {
		if( no == null ) no = new NoAnoMes( ano_mes, cod_funcionario, venda, null,
null );
		else if( ano_mes<no.chave ) no.esquerda = inserir( ano_mes, cod_funcionario,
venda, no.esquerda );
		else if( ano_mes>no.chave ) no.direita = inserir( ano_mes, cod_funcionario,
venda, no.direita );
		else if( no.chave == ano_mes){

			no.lista_vendas.adicionar(cod_funcionario, venda);
		}
		no = balanceia (no);
```

```
            return no;
        }
        public NoAnoMes balanceia (NoAnoMes no) {
            if ( getFatorBalanceamento(no) == 2 ) {
                if (getFatorBalanceamento (no.esquerda)>0) no = rotacionarDireita( no
);
                else no = duplaRotacaoDireita( no );
            }
            else if ( getFatorBalanceamento(no) == -2 ) {
                if ( getFatorBalanceamento(no.direita)<0 ) no = rotacionarEsquerda( no
);
                else no = duplaRotacaoEsquerda( no );
            }
            no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
            return no;
        }
        //Faz Rotação simples a direita
        private static NoAnoMes rotacionarDireita( NoAnoMes no ) {
            NoAnoMes aux = no.esquerda;
            no.esquerda = aux.direita;
            aux.direita = no;
            no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
            aux.altura = maximo( altura( aux.esquerda ), no.altura ) + 1;
            return aux;
        }
        //Rotação simples à esquerda
        private static NoAnoMes rotacionarEsquerda( NoAnoMes no ) {
            NoAnoMes aux = no.direita;
            no.direita = aux.esquerda;
            aux.esquerda = no;
            no.altura = maximo( altura( no.esquerda ), altura( no.direita ) ) + 1;
            aux.altura = maximo( altura( aux.direita ), no.altura ) + 1;
```

```
        return aux;
}
//Rotação dupla à direita
private static NoAnoMes duplaRotacaoDireita( NoAnoMes no ) {
        no.esquerda = rotacionarEsquerda( no.esquerda );
        return rotacionarDireita( no );
}
//Rotação dupla à esquerda
private static NoAnoMes duplaRotacaoEsquerda( NoAnoMes no ) {
        no.direita = rotacionarDireita( no.direita );
        return rotacionarEsquerda( no );
}

public NoAnoMes busca(int ano_mes) {
        return busca(raiz,ano_mes);
}

protected NoAnoMes busca(NoAnoMes no, int ano_mes) {
        while (no != null) {
                if (ano_mes==no.chave) return no;
                else if (ano_mes<no.chave) no = no.esquerda;
                else no = no.direita;
        }

        return null;
}

public float vendasPorIntervaloAno_mes(int ano_mes1, int ano_mes2){
        if(maximo(ano_mes1, ano_mes2)==ano_mes1){
                int aux = ano_mes1;
                ano_mes1 = ano_mes2;
```

```
                ano_mes2 = aux;
            }
            return vendasPorIntervaloAno_mes(this.raiz, ano_mes1, ano_mes2);
        }//Busca
        private float vendasPorIntervaloAno_mes(NoAnoMes no, int ano_mes1, int
ano_mes2){
            float soma = 0;
            if (no == null) return soma;
            if (no.chave>=ano_mes1 && no.chave<=ano_mes2){

                System.out.println("          Data "+no.chave+": vendeu
"+no.lista_vendas.somaVenda());
                soma += no.lista_vendas.somaVenda();
            }
            if (no.chave>=ano_mes1) soma += vendasPorIntervaloAno_mes(no.esquerda,
ano_mes1, ano_mes2);
            if (no.chave<=ano_mes2) soma += vendasPorIntervaloAno_mes(no.direita,
ano_mes1, ano_mes2);
            if (no == this.raiz) System.out.println("\n          Total de vendas: "+soma);
            return soma;
        }

        protected float getTotalzudo(){
            return getTotalzudo(this.raiz);
        }
        private float getTotalzudo(NoAnoMes no){
            float soma =0;
            if (no == null) return soma;
            soma += no.lista_vendas.somaVenda();
            soma += getTotalzudo(no.esquerda);
            soma += getTotalzudo(no.direita);
            return soma;
```

```
	}
}
```

```
package pacote.de.classes;

public class NoAnoMes {

	protected int altura;
	protected int chave; //ano_mes

	protected ListaVenda lista_vendas;
	protected NoAnoMes esquerda, direita;

	//Construtor - inicializa
	public NoAnoMes ( int ano_mes, int cod_vendedor, float venda ) {
		this( ano_mes, cod_vendedor, venda, null, null );
	}
	public NoAnoMes ( int ano_mes, int cod_vendedor, float venda, NoAnoMes esq,
NoAnoMes dir ) {
		chave = ano_mes;

		esquerda = esq;
		direita = dir;
		altura   = 0;
		lista_vendas = new ListaVenda();
		lista_vendas.adicionar(cod_vendedor, venda);
	}
}
```

```
package pacote.de.classes;


public class ListaVenda {
	protected int codFunc;
	protected double valor;
	protected ListaVenda proximo;

	public ListaVenda(){
		codFunc = -1;
		valor = 0.0;
		proximo = null;
	}

	protected void adicionar(int vendedor, double valorVenda){
		if(this.codFunc == -1 ){
			this.codFunc = vendedor;
			this.valor = valorVenda;
		}else{
			ListaVenda novaVenda = new ListaVenda();
			novaVenda.codFunc = vendedor;
			novaVenda.valor = valorVenda;
			ListaVenda aux = this;
			ListaVenda temp = aux.proximo;
			aux.proximo=novaVenda;
			novaVenda.proximo = temp;
		}
	}
	protected float somaVenda(){
		ListaVenda aux = this;
		float soma = 0;
		while(aux!=null){
```

```
                soma += aux.valor;
                aux = aux.proximo;
            }
            return soma;
        }
    }
```