Anexo – Linhas de Código: Objeto Viga

```
	package entity;

import java.text.DecimalFormat;
import java.util.List;

import javax.swing.JOptionPane;

import calculus.Interpolacao;
import main.CalculoViga;

public class Viga {

	public Viga(Double i, Double j, Double l, Double gamaS, AcoArmaduraAtiva acoArmaduraAtiva,
			AcoArmaduraPassiva acoArmaduraPassiva, String classeconcreto, Integer fck, Double umidade, Double gamaEndurecimento,
			Double mPermanentes, Double mAcidentais) throws Exception {
		super();
		this.base = i;
		this.altura = j;
//		this.hipotese = hipotese;
		this.l = l;
		this.mPermanentes = mPermanentes; //entrar com esses dados
		this.mAcidentais = mAcidentais; //entrar com esses dados
		this.area = base * altura;
```

```
this.gamaS = gamaS;
this.yCInf = altura/2 ;
this.yCsup = altura/2;
this.inerciaX = base * Math.pow(altura, 3)/12;
this.wCInf = inerciaX/yCInf;
this.wCSup = inerciaX/yCsup;
this.raioCInf = wCInf/area;
this.raioCSup = wCSup/area;
if(classeconcreto == "C30")
        fck = 30;
else if(classeconcreto == "C35")
        fck = 35;
else if(classeconcreto == "C40")
        fck = 40;
else if(classeconcreto == "C45")
        fck = 45;
else if(classeconcreto == "C50")
        fck = 50;
// checar segundo parâmetro.
this.concreto = new Concreto(area, classeconcreto, fck);
this.mG1 = concreto.getqG1() * Math.pow (l,2)/8;
this.mGI = mG1 + mPermanentes + mAcidentais;
this.mGiG1 = mGI - mG1;
this.acoArmaduraAtiva = acoArmaduraAtiva;



//Contas para processo K6
```

```
		//MSD DE KNm PARA MNm

		this.mSD = (1.4*mPermanentes + 1.3*mAcidentais)/1000; //ALTERAR
MSD

		//this.mSD = 0.1699;


		this.y0 = acoArmaduraAtiva.getCobrimentoMinimo() +
(acoArmaduraAtiva.getNominal()/2) ;

		this.dP = altura - this.y0;

		this.k6 = (base * (Math.pow(dP, 2)))/(this.mSD);




		this.betaX = Double.parseDouble(JOptionPane.showInputDialog("Valor
de k6: "+ k6 + "\nDigite o Valor de BetaX : "));

		this.betaZ = Double.parseDouble(JOptionPane.showInputDialog("Digite
o Valor de BetaZ: "));

		this.epsilonCD =
Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de Ecd : "));
//inserir em %

		this.deltaEpsilonPD =
Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de DELTAEpd
(%): ")); //inserir em %




		// RECEBER OS SEGUINTES DADOS: BETAX / BETAZ / EPSILON
CD / DELTAEPSILONPD

		//Condições para verificar o domínio

		if(deltaEpsilonPD == 10 && (epsilonCD >= 3.5 && epsilonCD > 0))

		{

			this.dominio = 2;

		}
```

```
else if(epsilonCD == 3.5 && (deltaEpsilonPD <= 10 && deltaEpsilonPD > 0))

{

	this.dominio = 3;

}


this.epsilonPD = acoArmaduraAtiva.getPreAlongamento() + deltaEpsilonPD; //resultado em %

this.tensaoAcoAtivo = Double.parseDouble(JOptionPane.showInputDialog("Valor de Epd : " + epsilonPD + "\nDigite o Valor de tensao Aco Ativo: "));









//Ap pg64 H

//quantidade minima a ser utilizado

this.areaAcoAtivoMinima = (mSD * Math.pow(10, 4))/ (betaZ * dP * tensaoAcoAtivo);

//pg 64 letra I

Double ntd = mSD/ (betaZ * dP);

Double ntd1 = (areaAcoAtivoMinima * tensaoAcoAtivo)/10000;//divindo para transformar

//Se esses cálculos não estiverem iguais, algo no cálculo está errado.

DecimalFormat decimal = new DecimalFormat( "0.0000" );

if(decimal.format(ntd) == decimal.format(ntd1)){

	this.forcaTracao = ntd;

} else {
```

```
                    //JOptionPane.showMessageDialog(null, "Calculo Errado, não é possível concluir a requisição!", "Erro", JOptionPane.ERROR_MESSAGE);
//                  throw new Exception("");
                }
                //numero minimo de cordoalhas a ser utilizado
                this.quantidadeAco = areaAcoAtivoMinima/(acoArmaduraAtiva.getArea()*Math.pow(10, 4));

                this.quantidadeCordoalhas2 = Double.parseDouble(JOptionPane.showInputDialog("Nº mín de Cordoalhas: " + quantidadeAco + "\nDigite a quantidade: "));
                //area de aco ativo final
                    this.areaAcoAtivoFinal =  quantidadeCordoalhas2 * (acoArmaduraAtiva.getArea()*Math.pow(10, 4));
                //FORMULA 36
                    this.forcaTracao = areaAcoAtivoMinima * tensaoAcoAtivo;


                this.acoArmaduraPassiva = acoArmaduraPassiva;


//                  this.tensaoAcoAtivo = tensaoAcoAtivo; //pg63


                // Interpolacao interpolacao = new Interpolacao(null, null, null, null, null);
```

```
/*	//Primeira hipótese acontece quando o cálculo for apenas da armadura ativa
		if(this.hipotese == 1) {




			//Esperando resposta do João Vitor à respeito do calculo de y0s
				this.k6 = (base - Math.pow(dP, 2))/this.mSD;


				//Interpolação
				interpolacao.interpolar(valorPopAnterior, k6, valorPopPosterior, valorPopAnterior2, valorPopPosterior);


				if(deltaEpsilonPD == 0.1 && (epsilonCD >= 0.035 && epsilonCD > 0))
				{
					this.dominio = 2;
				}
				else if(epsilonCD == 0.035 && (deltaEpsilonPD <= 0.1 && deltaEpsilonPD > 0))
				{
					this.dominio = 3;
				}


				this.epsilonPD = acoArmaduraAtiva.getPreAlongamento() + deltaEpsilonPD;
```

```
                        //pg 64 letra I
                        Double ntd = mSD/ (betaZ * dP);
                        Double ntd1 = areaAcoAtivoMinima * tensaoAcoAtivo;
                        //Se esses cálculos não estiverem iguais, algo no cálculo
está errado.
                        if(ntd == ntd1){
                                this.forcaTracao = ntd;
                        } else {
                                System.out.println("Cálculo Errado");
                        }



                }
                // Início da segunda hipótese, quando tiver aço de armadura passiva.
                else if (this.hipotese == 2){
                        if(acoArmaduraPassiva.getDiametroBarra() != 0 &&
acoArmaduraAtiva.getNominal() != 0){
                                this.y0s =  acoArmaduraAtiva.getCobrimentoMinimo() +
(acoArmaduraPassiva.getDiametroBarra()/2);


                                // DS
                                this.dS = altura - this.y0;


                                //Posição da linha neutra em relação a borda comprimida
                                this.x = betaX * dP;


                                //Deformação da armadura ativa
```

```
                    this.epsilonSD = (dS - x/dP - x) * deltaEpsilonPD;

                    this.tensaoAcoPassivo = tensaoAcoPassivo;

                    this.areaAcoArmaduraPassiva = (mSD * Math.pow(10,
4))/ betaZ * dS * tensaoAcoPassivo;

                    this.forcaTracao = areaAcoAtivoMinima *
tensaoAcoAtivo + areaAcoArmaduraPassiva * tensaoAcoPassivo;


                }



        }

        */

            //INICIO DA VERIFICACAO DOMINIO 3 ELU

            //tensão aço armadura ativa

            this.tensaoAcoPd = acoArmaduraAtiva.getFpyk() / gamaS;

            //tensão aço armadura passiva

            //this.tensaoAcoSd = acoArmaduraPassiva.getfyk() / gamaS;

            //Calculo das forças de tracao

            //FORÇA de Tração na armadura ativa

            this.forcaTracaoAtivo = ((areaAcoAtivoFinal*Math.pow(10, -4)) *
tensaoAcoPd); //MPA mathpow

            //Força de tração na armadura passiva

            //this.forcaTracaoPassivo = acoArmaduraPassiva.getArea() *
tensaoAcoSd;

            //Força de tração total NTD

            //this.forcaTracaoTotal = forcaTracaoAtivo + forcaTracaoPassivo;

            this.forcaTracaoTotal = forcaTracaoAtivo;

            //PG 67 Tensao no concreto

            this.tensaoCD = 0.85 * (concreto.getFck()/1.4);//MPA
```

```
//PG68 Aréa comprimida (Acc)
this.areaComprimida = forcaTracaoTotal/tensaoCD; //m²

//pg68 altura diagrama compressao
this.yAlturaDiagramaCompressao = areaComprimida/base; //m

//pg68 Posicao linha Neuta X OBS: Criar o lambaconcreto
this.posicaoLinhaNeutra = yAlturaDiagramaCompressao/0.8; //m

//pg68 LAMBA CONCRETO = 0.8

//EPSLONCU
this.epsilonCU = 3.5; // unidade em %

//PG69 INICIO DEFORMAÇÕES ARMADURAS ATIVAS
this.deltaEpsilonPD2 = ((dP - posicaoLinhaNeutra
)/posicaoLinhaNeutra)*epsilonCU;

//PG69 EpsilonPD
this.defEpsilonPD = acoArmaduraAtiva.getPreAlongamento() +
deltaEpsilonPD2;

//pg69 EpsilonPyD
```

```
this.epsilonPyD =
tensaoAcoPd/acoArmaduraAtiva.getElasticidadeacoativo();


//COM OS RESULTADOS ACIMA, E FEITO A VERIFICACAO DOS
ITENS C e D da verficiacao ELU dominio3




//Inicio das deformações das armaduras passivas

//pg69 Deformação A.Ativa

//this.defepsilonSD = ((dS - x)/x)*epsilonCU;

//Deformaçao simplificada A.Ativa

//this.epsilonSYD = tensaoAcoSd /
acoArmaduraPassiva.getElasticidadeacopassivo();

//COM ESSES RESULTADOS VERIFICA-SE CONDICAO B ELU
DOMINIO 3




//pg69 Posição do CG da área comprimida de altura y com tensão
uniforme do concreto

this.ylinha = yAlturaDiagramaCompressao / 2;


//pg70

//this.zs = dS - ylinha;

this.zp = dP - ylinha;

//pg70 Momento resistente de cálculo

//this.MRD = (forcaTracaoPassivo * zs) + (forcaTracaoAtivo * zp);

this.MRD = forcaTracaoAtivo*zp;

if(!(MRD >= mSD)){
```

```
                    JOptionPane.showMessageDialog(null, "Momento resistente de
cálculo não atende as solicitações", "Erro", JOptionPane.ERROR_MESSAGE);
//                  throw new Exception("");
                }
                //FIM DA VERIFICAÇÃO DO ESTADO LIMITE ÚLTIMO



                //INICIO ELS
                if(concreto.getFck() > 20 && concreto.getFck() < 50){
                    this.fctm = 0.3 * Math.pow(concreto.getFck(), 0.66666);
                } else {
                    this.fctm = 2.12 * Math.log(1+(0.11 * concreto.getFck()));
                }

                this.fctkf = (1.428 * 0.7 * fctm)*1000;//KPA
                //força de compressao (negativa)
                this.npinfinito = (-((areaAcoAtivoFinal*Math.pow(10, -4)) *
acoArmaduraAtiva.getPreAlongamento() *
(acoArmaduraAtiva.getElasticidadeacoativo()/Math.pow(10, 3))));//Verificar unidade
(x100)



                this.ep = (altura/2) - y0;



                this.mCQP = mGI;
                this.mCF = mGI;
```

```
		this.tensaoFibraSuperiorCF = -(npinfinito/area + (npinfinito * ep)/wCSup
+ mCF/wCSup);


		this.tensaoFibraInferiorCF = -(npinfinito/area + (npinfinito * ep)/wCInf +
mCF/wCInf);


		this.tensaoFibraSuperiorCQP = -(npinfinito/area + (npinfinito *
ep)/wCSup + mCQP/wCSup);


		this.tensaoFibraInferiorCQP = -(npinfinito/area + (npinfinito * ep)/wCInf
+ mCQP/wCInf);




		//Verificação pagina 72


		if( ! (tensaoFibraInferiorCF <= fctkf)){

			System.out.println("A fibra foi fissurada: Cálculo manual de
fissuras");

		//	JOptionPane.showMessageDialog(null, "A fibra foi fissurada",
"Erro", JOptionPane.ERROR_MESSAGE);

//			throw new Exception("");

		}


		else if( !(tensaoFibraSuperiorCF <= (Math.abs(0.6 *
concreto.getFck())*1000) )){

			System.out.println("A fibra foi fissurada: Cálculo manual de
fissuras");

			//JOptionPane.showMessageDialog(null, "A fibra foi fissurada",
"Erro", JOptionPane.ERROR_MESSAGE);

//			throw new Exception("");
```

```
                }



                else if(! (tensaoFibraInferiorCQP <= 0)){

                        System.out.println("A fibra foi fissurada: Cálculo manual de
fissuras");

                        //JOptionPane.showMessageDialog(null, "A fibra foi fissurada",
"Erro", JOptionPane.ERROR_MESSAGE);

//                      throw new Exception("");

                }



                else if( !(tensaoFibraSuperiorCQP <= Math.abs(0.6 *
concreto.getFck()))){

                        System.out.println("A fibra foi fissurada: Cálculo manual de
fissuras");

                        //JOptionPane.showMessageDialog(null, "A fibra foi fissurada",
"Erro", JOptionPane.ERROR_MESSAGE);

//                      throw new Exception("");

                }

                else{

                //Se passar todos os testes de fissura (quatro condições acima)

                        JOptionPane.showMessageDialog(null, "Como não houve
fissuração, o estádio 1 foi confirmado", "Sucesso", JOptionPane.DEFAULT_OPTION);

//                      throw new Exception("");

                }
```

```
//Verificação página 79



//INICIO  DIA 23/10

this.tensaoFibraSupProt = ((acoArmaduraAtiva.getPondPretracao() *(-acoArmaduraAtiva.getNp0())) / area) - ((acoArmaduraAtiva.getPondPretracao()*(-acoArmaduraAtiva.getNp0())*ep)/wCSup);

this.tensaoFibraInfProt = ((acoArmaduraAtiva.getPondPretracao() *(-acoArmaduraAtiva.getNp0())) / area) + ((acoArmaduraAtiva.getPondPretracao()*(-acoArmaduraAtiva.getNp0())*ep)/wCInf);

this.tensaoFibraSupPP = -(mG1 / wCSup);

this.tensaoFibraInfPP = mG1 / wCInf;

//resistencia do concreto ao J dias

this.t = 14; //UTILIZANDO 14 DIAS

//UTILIZANDO S=0.38

this.beta1 = 0.85436;

this.fckJ = beta1 * concreto.getFck();

//verificacões ELU simplificada (formulas 116 e 117) realizer o IF



//4 passo

//while((n * (tensaoFibraSupProt)+(tensaoFibraSupPP)) >= 1.2 * 0.3 * Math.pow(fckJ, 0.66666))
```

```
                this.nSup = (1.2 * ((0.3 * Math.pow(fckJ, 0.66666))*1000)-
tensaoFibraSupPP)/tensaoFibraSupProt;

                this.nInf = (-((0.7*(fckJ*1000))+tensaoFibraInfPP))/tensaoFibraInfProt;


                //if (nSup + nInf > acoArmaduraAtiva.getQuantidadeCordoalhas()){

                        //JOptionPane.showMessageDialog(null, "Quantidade máxima de
cordoalhas excedida", "Erro", JOptionPane.ERROR_MESSAGE);

//                      throw new Exception("");




                //5 Passo

                this.tensaoResultanteSup = quantidadeCordoalhas2 * tensaoFibraSupProt
+ tensaoFibraSupPP;

                this.tensaoResultanteInf = quantidadeCordoalhas2 * tensaoFibraInfProt +
tensaoFibraInfPP;


                //if( tensaoResultanteSup < 0 || tensaoResultanteInf < 0){






                //6 passo

//              this.h1 = null;

//              this.h2 = null;

//              this.frT = null;

//              this.asT = frT/250;


                //INICIO DOS CALCULOS DE PERDAS
```

```
		//PERDA INICIAL - RELAXACAO

		this.tensaoProtensao = (acoArmaduraAtiva.getNp0() / acoArmaduraAtiva.getArea())/1000; //MPA

		this.relaxacaoMilHoras = tensaoProtensao / acoArmaduraAtiva.getFptk();


		this.relaxacaoMilHoras2 = Double.parseDouble(JOptionPane.showInputDialog("Valor de Relaxacao 1000 : "+ relaxacaoMilHoras + "\nDigite o Valor de relaxacaoMilHoras2: "));


		//this.relaxacaoInterpolacao = relaxacaoMilHoras;

		//VERIFICAR -----

//		if(!(t < 41.66)){

			this.relaxacaoInterpolacao = relaxacaoMilHoras2 * Math.pow(((t-0)/41.67), 0.15);

//		}

		this.relaxacaoPerdas = relaxacaoInterpolacao * tensaoProtensao; // kpa

		this.perdaProtensaoRelaxacao = (relaxacaoPerdas * acoArmaduraAtiva.getArea())*10;// KN/por cordoalha

		//fim da relaxacao

		//Inicio retracao

		this.umidade = umidade;// ENTRAR COM ESSE DADO DE ACORDO COM TABELA

		this.gamaFicticio = 1+Math.pow(2.718281828, (-7.8+(0.1*umidade)));

		this.perimetroExternoAtmosfera = altura+altura+base+l;

		this.hFicticio = gamaFicticio * ((2*area)/perimetroExternoAtmosfera);

		this.temperaturaMedia = 35;

		this.gamaEndurecimento = gamaEndurecimento;// ENTRAR COM ESSE DADO DE ACORDO COM TABELA

		this.idadeFicticiaConcreto = gamaEndurecimento * ((temperaturaMedia+10)/30) * 4;
```

```
this.epsilon1S = ((-8.09)+(umidade/15)+(Math.pow(umidade, 2)/2284)+(Math.pow(umidade, 3)/133765)+(Math.pow(umidade, 4)/7608150))/Math.pow(10, 4);

this.epsilon2S = ((33+(2*hFicticio))/(20.8+(3*hFicticio)));



this.beta1Infinito = Double.parseDouble(JOptionPane.showInputDialog("Valor de Hfic: "+ hFicticio + "\nDigite o Valor de beta1Infinito: "));// ENTRAR COM ESSE DADO DE ACORDO COM TABELA

this.beta1 = 1.0;

this.epsilonCS = epsilon1S * epsilon2S * (1-beta1Infinito);

//this.epsilonCS = -(0.00054); //retirar

this.tensaoRetracaoInicial = epsilonCS * acoArmaduraAtiva.getElasticidadeacoativo(); //KPA

this.forcaRetracaoInicial = tensaoRetracaoInicial * acoArmaduraAtiva.getArea(); //KN por cordoalha

this.forcaFinal1 = acoArmaduraAtiva.getNp0() + forcaRetracaoInicial - perdaProtensaoRelaxacao;

this.porcentagemForçaFinal1 = 100-((forcaFinal1*100)/acoArmaduraAtiva.getNp0());



//INICIO DA HOMOGENIZAÇÃO DA SEÇÃO PAG97

this.gamaHomo = 15;

this.areaHomogenizada = area + acoArmaduraAtiva.getArea()*(gamaHomo - 1);

this.yCLinhaInf = ((area - acoArmaduraAtiva.getArea()*yCInf)+(acoArmaduraAtiva.getArea()*gamaHomo*y0))/areaHomogenizada;

this.deltaY = yCInf - yCLinhaInf;

this.inerciaLinhaC = inerciaX + (areaHomogenizada*Math.pow(deltaY, 2))+(acoArmaduraAtiva.getArea()*(gamaHomo - 1)*(Math.pow(yCInf - y0, 2)));
```

```
//FIM DA HOMOGENIZÇÃO


//INICIO DAS PERDAS IMEDIATAS

//ENCURTAMENTO IMEDIATO DO CONCRETO

this.eLinhaP = yCLinhaInf - y0;

this.gamaE = 1;

this.eCI = (gamaE * 5600 * Math.pow(concreto.getFck(), (0.5)))*1000;

//checar parenteses

this.encurtamentoImediato =
(acoArmaduraAtiva.getElasticidadeacoativo()/eCI)*((forcaFinal1/areaHomogenizada)+
(forcaFinal1*Math.pow(eLinhaP, 2)/inerciaLinhaC)+((mG1*Math.pow(l,
2))/8*(eLinhaP/inerciaLinhaC)));

//perda da força pelo encurtamento imediato

this.deltaPP = encurtamentoImediato * acoArmaduraAtiva.getArea();

//valor da força final de protensão considerando as perdas imediatas e que
todos os valores estão em módulo:

this.forcaFinal2 = forcaFinal1 - deltaPP;

this.porcentagemForçaFinal2 = 100-
((forcaFinal2*100)/acoArmaduraAtiva.getNp0());


//INICIO PERDAS PROGRESSIVAS PG90


//para t = 21 dias


this.beta1Infinito1 =
Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de Beta s(t0) :
"));// ENTRAR COM ESSE DADO DE ACORDO COM TABELA

this.epsilonCS2 = epsilon1S * epsilon2S * (1-beta1Infinito1);
```

```
//para t = 21 dias

this.fcT0 = Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de fcT0: ")); //receber da tabela

this.fcTInfinito = Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de fcT infinito : ")); //receber da tabela

//this.beta1Infinito2 = Double.parseDouble(JOptionPane.showInputDialog("Valor de hFicticio: "+ hFicticio + "\nDigite o Valor de beta1Infinito: "));

this.fluenciaRapida = 0.8*(1-(fcT0/fcTInfinito)); //phiA

this.phi1C = 4.45 - (0.035 * (umidade*100));

this.phi2C = (42+hFicticio)/(20+hFicticio);

this.phiInfinito = phi1C * phi2C;


this.betaFInfinito = 1.0;

this.betaF0 = Double.parseDouble(JOptionPane.showInputDialog("Valor de idade Ficticia Concreto: " + idadeFicticiaConcreto + "\n  phi Infinito: " + phiInfinito + "\nValor de Betaf(t0): ")); //receber da tabela

this.betaD = 1;

this.coeficienteFluencia = (fluenciaRapida +(phiInfinito*(1-betaF0))+0.4)/10;//verificar unidade (/10)


//para t = 90 dias

this.fcT02 = Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de fcT0: ")); //receber da tabela

this.fcTInfinito2 = Double.parseDouble(JOptionPane.showInputDialog("Digite o Valor de fcT Inifinito: ")); //receber da tabela


//this.epsilonCS3 = epsilon1S * epsilon2S * (1-beta1Infinito3);
```

```
this.fluenciaRapida2 = 0.8*(1-(fcT02/fcTInfinito2)); //phiA

this.phi1C2 = 4.45 - (0.035 * (umidade*100));

this.phi2C2 = (42+hFicticio)/(20+hFicticio);

this.phiInfinito2 = phi1C2 * phi2C2;

//this.beta1Infinito3 =
Double.parseDouble(JOptionPane.showInputDialog("Valor de hFicticio: "+ hFicticio +
"\nDigite o Valor de beta1Infinito: "));


this.betaF02 =
Double.parseDouble(JOptionPane.showInputDialog("Valor de idade Ficticia Concreto:
" + idadeFicticiaConcreto + "\n  PHI Infinito: " + phiInfinito2 + "\nValor de Beta f(t0):
")); //receber da tabela

this.coeficienteFluencia2 = (fluenciaRapida +(phiInfinito*(1-
betaF02))+0.4)/10;//verificar unidade (/10)




//5passo

this.mG1Y = (mG1*ep)/inerciaX; //KPA

this.mGiG1Y = (mGiG1*ep)/inerciaX; //KPA

this.tensaoNormalProtensao = (-(forcaFinal2 / area) +
(((forcaFinal2*ep)/inerciaX)*ep)); //KPA

this.tensaoProntensaoAco = forcaFinal2/acoArmaduraAtiva.getArea();
//KPA

this.alphaP = acoArmaduraAtiva.getElasticidadeacoativo() / eCI;

this.perdaRetracaoFluencia =
((((epsilonCS*acoArmaduraAtiva.getElasticidadeacoativo())+(alphaP*coeficienteFluen
cia*mG1Y))+(mGiG1Y*coeficienteFluencia))/(1-
(alphaP*(tensaoNormalProtensao/tensaoProntensaoAco)*(1+(coeficienteFluencia/2))))/
10000);

this.valorPerdaRetracaoFluencia = perdaRetracaoFluencia *
acoArmaduraAtiva.getArea();
```

```
this.tensaoP0 = forcaFinal2/acoArmaduraAtiva.getArea(); //KPA

this.deltaTensaoP0 = mGiG1*(ep/inerciaX)*alphaP; //KPA

this.tensaoPI = tensaoP0 + deltaTensaoP0; //KPA

this.coeficienteFinalRelaxacaoPura = (tensaoPI /
acoArmaduraAtiva.getFptk())/1000; //verificar unidade (/1000)

//popup

this.coeficienteFinalRelaxacaoPura2 =
Double.parseDouble(JOptionPane.showInputDialog("Valor de Coeficiente final de
Relaxação Pura  : " + coeficienteFinalRelaxacaoPura + "\nValor do Coeficiente final 2:
"));

this.relaxacaoPura =
(coeficienteFinalRelaxacaoPura2*(tensaoPI/100000));

this.relaxacaoRelativa = (relaxacaoPura*((1-
(2*perdaRetracaoFluencia))/(tensaoPI/10000)));

this.perdaProtensaoRelaxacaoAco =
(relaxacaoRelativa*1000)*acoArmaduraAtiva.getArea();

this.forcaFinal3 = (forcaFinal2 + perdaRetracaoFluencia -
perdaProtensaoRelaxacaoAco);

this.porcentagemForçaFinal3 = 100-
((forcaFinal3*100)/acoArmaduraAtiva.getNp0());

//INICIO DAS VERIFICACOES PARA PERDAS

// ELS


this.npinfinito = areaAcoAtivoFinal *
acoArmaduraAtiva.getPreAlongamento() *
acoArmaduraAtiva.getElasticidadeacoativo();



this.ep = (altura/2) - y0;
```

```
this.mCQP = mGI;

this.mCF = mGI;


this.tensaoFibraSuperiorCF = forcaFinal3/area + (forcaFinal3 *
ep)/wCSup + mCF/wCSup;


this.tensaoFibraInferiorCF = forcaFinal3/area + (forcaFinal3 * ep)/wCInf
+ mCF/wCInf;


this.tensaoFibraSuperiorCQP = forcaFinal3/area + (forcaFinal3 *
ep)/wCSup + mCQP/wCSup;


this.tensaoFibraInferiorCQP = forcaFinal3/area + (forcaFinal3 *
ep)/wCInf + mCQP/wCInf;


//ELU

this.preAlongamentoPerdas = (forcaFinal3
/(acoArmaduraAtiva.getElasticidadeacoativo() * acoArmaduraAtiva.getArea()))  *
1000;


//tensão aço armadura ativa

this.tensaoPDPerdas = acoArmaduraAtiva.getFpyk()*(1-
(forcaFinal3/100)) / gamaS;


//Calculo das forças de tracao

//FORÇA de Tração na armadura ativa

this.forcaTracaoAtivo2 =
(areaAcoAtivoFinal*Math.pow(10, -4)) * tensaoPDPerdas;


//Força de tração total NTD
```

```
                this.forcaTracaoTotal2 = forcaTracaoAtivo2;

                //PG 67 Tensao no concreto
                this.tensaoCD2 = 0.85 * (concreto.getFck()/1.4);
                //PG68 Aréa comprimida (Acc)
                this.areaComprimida2 = forcaTracaoTotal2/tensaoCD2;

                //pg68 altura diagrama compressao
                this.yAlturaDiagramaCompressao2 =
areaComprimida2/base;

                //pg68 Posicao linha Neuta X OBS: Criar o lambaconcreto
                this.posicaoLinhaNeutra2 =
yAlturaDiagramaCompressao2/0.8;


                //PG69 INICIO DEFORMAÇÕES ARMADURAS
ATIVAS
                this.deltaEpsilonPD2 = ((dP - posicaoLinhaNeutra2
)/posicaoLinhaNeutra2)*epsilonCU;


                //PG69 EpsilonPD
                this.defEpsilonPD2 = preAlongamentoPerdas +
deltaEpsilonPD2;

                //pg69 EpsilonPyD
```

```
                this.epsilonPyD2 =
tensaoPDPerdas/acoArmaduraAtiva.getElasticidadeacoativo();


                //COM OS RESULTADOS ACIMA, E FEITO A
VERIFICACAO DOS ITENS C e D da verficiacao ELU dominio3


                //pg69 Posição do CG da área comprimida de altura y com
tensão uniforme do concreto

                this.ylinha2 = yAlturaDiagramaCompressao2 / 2;


                //pg70


                this.zp2 = dP - ylinha2;

                //pg70 Momento resistente de cálculo

                this.MRD1 = (forcaTracaoAtivo2 * zp2);

                if(!(MRD1 >= mSD)){

                        JOptionPane.showMessageDialog(null, "Momento
resistente de cálculo não atende as solicitações", "Erro",
JOptionPane.ERROR_MESSAGE);

//                      throw new Exception("");

                }


                //FIM DA VERIFICAÇÃO DO ESTADO LIMITE
ÚLTIMO


        }
```

```
private Double base;

private Double altura;

private Double area;


//Largura da seção

private Double l;


//Distância do centro da seção da viga até a borda inferior;

private Double yCInf;


private Double yCsup;


//O usuário informa qual hipótese do cálculo. 1 para primeira condição e 2 para a segunda condição

private Integer hipotese;


//Inercia da seção em relação ao eixo X

private Double inerciaX;


//Modulo resistente inferior

private Double wCInf;


private Double wCSup;


//Raio resitente relativo à fibra infeior
```

```
private Double raioCInf;

private Double raioCSup;

private Concreto concreto;

//Momento causado pelo peso próprio
private Double mG1;
//cargas acidentais
private Double mPermanentes;
private Double mAcidentais;
private Double mGI;
private Double mGiG1;

private AcoArmaduraAtiva acoArmaduraAtiva;

private AcoArmaduraPassiva acoArmaduraPassiva;


//Dimensionamento no ELU usando processo prático k6
private Double k6;

//Momento solicitante de calculo
private Double mSD;

//posição mínima do centro de gravidade da armadura ativa
private Double y0;
```

```
//posição mínima do centro de gravidade da armadura passiva
private Double y0s;

//Altura úlil relativa ao centro de gravidade da armadura ativa
private Double dP;
//Quantidade Cordoalhas
private Double quantidadeCordoalhas2;
//Quantidade de cordoalhas * area de aco
private Double areaAcoAtivoFinal;


//Altura úlil relativa ao centro de gravidade da armadura ativa
private Double dS;



//Inicio interpolação
//Posição da linha neutra
private Double betaX;

//Variação do braço de alavanca
private Double betaZ;

//Deformação máxima do concreto na borda comprimida
private Double epsilonCD;
```

```
//Deformação da armadura ativa em conjunto com o concreto
private Double deltaEpsilonPD;


//Quinto resultado interpolação
private Double tensaoAcoAtivo;

//Quinto resultado interpolação
private Double tensaoAcoPassivo;

//Fim Interpolação

private Integer dominio;

//Cálculo da deformação total da armadura ativa
private Double epsilonPD;


//Área total do aço de armadura Ativa
private Double areaAcoAtivoMinima;

//Área total do aço de armadura Ativa
private Double areaAcoArmaduraPassiva;


private Double forcaTracao;
```

```
//Quantidade necessária de aço
private Double quantidadeAco;

//Posição da linha neutra em relação a borda comprimida
private Double x;

private Double epsilonSD;

//pg69 o calculo
private Double epsilonPyD;

//pg69 deformação simplificada A.Passivo
private Double epsilonSYD;


//Tensão aço ativo 2º calculo
private Double tensaoAcoPd;


//Tensão aço ativo 2º calculo
private Double tensaoAcoSd;

//Coeficiente resistencia do aço (Tabela 12)
private Double gamaS;

//Força tração aço ativo
private Double forcaTracaoAtivo;
```

```
private Double forcaTracaoAtivo2;
//Força tração aço passivo
private Double forcaTracaoPassivo;

//Força tração total
private Double forcaTracaoTotal;
private Double forcaTracaoTotal2;
//Equilibrio Tracao = Compressao pag67
private Double forcaCompressaoComprimida;
private Double forcaCompressaoComprimida2;

//pg 67 tensao no concreto
private Double tensaoCD;
private Double tensaoCD2;
//pg 67 Area comprimida Acc
private Double areaComprimida;
private Double areaComprimida2;
//pg67 altura do diagrama compressao
private Double yAlturaDiagramaCompressao;
private Double yAlturaDiagramaCompressao2;

//pg67 Posicao linha Neuta X
private Double posicaoLinhaNeutra;
private Double posicaoLinhaNeutra2;
//pg68 Lambda concreto
private Double lambdaConcreto;
```

```
//pg69 Deformação armadura ativa DELTAPD
private Double deltaEpsilonPD2;

//pg69 Fazer EPSILONCU que está em verificações
private Double epsilonCU;

//pg69 Deformação A.Ativa EpsilonPD
private Double defEpsilonPD;
private Double defEpsilonPD2;

//pg69 Deformação A.Passiva
private Double defepsilonSD;
private Double defepsilonSD2;
private Double epsilonPyD2;

//pg69 Posição do CG da área comprimida de altura y com tensão uniforme do concreto
private Double ylinha;
private Double ylinha2;

//pg70 Braços de alavanca da armadura ativa (zp) e da armadura passiva (zs):
private Double zp;
private Double zp2;
private Double zs;

//pg70 Cálculo do momento resistente de cálculo MRd
private Double MRD;
```

```
private Double MRD1;



//PG 72 INICIO PROTENSAO LIMITADA
//Resistencia à tração direta média do concreto
private Double fctm;

//Resistencia à tração na flexão do concreto
private Double fctkf;



//Força protenção para um tempo infinito
private Double npinfinito;

//Tensão fibra superior CQP
private Double tensaoFibraSuperiorCQP;

//Tensão fibra inferior CQP
private Double tensaoFibraInferiorCQP;

//Tensão fibra superior
private Double tensaoFibraSuperiorCF;

//Tensão fibra inferior
private Double tensaoFibraInferiorCF;
```

```
//Valor da excentricidade
private Double ep;

//Verificar calculo - (MOMENTO)
private Double mCQP;

//Verificar calculo - (MOMENTO)
private Double mCF;

//INICIO JV DIA 23/10
//VERIFICACAO PARA ELU NO ATO DA PROTENSÃO PG 79

//1º Passo - Efeito da protensão para um cabo, ou seja, uma cordoalha ou um fio:
//Tensão na fibra superior e inferior causada pela protensão
private Double nSup;
private Double nInf;
private Double tensaoFibraSupProt;
private Double tensaoFibraInfProt;
//2º Passo - Efeito do momento (Mg1) provocado pelo peso próprio da viga:
private Double tensaoFibraSupPP;
private Double tensaoFibraInfPP;
```

//3º Passo – Resistência do concreto aos j dias, ou seja, em j dias foi aplicada a protensão:

private Double fckJ;

private Double beta1;

//PARA CALCULO DE BETA 1

private Integer t;

//4º Passo

//5º Passo

private Double tensaoResultanteSup;

private Double tensaoResultanteInf;

//6º Passo – Cálculo da armadura passiva para ser distribuída na região tracionada:

private Double h1;

private Double h2;

//FT - Força resultante das tensões de tração pg83

private Double frT;

//AST – armadura passiva distribuída na região tracionada pg83

private Double asT;

// INICIO DO CALCULO DE PERDAS DE PROTENSAO, NO CASO DE PRE TRACAO Pag 83

//Perdas Iniciais

//Relaxação inicial da armadura

//Cálculo da tensão provocada pela protensão

```
        private Double tensaoProtensao;
        //1000 pg83
        private Double relaxacaoMilHoras;
        private Double relaxacaoMilHoras2;
        //resultado interpolacao
        private Double relaxacaoInterpolacao;
        private Double relaxacaoPerdas;
        private Double relaxacaoInicial;
        private Double perdaProtensaoRelaxacao;

        //Retracao inicial do concreto PG 84
        //1º Passo - Cálculo da espessura fictícia da viga (hfic):
        private Double hFicticio;
        private Double gamaFicticio;
        private Double perimetroExternoAtmosfera;
        //2º Passo - Idade fictícia do concreto (t):
        private Double idadeFicticiaConcreto;
        private Integer temperaturaMedia;
        private Double gamaEndurecimento;
        //3º Passo - Cálculo do coeficiente (1s) que depende de U e da consistência do
concreto:
        private Double epsilon1S;
        private Double umidade;
        //4º Passo - Cálculo do coeficiente (2s) que depende de hfic em cm:
        private Double epsilon2S;
        //5º Passo - Encontrar coeficiente s(t0) relativo à retração no instante inicial na
tabela
        private Double betaS;
```

```
        //7º Passo – O valor da deformação por retração pode ser expresso abaixo:
        private Double epsilonCS;
        private Double epsilonCS2;
        private Double epsilonCS3;
        private Double beta1Infinito;
        private Double beta1Infinito1;
        private Double beta1Infinito2;
        private Double beta1Infinito3;
        //8º Passo – Cálculo da perda de tensão por retração inicial do concreto (ps):
        private Double tensaoRetracaoInicial;
    //9º Passo – Perda da força por retração inicial do concreto:
        private Double forcaRetracaoInicial;
        //Força protensao final (APENAS FORÇAS INICIAIS)
        private Double forcaFinal1;

        //HOMOGENIZAÇÃO DA SEÇÃO PROTENDIDA PAG 97
        //Área de concreto (A'c):
        private Double areaHomogenizada;
        private Integer gamaHomo;
        private Double yCLinhaInf;
        private Double deltaY;
        private Double inerciaLinhaC;

        //INICIO DAS PERDAS IMEDIATAS
        //ENCURTAMENTO IMEDIATO DO CONCRETO PG 89
        private Double eLinhaP;
        //módulo de elasticidade do concreto (módulo tangente inicial)
```

```
        private Double eCI;
        private Integer gamaE;
        //perda da força pelo encurtamento imediato
        private Double deltaPP;
        //perda de tensão por encurtamento imediato
        private Double encurtamentoImediato;
        //valor da força final de protensão considerando as perdas imediatas e que todos os valores estão em módulo
        private Double forcaFinal2;

        //PERDAS PROGRESSIVAS pg 90

        private Double fcT0; // receber da tabela
        private Double fcTInfinito; // receber da tabela
        private Double fluenciaRapida;
        private Double phi1C;
        private Double phi2C;
        //Cálculo do valor final do coeficiente de deformação lenta irreversível
        private Double phiInfinito;
        private Double betaF0;//receber da tabela
        private Double betaFInfinito;
        private Integer betaD;
        //Cálculo do coeficiente de fluência:
        private Double coeficienteFluencia; //formula 168

        private Double fcT02; // receber da tabela
```

```
        private Double fcTInfinito2; // receber da tabela
        private Double fluenciaRapida2;
        private Double phi1C2;
        private Double phi2C2;
        //Cálculo do valor final do coeficiente de deformação lenta irreversível
        private Double phiInfinito2;
        private Double betaF02;//receber da tabela
        private Double betaFInfinito2;
        //Cálculo do coeficiente de fluência:
        private Double coeficienteFluencia2; //formula 168


        private Double mG1Y;
        private Double mGiG1Y;
        private Double tensaoNormalMomento;
        private Double tensaoNormalProtensao;
        private Double tensaoProntensaoAco;
        //Perda de tensão por retração e fluência:
        private Double perdaRetracaoFluencia;
        private Double alphaP;
        //6º Passo – Valor da perda de protensão devido à fluência e retração do
concreto:
        private Double valorPerdaRetracaoFluencia;
        private Double tensaoP0;
        private Double deltaTensaoP0;
        private Double tensaoPI;
        private Double coeficienteFinalRelaxacaoPura;
```

```
private Double relaxacaoPura;
private Double relaxacaoRelativa;
private Double perdaProtensaoRelaxacaoAco;
private Double forcaFinal3;
private Double perdasFinais;

//Novas variaveis para verificacoes de perdas
private Double preAlongamentoPerdas;
private Double tensaoPDPerdas;
private Double coeficienteFinalRelaxacaoPura2;

private Double porcentagemForçaFinal1;
private Double porcentagemForçaFinal2;
private Double porcentagemForçaFinal3;


public Double getBase() {
	return base;
}

public void setBase(Double base) {
	this.base = base;
}

public Double getAltura() {
	return altura;
}
```

```
public void setAltura(Double altura) {
	this.altura = altura;
}

public Double getArea() {
	return area;
}

public void setArea(Double area) {

	this.area = area;
}

public Double getyCInf() {
	return yCInf;
}

public void setyCInf(Double yCInf) {
	this.yCInf = yCInf;
}

public Double getyCsup() {
	return yCsup;
}

public void setyCsup(Double yCsup) {
```

```
        this.yCsup = yCsup;
}

public Double getInerciaX() {
        return inerciaX;
}

public void setInerciaX(Double inerciaX) {
        this.inerciaX = inerciaX;
}

public Double getwCInf() {
        return wCInf;
}

public void setwCInf(Double wCInf) {
        this.wCInf = wCInf;
}

public Double getwCSup() {
        return wCSup;
}

public void setwCSup(Double wCSup) {
        this.wCSup = wCSup;
}
```

```
public Double getRaioCInf() {
	return raioCInf;
}

public void setRaioCInf(Double raioCInf) {
	this.raioCInf = raioCInf;
}

public Double getRaioCSup() {
	return raioCSup;
}

public void setRaioCSup(Double raioCSup) {
	this.raioCSup = raioCSup;
}

public AcoArmaduraAtiva getAcoArmaduraAtiva() {
	return acoArmaduraAtiva;
}

public void setAcoArmaduraAtiva(AcoArmaduraAtiva acoArmaduraAtiva) {
	this.acoArmaduraAtiva = acoArmaduraAtiva;
}

public AcoArmaduraPassiva getAcoArmaduraPassiva() {
	return acoArmaduraPassiva;
}
```

```
	public void setAcoArmaduraPassiva(AcoArmaduraPassiva
acoArmaduraPassiva) {
		this.acoArmaduraPassiva = acoArmaduraPassiva;
	}

	public Concreto getConcreto() {
		return concreto;
	}

	public void setConcreto(Concreto concreto) {
		this.concreto = concreto;
	}

	public Double getmG1() {
		return mG1;
	}

	public void setmG1(Double mG1) {
		this.mG1 = mG1;
	}

	public Double getL() {
		return l;
	}

	public void setL(Double l) {
```

```
        this.l = l;
}

public Double getK6() {
        return k6;
}

public void setK6(Double k6) {
        this.k6 = k6;
}

public Double getmSD() {
        return mSD;
}

public void setmSD(Double mSD) {
        this.mSD = mSD;
}

public Double getY0() {
        return y0;
}

public void setY0(Double y0) {
        this.y0 = y0;
}
```

```
public Double getdP() {
	return dP;
}

public void setdP(Double dP) {
	this.dP = dP;
}

public Double getBetaX() {
	return betaX;
}

public void setBetaX(Double betaX) {
	this.betaX = betaX;
}

public Double getBetaZ() {
	return betaZ;
}

public void setBetaZ(Double betaZ) {
	this.betaZ = betaZ;
}

public Double getEpsilonCD() {
	return epsilonCD;
}
```

```
public void setEpsilonCD(Double epsilonCD) {
	this.epsilonCD = epsilonCD;
}

public Double getDeltaEpsilonPD() {
	return deltaEpsilonPD;
}

public void setDeltaEpsilonPD(Double deltaEpsilonPD) {
	this.deltaEpsilonPD = deltaEpsilonPD;
}

public Double getEpsilonPD() {
	return epsilonPD;
}

public void setEpsilonPD(Double epsilonPD) {
	this.epsilonPD = epsilonPD;
}

public Integer getDominio() {
	return dominio;
}

public void setDominio(Integer dominio) {
	this.dominio = dominio;
```

```
}

public Double getAreaAcoAtivoMinima() {
	return areaAcoAtivoMinima;
}

public void setAreaAcoAtivoMinima(Double areaAcoAtivoMinima) {
	this.areaAcoAtivoMinima = areaAcoAtivoMinima;
}

public Double getY0s() {
	return y0s;
}

public void setY0s(Double y0s) {
	this.y0s = y0s;
}

public Double getdS() {
	return dS;
}

public void setdS(Double dS) {
	this.dS = dS;
}

public Double getTensaoAcoAtivo() {
```

```
        return tensaoAcoAtivo;
}

public void setTensaoAcoAtivo(Double tensaoAcoAtivo) {
        this.tensaoAcoAtivo = tensaoAcoAtivo;
}

public Double getTensaoAcoPassivo() {
        return tensaoAcoPassivo;
}

public void setTensaoAcoPassivo(Double tensaoAcoPassivo) {
        this.tensaoAcoPassivo = tensaoAcoPassivo;
}

public Double getAreaAcoArmaduraPassiva() {
        return areaAcoArmaduraPassiva;
}

public void setAreaAcoArmaduraPassiva(Double areaAcoArmaduraPassiva) {
        this.areaAcoArmaduraPassiva = areaAcoArmaduraPassiva;
}

public Double getForcaTracao() {
        return forcaTracao;
}
```

```
public void setForcaTracao(Double forcaTracao) {
	this.forcaTracao = forcaTracao;
}



public Double getQuantidadeCordoalhas2() {
	return quantidadeCordoalhas2;
}

public void setQuantidadeCordoalhas2(Double quantidadeCordoalhas2) {
	this.quantidadeCordoalhas2 = quantidadeCordoalhas2;
}

public Double getQuantidadeAco() {
	return quantidadeAco;
}

public void setQuantidadeAco(Double quantidadeAco) {
	this.quantidadeAco = quantidadeAco;
}

public Double getX() {
	return x;
}

public void setX(Double x) {
```

```
        this.x = x;
}

public Double getEpsilonSD() {
        return epsilonSD;
}

public void setEpsilonSD(Double epsilonSD) {
        this.epsilonSD = epsilonSD;
}

public Double getEpsilonPyD() {
        return epsilonPyD;
}

public void setEpsilonPyD(Double epsilonPyD) {
        this.epsilonPyD = epsilonPyD;
}

public Integer getHipotese() {
        return hipotese;
}

public void setHipotese(Integer hipotese) {
        this.hipotese = hipotese;
}
```

```
public Double getTensaoAcoPd() {
	return tensaoAcoPd;
}

public void setTensaoAcoPd(Double tensaoAcoPd) {
	this.tensaoAcoPd = tensaoAcoPd;
}

public Double getTensaoAcoSd() {
	return tensaoAcoSd;
}

public void setTensaoAcoSd(Double tensaoAcoSd) {
	this.tensaoAcoSd = tensaoAcoSd;
}

public Double getGamaS() {
	return gamaS;
}

public void setGamaS(Double gamaS) {
	this.gamaS = gamaS;
}

public Double getForcaTracaoAtivo() {
	return forcaTracaoAtivo;
}
```

```
	public void setForcaTracaoAtivo(Double forcaTracaoAtivo) {
		this.forcaTracaoAtivo = forcaTracaoAtivo;
	}

	public Double getForcaTracaoPassivo() {
		return forcaTracaoPassivo;
	}

	public void setForcaTracaoPassivo(Double forcaTracaoPassivo) {
		this.forcaTracaoPassivo = forcaTracaoPassivo;
	}

	public Double getForcaTracaoTotal() {
		return forcaTracaoTotal;
	}

	public void setForcaTracaoTotal(Double forcaTracaoTotal) {
		this.forcaTracaoTotal = forcaTracaoTotal;
	}

	public Double getForcaCompressaoComprimida() {
		return forcaCompressaoComprimida;
	}

	public void setForcaCompressaoComprimida(Double
forcaCompressaoComprimida) {
```

```
		this.forcaCompressaoComprimida = forcaCompressaoComprimida;
	}

	public Double getTensaoCD() {
		return tensaoCD;
	}

	public void setTensaoCD(Double tensaoCD) {
		this.tensaoCD = tensaoCD;
	}

	public Double getAreaComprimida() {
		return areaComprimida;
	}

	public void setAreaComprimida(Double areaComprimida) {
		this.areaComprimida = areaComprimida;
	}

	public Double getyAlturaDiagramaCompressao() {
		return yAlturaDiagramaCompressao;
	}

	public void setyAlturaDiagramaCompressao(Double
yAlturaDiagramaCompressao) {
		this.yAlturaDiagramaCompressao = yAlturaDiagramaCompressao;
	}
```

```
public Double getPosicaoLinhaNeutra() {
	return posicaoLinhaNeutra;
}

public void setPosicaoLinhaNeutra(Double posicaoLinhaNeutra) {
	this.posicaoLinhaNeutra = posicaoLinhaNeutra;
}

public Double getLambdaConcreto() {
	return lambdaConcreto;
}

public void setLambdaConcreto(Double lambdaConcreto) {
	this.lambdaConcreto = lambdaConcreto;
}

public Double getDeltaEpsilonPD2() {
	return deltaEpsilonPD2;
}

public void setDeltaEpsilonPD2(Double deltaEpsilonPD2) {
	this.deltaEpsilonPD2 = deltaEpsilonPD2;
}

public Double getEpsilonCU() {
	return epsilonCU;
```

```
}

public void setEpsilonCU(Double epsilonCU) {
	this.epsilonCU = epsilonCU;
}

public Double getDefEpsilonPD() {
	return defEpsilonPD;
}

public void setDefEpsilonPD(Double defEpsilonPD) {
	this.defEpsilonPD = defEpsilonPD;
}

public Double getEpsilonSYD() {
	return epsilonSYD;
}

public void setEpsilonSYD(Double epsilonSYD) {
	this.epsilonSYD = epsilonSYD;
}

public Double getDefepsilonSD() {
	return defepsilonSD;
}

public void setDefepsilonSD(Double defepsilonSD) {
```

```
        this.defepsilonSD = defepsilonSD;
}

public Double getYlinha() {
        return ylinha;
}

public void setYlinha(Double ylinha) {
        this.ylinha = ylinha;
}

public Double getZp() {
        return zp;
}

public void setZp(Double zp) {
        this.zp = zp;
}

public Double getZs() {
        return zs;
}

public void setZs(Double zs) {
        this.zs = zs;
}
```

```
public Double getMRD() {
	return MRD;
}

public void setMRD(Double mRD) {
	MRD = mRD;
}

public Double getMRD1() {
	return MRD1;
}

public void setMRD1(Double mRD1) {
	MRD1 = mRD1;
}

public Double getFctm() {
	return fctm;
}

public void setFctm(Double fctm) {
	this.fctm = fctm;
}

public Double getFctkf() {
	return fctkf;
}
```

```
public void setFctkf(Double fctkf) {
	this.fctkf = fctkf;
}

public Double getNpinfinito() {
	return npinfinito;
}

public void setNpinfinito(Double npinfinito) {
	this.npinfinito = npinfinito;
}

public Double getTensaoFibraSuperiorCQP() {
	return tensaoFibraSuperiorCQP;
}

public void setTensaoFibraSuperiorCQP(Double tensaoFibraSuperiorCQP) {
	this.tensaoFibraSuperiorCQP = tensaoFibraSuperiorCQP;
}

public Double getTensaoFibraInferiorCQP() {
	return tensaoFibraInferiorCQP;
}

public void setTensaoFibraInferiorCQP(Double tensaoFibraInferiorCQP) {
	this.tensaoFibraInferiorCQP = tensaoFibraInferiorCQP;
```

```
}

public Double getTensaoFibraSuperiorCF() {
	return tensaoFibraSuperiorCF;
}

public void setTensaoFibraSuperiorCF(Double tensaoFibraSuperiorCF) {
	this.tensaoFibraSuperiorCF = tensaoFibraSuperiorCF;
}

public Double getTensaoFibraInferiorCF() {
	return tensaoFibraInferiorCF;
}

public void setTensaoFibraInferiorCF(Double tensaoFibraInferiorCF) {
	this.tensaoFibraInferiorCF = tensaoFibraInferiorCF;
}

public Double getEp() {
	return ep;
}

public void setEp(Double ep) {
	this.ep = ep;
}

public Double getmCQP() {
```

```
        return mCQP;
}

public void setmCQP(Double mCQP) {
        this.mCQP = mCQP;
}

public Double getmCF() {
        return mCF;
}

public void setmCF(Double mCF) {
        this.mCF = mCF;
}

public Double gettensaoFibraSupProt() {
        return tensaoFibraSupProt;
}

public void settensaoFibraSupProt(Double tensaoFibraSupProt) {
        this.tensaoFibraSupProt = tensaoFibraSupProt;
}

public Double getTensaoFibraInfProt() {
        return tensaoFibraInfProt;
}
```

```
public void setTensaoFibraInfProt(Double tensaoFibraInfProt) {
	this.tensaoFibraInfProt = tensaoFibraInfProt;
}

public Double getTensaoFibraSupPP() {
	return tensaoFibraSupPP;
}

public void setTensaoFibraSupPP(Double tensaoFibraSupPP) {
	this.tensaoFibraSupPP = tensaoFibraSupPP;
}

public Double getTensaoFibraInfPP() {
	return tensaoFibraInfPP;
}

public void setTensaoFibraInfPP(Double tensaoFibraInfPP) {
	this.tensaoFibraInfPP = tensaoFibraInfPP;
}

public Double getFckJ() {
	return fckJ;
}

public void setFckJ(Double fckJ) {
	this.fckJ = fckJ;
}
```

```
public Double getBeta1() {
	return beta1;
}

public void setBeta1(Double beta1) {
	this.beta1 = beta1;
}

public Integer getT() {
	return t;
}

public void setT(Integer t) {
	this.t = t;
}

public Double getTensaoFibraSupProt() {
	return tensaoFibraSupProt;
}

public void setTensaoFibraSupProt(Double tensaoFibraSupProt) {
	this.tensaoFibraSupProt = tensaoFibraSupProt;
}

public Double getTensaoResultanteSup() {
	return tensaoResultanteSup;
```

```
}

public void setTensaoResultanteSup(Double tensaoResultanteSup) {
	this.tensaoResultanteSup = tensaoResultanteSup;
}

public Double getTensaoResultanteInf() {
	return tensaoResultanteInf;
}

public void setTensaoResultanteInf(Double tensaoResultanteInf) {
	this.tensaoResultanteInf = tensaoResultanteInf;
}

public Double getH1() {
	return h1;
}

public void setH1(Double h1) {
	this.h1 = h1;
}

public Double getH2() {
	return h2;
}

public void setH2(Double h2) {
```

```
        this.h2 = h2;
}

public Double getFrT() {
        return frT;
}

public void setFrT(Double frT) {
        this.frT = frT;
}

public Double getAsT() {
        return asT;
}

public void setAsT(Double asT) {
        this.asT = asT;
}

public Double getTensaoProtensao() {
        return tensaoProtensao;
}

public void setTensaoProtensao(Double tensaoProtensao) {
        this.tensaoProtensao = tensaoProtensao;
}
```

```
public Double getRelaxacaoMilHoras() {
	return relaxacaoMilHoras;
}

public void setRelaxacaoMilHoras(Double relaxacaoMilHoras) {
	this.relaxacaoMilHoras = relaxacaoMilHoras;
}


public Double getRelaxacaoMilHoras2() {
	return relaxacaoMilHoras2;
}

public void setRelaxacaoMilHoras2(Double relaxacaoMilHoras2) {
	this.relaxacaoMilHoras2 = relaxacaoMilHoras2;
}

public Double getRelaxacaoInterpolacao() {
	return relaxacaoInterpolacao;
}

public void setRelaxacaoInterpolacao(Double relaxacaoInterpolacao) {
	this.relaxacaoInterpolacao = relaxacaoInterpolacao;
}

public Double getRelaxacaoPerdas() {
	return relaxacaoPerdas;
```

```
}

public void setRelaxacaoPerdas(Double relaxacaoPerdas) {
	this.relaxacaoPerdas = relaxacaoPerdas;
}

public Double getRelaxacaoInicial() {
	return relaxacaoInicial;
}

public void setRelaxacaoInicial(Double relaxacaoInicial) {
	this.relaxacaoInicial = relaxacaoInicial;
}

public Double getPerdaProtensaoRelaxacao() {
	return perdaProtensaoRelaxacao;
}

public void setPerdaProtensaoRelaxacao(Double perdaProtensaoRelaxacao) {
	this.perdaProtensaoRelaxacao = perdaProtensaoRelaxacao;
}

public Double gethFicticio() {
	return hFicticio;
}

public void sethFicticio(Double hFicticio) {
```

```
        this.hFicticio = hFicticio;
    }

    public Double getGamaFicticio() {
        return gamaFicticio;
    }

    public void setGamaFicticio(Double gamaFicticio) {
        this.gamaFicticio = gamaFicticio;
    }

    public Double getPerimetroExternoAtmosfera() {
        return perimetroExternoAtmosfera;
    }

    public void setPerimetroExternoAtmosfera(Double perimetroExternoAtmosfera)
{
        this.perimetroExternoAtmosfera = perimetroExternoAtmosfera;
    }

    public Double getIdadeFicticiaConcreto() {
        return idadeFicticiaConcreto;
    }

    public void setIdadeFicticiaConcreto(Double idadeFicticiaConcreto) {
        this.idadeFicticiaConcreto = idadeFicticiaConcreto;
    }
```

```
public Integer getTemperaturaMedia() {
	return temperaturaMedia;
}

public void setTemperaturaMedia(Integer temperaturaMedia) {
	this.temperaturaMedia = temperaturaMedia;
}

public Double getGamaEndurecimento() {
	return gamaEndurecimento;
}

public void setGamaEndurecimento(Double gamaEndurecimento) {
	this.gamaEndurecimento = gamaEndurecimento;
}

public Double getEpsilon1S() {
	return epsilon1S;
}

public void setEpsilon1S(Double epsilon1s) {
	epsilon1S = epsilon1s;
}

public Double getUmidade() {
	return umidade;
```

```
}

public void setUmidade(Double umidade) {
	this.umidade = umidade;
}

public Double getEpsilon2S() {
	return epsilon2S;
}

public void setEpsilon2S(Double epsilon2s) {
	epsilon2S = epsilon2s;
}

public Double getBetaS() {
	return betaS;
}

public void setBetaS(Double betaS) {
	this.betaS = betaS;
}

public Double getEpsilonCS() {
	return epsilonCS;
}

public void setEpsilonCS(Double epsilonCS) {
```

```
        this.epsilonCS = epsilonCS;
}



public Double getBeta1Infinito() {
        return beta1Infinito;
}

public void setBeta1Infinito(Double beta1Infinito) {
        this.beta1Infinito = beta1Infinito;
}

public Double getTensaoRetracaoInicial() {
        return tensaoRetracaoInicial;
}

public void setTensaoRetracaoInicial(Double tensaoRetracaoInicial) {
        this.tensaoRetracaoInicial = tensaoRetracaoInicial;
}

public Double getForcaRetracaoInicial() {
        return forcaRetracaoInicial;
}

public void setForcaRetracaoInicial(Double forcaRetracaoInicial) {
        this.forcaRetracaoInicial = forcaRetracaoInicial;
```

```
}

public Double getForcaFinal1() {
	return forcaFinal1;
}

public void setForcaFinal1(Double forcaFinal1) {
	this.forcaFinal1 = forcaFinal1;
}

public Double getAreaHomogenizada() {
	return areaHomogenizada;
}

public void setAreaHomogenizada(Double areaHomogenizada) {
	this.areaHomogenizada = areaHomogenizada;
}

public Integer getGamaHomo() {
	return gamaHomo;
}

public void setGamaHomo(Integer gamaHomo) {
	this.gamaHomo = gamaHomo;
}

public Double getyCLinhaInf() {
```

```
        return yCLinhaInf;
}

public void setyCLinhaInf(Double yCLinhaInf) {
        this.yCLinhaInf = yCLinhaInf;
}

public Double getDeltaY() {
        return deltaY;
}

public void setDeltaY(Double deltaY) {
        this.deltaY = deltaY;
}

public Double getInerciaLinhaC() {
        return inerciaLinhaC;
}

public void setInerciaLinhaC(Double inerciaLinhaC) {
        this.inerciaLinhaC = inerciaLinhaC;
}

public Double geteLinhaP() {
        return eLinhaP;
}
```

```
public void seteLinhaP(Double eLinhaP) {
	this.eLinhaP = eLinhaP;
}

public Double geteCI() {
	return eCI;
}

public void seteCI(Double eCI) {
	this.eCI = eCI;
}

public Integer getGamaE() {
	return gamaE;
}

public void setGamaE(Integer gamaE) {
	this.gamaE = gamaE;
}

public Double getDeltaPP() {
	return deltaPP;
}

public void setDeltaPP(Double deltaPP) {
	this.deltaPP = deltaPP;
}
```

```
public Double getEncurtamentoImediato() {
	return encurtamentoImediato;
}

public void setEncurtamentoImediato(Double encurtamentoImediato) {
	this.encurtamentoImediato = encurtamentoImediato;
}

public Double getForcaFinal2() {
	return forcaFinal2;
}

public void setForcaFinal2(Double forcaFinal2) {
	this.forcaFinal2 = forcaFinal2;
}

public Double getFcT0() {
	return fcT0;
}

public void setFcT0(Double fcT0) {
	this.fcT0 = fcT0;
}

public Double getFcTInfinito() {
	return fcTInfinito;
```

```
}

public void setFcTInfinito(Double fcTInfinito) {
	this.fcTInfinito = fcTInfinito;
}

public Double getFluenciaRapida() {
	return fluenciaRapida;
}

public void setFluenciaRapida(Double fluenciaRapida) {
	this.fluenciaRapida = fluenciaRapida;
}

public Double getPhi1C() {
	return phi1C;
}

public void setPhi1C(Double phi1c) {
	phi1C = phi1c;
}

public Double getPhi2C() {
	return phi2C;
}

public void setPhi2C(Double phi2c) {
```

```
        phi2C = phi2c;
}

public Double getPhiInfinito() {
        return phiInfinito;
}

public void setPhiInfinito(Double phiInfinito) {
        this.phiInfinito = phiInfinito;
}

public Double getBetaF0() {
        return betaF0;
}

public void setBetaF0(Double betaF0) {
        this.betaF0 = betaF0;
}

public Double getBetaFInfinito() {
        return betaFInfinito;
}

public void setBetaFInfinito(Double betaFInfinito) {
        this.betaFInfinito = betaFInfinito;
}
```

```
public Integer getBetaD() {
	return betaD;
}

public void setBetaD(Integer betaD) {
	this.betaD = betaD;
}


public Double getCoeficienteFluencia() {
	return coeficienteFluencia;
}

public void setCoeficienteFluencia(Double coeficienteFluencia) {
	this.coeficienteFluencia = coeficienteFluencia;
}


public Double getTensaoNormalMomento() {
	return tensaoNormalMomento;
}

public void setTensaoNormalMomento(Double tensaoNormalMomento) {
	this.tensaoNormalMomento = tensaoNormalMomento;
}

public Double getTensaoNormalProtensao() {
```

```
        return tensaoNormalProtensao;
}

public void setTensaoNormalProtensao(Double tensaoNormalProtensao) {
        this.tensaoNormalProtensao = tensaoNormalProtensao;
}

public Double getTensaoProntensaoAco() {
        return tensaoProntensaoAco;
}

public void setTensaoProntensaoAco(Double tensaoProntensaoAco) {
        this.tensaoProntensaoAco = tensaoProntensaoAco;
}

public Double getPerdaRetracaoFluencia() {
        return perdaRetracaoFluencia;
}

public void setPerdaRetracaoFluencia(Double perdaRetracaoFluencia) {
        this.perdaRetracaoFluencia = perdaRetracaoFluencia;
}

public Double getAlphaP() {
        return alphaP;
}
```

```
public void setAlphaP(Double alphaP) {
	this.alphaP = alphaP;
}

public Double getmPermanentes() {
	return mPermanentes;
}

public void setmPermanentes(Double mPermanentes) {
	this.mPermanentes = mPermanentes;
}

public Double getmAcidentais() {
	return mAcidentais;
}

public void setmAcidentais(Double mAcidentais) {
	this.mAcidentais = mAcidentais;
}

public Double getmGI() {
	return mGI;
}

public void setmGI(Double mGI) {
	this.mGI = mGI;
}
```

```
public Double getmGiG1() {
	return mGiG1;
}

public void setmGiG1(Double mGiG1) {
	this.mGiG1 = mGiG1;
}

public Double getmG1Y() {
	return mG1Y;
}

public void setmG1Y(Double mG1Y) {
	this.mG1Y = mG1Y;
}

public Double getmGiG1Y() {
	return mGiG1Y;
}

public void setmGiG1Y(Double mGiG1Y) {
	this.mGiG1Y = mGiG1Y;
}

public Double getValorPerdaRetracaoFluencia() {
	return valorPerdaRetracaoFluencia;
```

```
	}

	public void setValorPerdaRetracaoFluencia(Double
valorPerdaRetracaoFluencia) {
		this.valorPerdaRetracaoFluencia = valorPerdaRetracaoFluencia;
	}

	public Double getTensaoP0() {
		return tensaoP0;
	}

	public void setTensaoP0(Double tensaoP0) {
		this.tensaoP0 = tensaoP0;
	}

	public Double getDeltaTensaoP0() {
		return deltaTensaoP0;
	}

	public void setDeltaTensaoP0(Double deltaTensaoP0) {
		this.deltaTensaoP0 = deltaTensaoP0;
	}

	public Double getTensaoPI() {
		return tensaoPI;
	}
```

```
	public void setTensaoPI(Double tensaoPI) {
		this.tensaoPI = tensaoPI;
	}

	public Double getRelaxacaoPura() {
		return relaxacaoPura;
	}

	public void setRelaxacaoPura(Double relaxacaoPura) {
		this.relaxacaoPura = relaxacaoPura;
	}

	public Double getRelaxacaoRelativa() {
		return relaxacaoRelativa;
	}

	public void setRelaxacaoRelativa(Double relaxacaoRelativa) {
		this.relaxacaoRelativa = relaxacaoRelativa;
	}

	public Double getPerdaProtensaoRelaxacaoAco() {
		return perdaProtensaoRelaxacaoAco;
	}

	public void setPerdaProtensaoRelaxacaoAco(Double
perdaProtensaoRelaxacaoAco) {
		this.perdaProtensaoRelaxacaoAco = perdaProtensaoRelaxacaoAco;
```

```
	}

	public Double getForcaFinal3() {
		return forcaFinal3;
	}

	public void setForcaFinal3(Double forcaFinal3) {
		this.forcaFinal3 = forcaFinal3;
	}

	public Double getPerdasFinais() {
		return perdasFinais;
	}

	public void setPerdasFinais(Double perdasFinais) {
		this.perdasFinais = perdasFinais;
	}

	public Double getCoeficienteFinalRelaxacaoPura() {
		return coeficienteFinalRelaxacaoPura;
	}

	public void setCoeficienteFinalRelaxacaoPura(Double
coeficienteFinalRelaxacaoPura) {
		this.coeficienteFinalRelaxacaoPura = coeficienteFinalRelaxacaoPura;
	}
```

```
public Double getAreaAcoAtivoFinal() {
	return areaAcoAtivoFinal;
}

public void setAreaAcoAtivoFinal(Double areaAcoAtivoFinal) {
	this.areaAcoAtivoFinal = areaAcoAtivoFinal;
}

public Double getPreAlongamentoPerdas() {
	return preAlongamentoPerdas;
}

public void setPreAlongamentoPerdas(Double preAlongamentoPerdas) {
	this.preAlongamentoPerdas = preAlongamentoPerdas;
}

public Double getTensaoPDPerdas() {
	return tensaoPDPerdas;
}

public Double getPorcentagemForçaFinal1() {
	return porcentagemForçaFinal1;
}

public void setPorcentagemForçaFinal1(Double porcentagemForçaFinal1) {
	this.porcentagemForçaFinal1 = porcentagemForçaFinal1;
}
```

```
public Double getPorcentagemForçaFinal2() {
	return porcentagemForçaFinal2;
}

public void setPorcentagemForçaFinal2(Double porcentagemForçaFinal2) {
	this.porcentagemForçaFinal2 = porcentagemForçaFinal2;
}

public Double getPorcentagemForçaFinal3() {
	return porcentagemForçaFinal3;
}

public void setPorcentagemForçaFinal3(Double porcentagemForçaFinal3) {
	this.porcentagemForçaFinal3 = porcentagemForçaFinal3;
}

public void setTensaoPDPerdas(Double tensaoPDPerdas) {
	this.tensaoPDPerdas = tensaoPDPerdas;
}



public Double getnSup() {
	return nSup;
}
```

```
	public void setnSup(Double nSup) {
		this.nSup = nSup;
	}

	public Double getnInf() {
		return nInf;
	}

	public void setnInf(Double nInf) {
		this.nInf = nInf;
	}

	public Double getCoeficienteFinalRelaxacaoPura2() {
		return coeficienteFinalRelaxacaoPura2;
	}

	public void setCoeficienteFinalRelaxacaoPura2(Double
coeficienteFinalRelaxacaoPura2) {
		this.coeficienteFinalRelaxacaoPura2 = coeficienteFinalRelaxacaoPura2;
	}

	public Double getFcT02() {
		return fcT02;
	}

	public void setFcT02(Double fcT02) {
		this.fcT02 = fcT02;
```

```
}

public Double getFcTInfinito2() {
	return fcTInfinito2;
}

public void setFcTInfinito2(Double fcTInfinito2) {
	this.fcTInfinito2 = fcTInfinito2;
}

public Double getFluenciaRapida2() {
	return fluenciaRapida2;
}

public void setFluenciaRapida2(Double fluenciaRapida2) {
	this.fluenciaRapida2 = fluenciaRapida2;
}

public Double getPhi1C2() {
	return phi1C2;
}

public void setPhi1C2(Double phi1c2) {
	phi1C2 = phi1c2;
}

public Double getPhi2C2() {
```

```
        return phi2C2;
}

public void setPhi2C2(Double phi2c2) {
        phi2C2 = phi2c2;
}

public Double getPhiInfinito2() {
        return phiInfinito2;
}

public void setPhiInfinito2(Double phiInfinito2) {
        this.phiInfinito2 = phiInfinito2;
}

public Double getBetaF02() {
        return betaF02;
}

public void setBetaF02(Double betaF02) {
        this.betaF02 = betaF02;
}

public Double getBetaFInfinito2() {
        return betaFInfinito2;
}
```

```
public void setBetaFInfinito2(Double betaFInfinito2) {
	this.betaFInfinito2 = betaFInfinito2;
}

public Double getCoeficienteFluencia2() {
	return coeficienteFluencia2;
}

public void setCoeficienteFluencia2(Double coeficienteFluencia2) {
	this.coeficienteFluencia2 = coeficienteFluencia2;
}

public Double getBeta1Infinito2() {
	return beta1Infinito2;
}

public void setBeta1Infinito2(Double beta1Infinito2) {
	this.beta1Infinito2 = beta1Infinito2;
}

public Double getBeta1Infinito3() {
	return beta1Infinito3;
}

public void setBeta1Infinito3(Double beta1Infinito3) {
	this.beta1Infinito3 = beta1Infinito3;
}
```

```
public Double getEpsilonCS2() {
	return epsilonCS2;
}

public void setEpsilonCS2(Double epsilonCS2) {
	this.epsilonCS2 = epsilonCS2;
}

public Double getEpsilonCS3() {
	return epsilonCS3;
}

public void setEpsilonCS3(Double epsilonCS3) {
	this.epsilonCS3 = epsilonCS3;
}

public Double getForcaTracaoAtivo2() {
	return forcaTracaoAtivo2;
}

public void setForcaTracaoAtivo2(Double forcaTracaoAtivo2) {
	this.forcaTracaoAtivo2 = forcaTracaoAtivo2;
}

public Double getForcaTracaoTotal2() {
	return forcaTracaoTotal2;
```

```
	}

	public void setForcaTracaoTotal2(Double forcaTracaoTotal2) {
		this.forcaTracaoTotal2 = forcaTracaoTotal2;
	}

	public Double getForcaCompressaoComprimida2() {
		return forcaCompressaoComprimida2;
	}

	public void setForcaCompressaoComprimida2(Double
forcaCompressaoComprimida2) {
		this.forcaCompressaoComprimida2 = forcaCompressaoComprimida2;
	}

	public Double getTensaoCD2() {
		return tensaoCD2;
	}

	public void setTensaoCD2(Double tensaoCD2) {
		this.tensaoCD2 = tensaoCD2;
	}

	public Double getAreaComprimida2() {
		return areaComprimida2;
	}
```

```
	public void setAreaComprimida2(Double areaComprimida2) {
		this.areaComprimida2 = areaComprimida2;
	}

	public Double getyAlturaDiagramaCompressao2() {
		return yAlturaDiagramaCompressao2;
	}

	public void setyAlturaDiagramaCompressao2(Double
yAlturaDiagramaCompressao2) {
		this.yAlturaDiagramaCompressao2 = yAlturaDiagramaCompressao2;
	}

	public Double getPosicaoLinhaNeutra2() {
		return posicaoLinhaNeutra2;
	}

	public void setPosicaoLinhaNeutra2(Double posicaoLinhaNeutra2) {
		this.posicaoLinhaNeutra2 = posicaoLinhaNeutra2;
	}

	public Double getDefEpsilonPD2() {
		return defEpsilonPD2;
	}

	public void setDefEpsilonPD2(Double defEpsilonPD2) {
		this.defEpsilonPD2 = defEpsilonPD2;
```

```
}

public Double getDefepsilonSD2() {
	return defepsilonSD2;
}

public void setDefepsilonSD2(Double defepsilonSD2) {
	this.defepsilonSD2 = defepsilonSD2;
}

public Double getEpsilonPyD2() {
	return epsilonPyD2;
}

public void setEpsilonPyD2(Double epsilonPyD2) {
	this.epsilonPyD2 = epsilonPyD2;
}

public Double getYlinha2() {
	return ylinha2;
}

public void setYlinha2(Double ylinha2) {
	this.ylinha2 = ylinha2;
}

public Double getZp2() {
```

```
        return zp2;
    }

    public void setZp2(Double zp2) {
        this.zp2 = zp2;
    }

    public Double getBeta1Infinito1() {
        return beta1Infinito1;
    }

    public void setBeta1Infinito1(Double beta1Infinito1) {
        this.beta1Infinito1 = beta1Infinito1;
    }


}
```