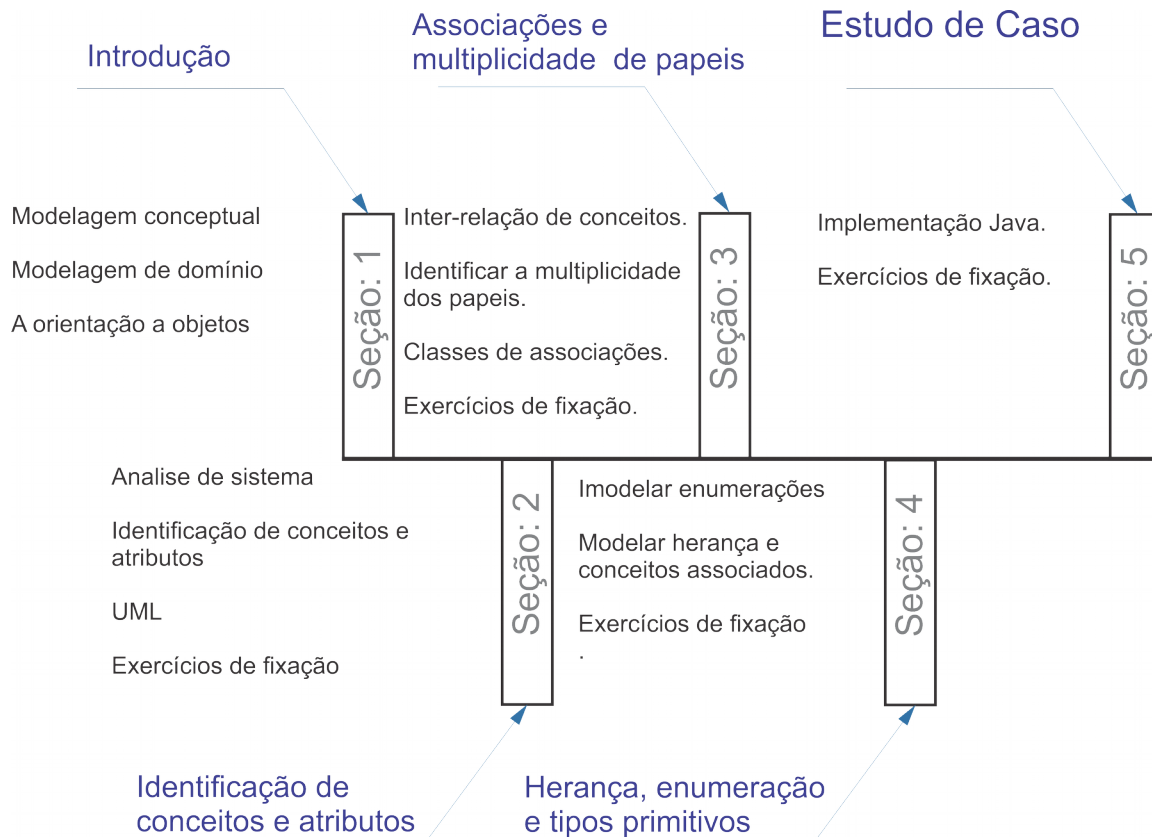


# **Técnicas de Programação 1**

**Professor Cristhian Riaño**

# Decomposição de problemas usando classes e objetos.



# Programação Orientada a objetos - Java

Nesta secção se apresentam os elementos fornecidos pela linguagem de programação Java, associados com os conceitos de programação orientada a objetos.

Os temas abordados serão:

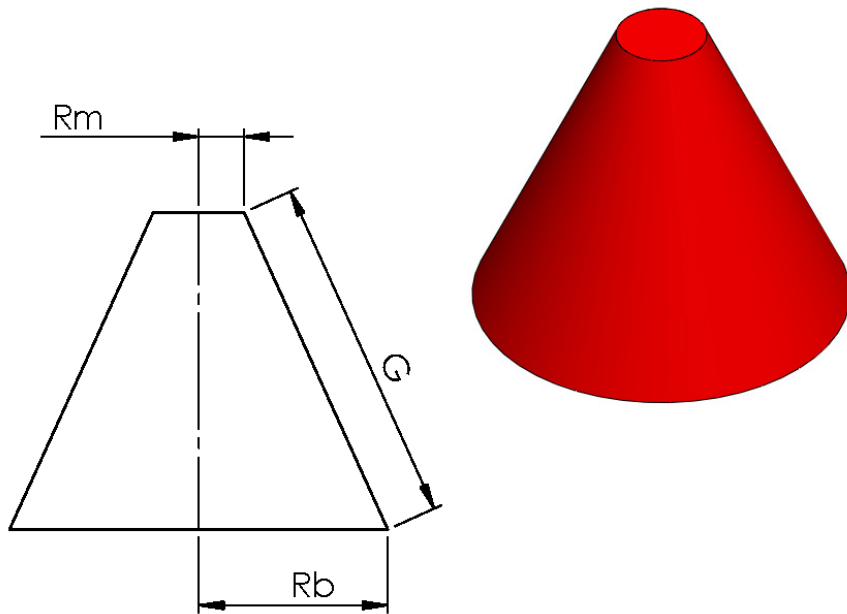
- O conceito de classe e objeto
- Acesso aos atributos e métodos de uma classe

Para a abordagem do tópico vamos a resolver um problema usando o paradigma de orientação a objetos.

# Resolvendo o problema sem orientação a objetos

O problema consiste em determinar o volume de um cone truncado concêntrico.

- Precisa ler as variáveis de raio da base ( $R_b$ ), raio mínimo ( $R_m$ ) e geratriz ( $G$ ).

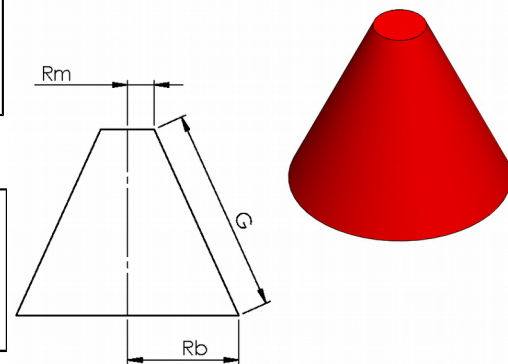


# Exemplo Cone Truncado Concêntrico

$$F_f = 1/3 \cdot \pi \cdot \sqrt{L_e^2 - (R_b - R_m)^2} \cdot (R_b^2 + R_m^2 + R_b \cdot R_m) + H \cdot \pi \cdot R^2 + L_r \cdot \pi \cdot R_b^2$$

Vc1	Raio da base (Rb)	125 mm	3540763.08 mm <sup>3</sup>
	Raio mínimo (Rm)	10 mm	
	Geratriz (G)	230 mm	

Vc2	Raio da base (Rb)	150 mm	4749074.22 mm <sup>3</sup>
	Raio mínimo (Rm)	25 mm	
	Geratriz (G)	210 mm	



```

package ExercicioVolumeCono;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class VolumeCono {

    public static void main(String[] args) throws Exception {
        int Rb,Rm,G;
        double Vc;
        System.out.println("Digitar o Valor de [Rb]: ");
        BufferedReader Rbin =new BufferedReader(new InputStreamReader(System.in));
        String S1= Rbin.readLine();
        Rb= Integer.parseInt(S1);

        System.out.println("Digitar o Valor de [Rm]: ");
        BufferedReader Rmin =new BufferedReader(new InputStreamReader(System.in));
        String S2= Rmin.readLine();
        Rm= Integer.parseInt(S2);

        System.out.println("Digitar o Valor de [G]: ");
        BufferedReader Gin =new BufferedReader(new InputStreamReader(System.in));
        String S3= Gin.readLine();
        G= Integer.parseInt(S3);

        System.out.println("Valor Rb,Rm e G : "+ Rb + " " + Rm + " "+ " "+ G);

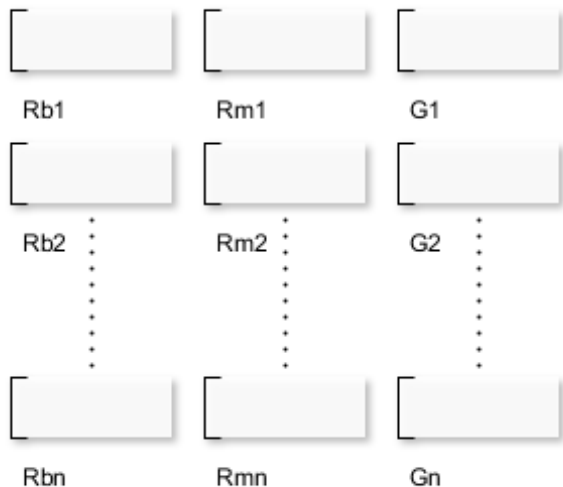
        Vc= ((Math.PI)*(Math.sqrt(Math.pow(G, 2)- Math.pow((Rb-Rm), 2)))*(Math.pow(Rb,2)+Math.pow(Rm, 2) + (Rb*Rm)))/3;
        System.out.println("Volume Vc:   "+ Vc);
    }
}

```

# Discussão

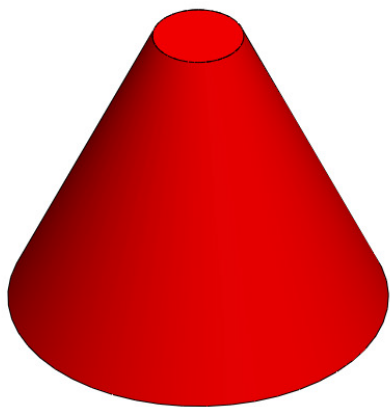
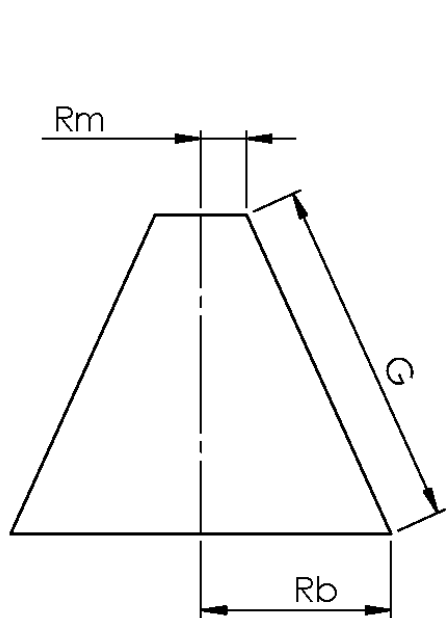
Vamos melhorar a representação do problema desde a perspectiva de orientação a objetos.

Se desejamos saber qual é o menor volume entre dois ou mais cones, vamos a precisar de valores para cada uma das três variáveis.



# Discussão

Podemos considerar o cone truncado como uma entidade com três atributos.



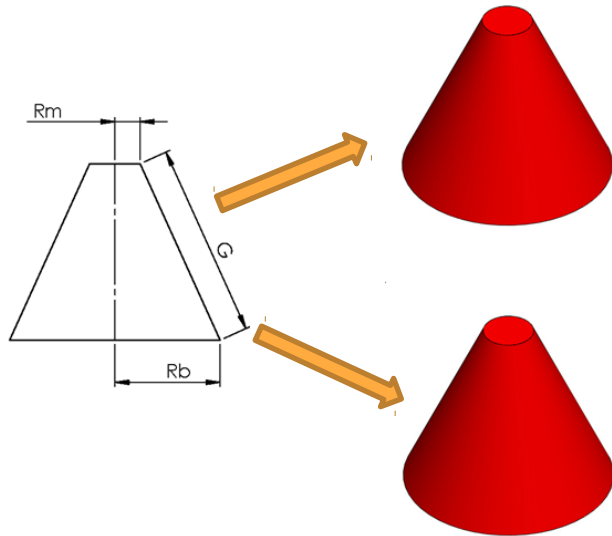
ConeTruncado	
<ul style="list-style-type: none"><li>- &lt;&lt;Oid&gt;&gt; id : int</li><li>- radioBase : int</li><li>- radioMenor : int</li><li>- generatrix : int</li></ul>	
<ul style="list-style-type: none"><li>+ mostrarDados() : String</li><li>+ CalcularVolume() : double</li></ul>	



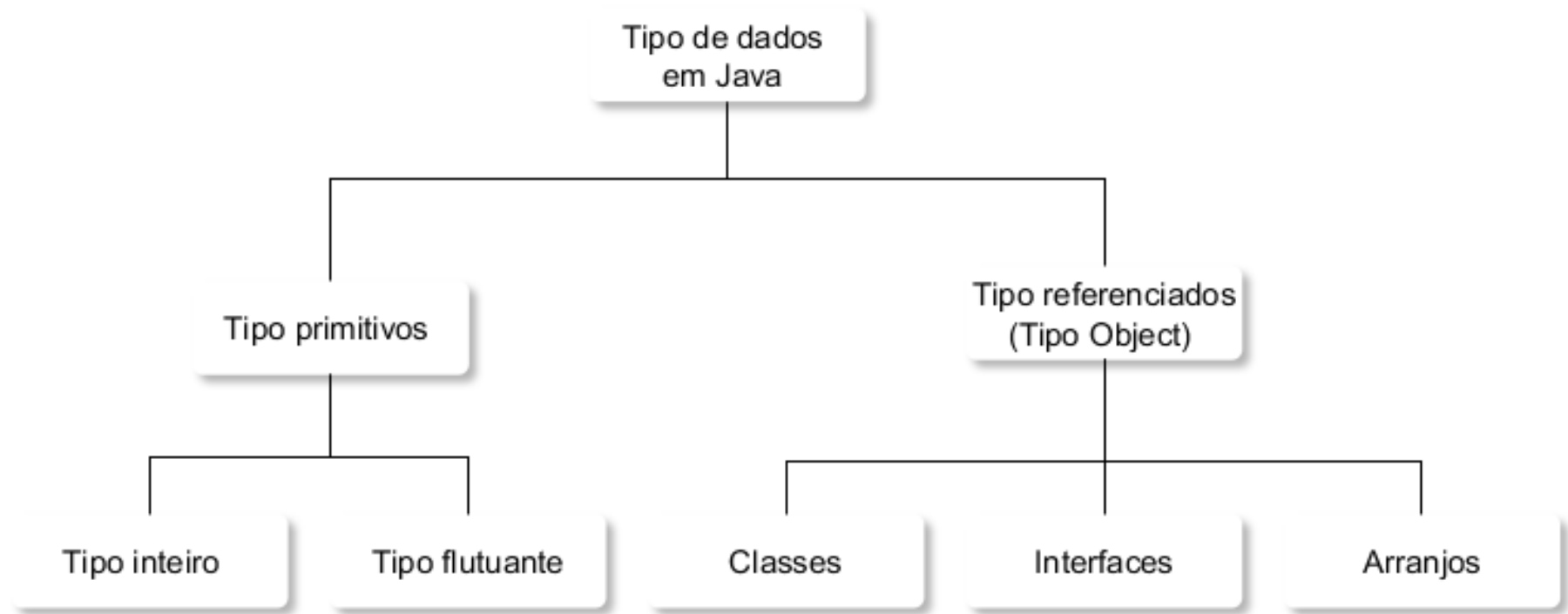
# Discussão

Vamos a criar a classe com dois métodos. Um para calcular o volume e outro para mostrar os valores na tela.

- Uma classe gera um novo tipo de dados em java (template).
- Objeto é uma instância de uma classe. (Objeto e instância tem significado equivalente)



# Tipos de dados em Java



# Tipos de dados em Java

		Valores possíveis				
Tipos	Primitivo	Menor	Maior	Valor Padrão	Tamanho	Exemplo
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;
	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d; ou double ex6 = 10.20;
Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

# Sintaxe - Classe

Para declarar uma classe em java usamos a seguinte sintaxe:

```
<modificador de acesso> class NomeDaClasse {  
    // Local onde atributos, construtores e métodos são criados  
  
}
```

Exemplo:

```
package Volume;  
public class ConeTruncado {  
  
}
```

Convenções:

- Manter o nome simples e descritivo.
- Usar palavras inteiras.
- A primeira letra de cada palavra deve estar em caixa alta.
- O arquivo gerado deve ser salvo com o mesmo nome da classe e extensão \*.java

# Modificador de Acesso

	Classe	Pacote	Subclasse	Outros
public	O	O	O	O
private	O	X	X	X
protected	O	O	O	X
Default	O	O	X	X

O

Possível

X

Não

# Modificador de Acesso

- Public: Indica que um método é acessível a traves de uma instância de um objeto.
- Private: Indica que o método não está acessível desde uma instância.
- Protected: Indica que é acessível unicamente por classes derivadas.
- Default: É possível usar desde instâncias no mesmo pacote.

# Sintaxe – Atributos

Para declarar um atributo em java usamos a seguinte sintaxe:

```
[modifVisibilidade] [modifAtributo] tipo nomeVariable [= valorInicial] ;
```

Tipo é o tipo da variável e pode ser básico, objeto de uma classe vector, matriz interface.

```
package Volume;  
public class ConeTruncado {  
    public int Rb, Rm, G; //atributos  
}
```

# Sintaxe – Métodos

Para declarar um método em java usamos a seguinte sintaxe:

```
[especificadores] tipoDevuelto nomeMetodo([lista parámetros]) [throws  
listaExcepciones]  
{  
    // instrucciones  
    [return valor;]  
}
```

- Descreva o que o método deve fazer.
- Determine as entradas do método.
- Determine os tipos de entradas.
- Determine o tipo do valor retornado.
- Escreva as instruções que formam o corpo do método.
- Método de teste: projetar diferentes casos de teste



# Sintaxe – Métodos

```
package Volume;
public class ConeTruncado {
    public int Rb, Rm, G;
    public void mostrarDados() {
        System.out.println("Valor Raio Base: "+ Rb);
        System.out.println("Valor Raio Menor: "+ Rm);
        System.out.println("Valor Generatrix: "+ G);
    }
    double CalcularVolume( ){
        return ((Math.PI)*(Math.sqrt(Math.pow(this.G, 2)- Math.pow((this.Rb-
this.Rm), 2)))*(Math.pow(this.Rb,2)+Math.pow(this.Rm, 2) + (this.Rb*this.Rm)))/3;
    }
}
```

Para conseguir executar a classe é preciso realizar uma instanciação de classe.

# Construtores em Java

Os construtores são os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida.

Os construtores podem ter identificador de acesso como: public, private ou protected.

```
public class ConoTroncado {  
    public int Rb, Rm, G;  
    //Construtor vazio  
    ConoTroncado(){}  
    //Construtor com argumentos  
    ConoTroncado(int Rb, int Rm, int G){  
        this.Rb = Rb;  
        this.Rm = Rm;  
        this.G = G;  
    }  
}
```

# Construtores em Java

- Devido a que pode em algum instante ser tedioso inicializar variáveis de uma classe, Java permite dar valores a variáveis desde o mesmo instante de sua criação. Este conceito é conhecido como construtores.
- Os construtores são executados só no momento de criação.
- Não retornam nenhum valor.
- O nome do construtor deve ser idêntico da classe (isso permite identificar se uma classe tem construtores o não).
- Java por padrão agrega um construtor vazio, mas se definimos um construtor distinto Java omite o construtor padrão.
- O construtor inicializa o objeto ao momento da criação e reserva a memória para associar os dados de seu atributos

# Escopo das variáveis em Java

Entre os tipos de variáveis em Java encontramos variáveis de classe e variáveis locais e a duração da mesma depende de onde ela foi definida.

Exemplo:

```
public class ConoTroncado {  
    //Definimos os atributos e variáveis de nossa classe.
```

```
    int Rb;
```

```
    int Rm;
```

```
    int G;
```



Variáveis de classe

```
double CalcularVolume(int Rb, int Rm, int G ){
```

```
double part0 =(Math.PI);
```

```
double part1 = (Math.sqrt(Math.pow(G, 2)- Math.pow((Rb-Rm), 2)));
```

```
double part2 = (Math.pow(Rb,2)+Math.pow(Rm, 2) + (Rb-Rm));
```

```
double Vc=(part0*part1*part2)/3;
```

```
return Vc;}
```



Variáveis locais

# Escopo das variáveis em Java

- Variáveis de Classe

- Podem se usada em qualquer método da classe.
- Inicializam-se com valores padrões

- Variáveis locais

- Podem ser usadas apenas no método que foi definida. Se devem inicializar.
- As variáveis locais ocultam as variáveis de classe.
- Para usar as variáveis de classe em um método onde foram definidas as variáveis locais com o mesmo nome que as variáveis de classe devemos usar a palavra [this](#).

# Palavra This

- Variáveis de Classe


- Podem se usada em qualquer método da classe.
- Inicializam-se com valores padrões

- Variáveis locais

- Podem ser usadas apenas no método que foi definida. Se devem inicializar.
- As variáveis locais ocultam as variáveis de classe.
- Para usar as variáveis de classe em um método onde foram definidas as variáveis locais com o mesmo nome que as variáveis de classe devemos usar a palavra [this](#).

# Palavra This

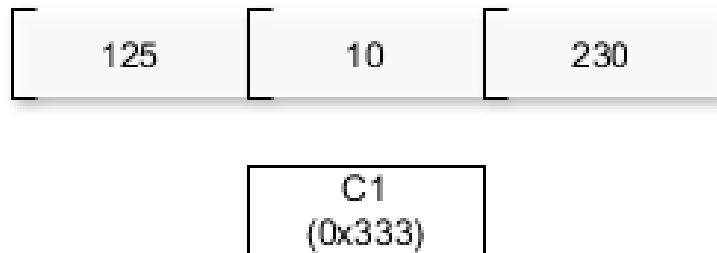
```
System.out.println("Mostrar Valor Volume 2: " + C2.CalcularVolume(150, 25, 210));
```



```
ConoTroncado(int Rb, int Rm, int G){  
    this.Rb = Rb;  
    this.Rm = Rm;  
    this.G = G;  
}
```

**this.Rb = Rb**

Atributo da classe



Argumento Recebido

# Objetos em Java

- Sabemos que o objeto é a instância de uma classe.
- Existem casos que podemos trabalhar diretamente na classe e é chamado de contexto estático.
- No contexto dinâmico precisamos de uma instância para trabalhar com a classe.

```
public class ConoTest {  
    public static void main(String[] args) throws Exception {  
        double Vc;  
        //Criar objetos  
        ConoTruncado C1;  
        C1=new ConeTruncado();  
        ConoeTruncado C2= new ConeTruncado();  
    }  
}
```



# Objetos em Java

Podemos considerar o *cone truncado* como uma entidade com três atributos.

ConoTroncado C1, C2;



Representa dois objetos cones a ser criados C1 e C2 do tipo ConoTroncado.

C1=new ConoTroncado();

C2=new ConoTroncado();



ConoTroncado C2= new ConoTroncado();



# Objetos em Java

Para acessar o valor de um atributo, usamos o nome do objeto criado e seguido pelo ponto e nome do atributo.

```
import java.io.BufferedReader;  
import java.io.InputStreamReader;
```

```
//Modificar valores de atributos
```

```
System.out.println("Digitar o Valor de [Rb]:  ");
```

```
BufferedReader Rbin=new BufferedReader(new InputStreamReader(System.in));
```

```
String S1= Rbin.readLine();
```

```
C1.Rb = Integer.parseInt(S1);
```

```
C1.Rb = 125;
```

```
C1.Rm = 10;
```

```
C1.G = 230;
```

C1	[ 125	[ 10	[ 230
C2	[ 150	[ 25	[ 210

```

package Volume;
public class ConoTroncado {
    int Rb, Rm, G;
    //Construtor vazio
    ConoTroncado(){
    }
    //Construtor com argumentos
    ConoTroncado(int Rb, int Rm, int G){
        this.Rb = Rb;
        this.Rm = Rm;
        this.G = G;
    }
    public void mostrarDados() {
        System.out.println("Valor Raio Base: "+this.Rb);
        System.out.println("Valor Raio Menor: "+this.Rm);
        System.out.println("Valor Generatrix: "+this.G);
    }

    double CalcularVolume( ){
    return ((Math.PI)*(Math.sqrt(Math.pow(this.G, 2)- Math.pow((this.Rb-this.Rm), 2)))*(Math.pow(this.Rb,2)+Math.pow(this.Rm, 2)
+ (this.Rb*this.Rm)))/3;
    }

    double CalcularVolume(int Rb, int Rm, int G ){
        double part0 =(Math.PI);
        double part1 = (Math.sqrt(Math.pow(G, 2)- Math.pow((Rb-Rm), 2)));
        double part2 = (Math.pow(Rb,2)+Math.pow(Rm, 2) + (Rb*Rm));
        double Vc=(part0*part1*part2)/3;
        return Vc;
    }
}

```

```

package Volume;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class ConoTest {
    public static void main(String[] args) throws Exception {
        double Vc;
        ConoTroncado C1;
        C1=new ConoTroncado();
        ConoTroncado C2= new ConoTroncado();
        System.out.println("Digitar o Valor de [Rb]: ");
        BufferedReader Rbin =new BufferedReader(new InputStreamReader(System.in));
        String S1= Rbin.readLine();
        C1.Rb = Integer.parseInt(S1);
        System.out.println("Digitar o Valor de [Rm]: ");
        BufferedReader Rmin =new BufferedReader(new InputStreamReader(System.in));
        String S2= Rmin.readLine();
        C1.Rm= Integer.parseInt(S2);
        System.out.println("Digitar o Valor de [G]: ");
        BufferedReader Gin =new BufferedReader(new InputStreamReader(System.in));
        String S3= Gin.readLine();
        C1.G= Integer.parseInt(S3);
        System.out.println("Mostrar Valores C1");
        C1.mostrarDados();
        Vc = C1.CalcularVolume();
        System.out.println("Mostrar Valor Volume 1: " + Vc);
        System.out.println("Mostrar Valor Volume 2: " + C2.CalcularVolume(150, 25, 210));
    }
}

```

# Discussão

Vamos a criar a classe com dois métodos. Um para calcular o volume e outro para mostrar os valores na tela.

$$F_f = 1/3 \cdot \pi \cdot \sqrt{L_e^2 - (R_b - R_m)^2} \cdot (R_b^2 + R_m^2 + R_b \cdot R_m) + H \cdot \pi \cdot R^2 + L_r \cdot \pi \cdot R_b^2$$

```
public void mostrarDados() {  
    System.out.println("Valor Raio Base: "+this.Rb);  
    System.out.println("Valor Raio Menor: "+this.Rm);  
    System.out.println("Valor Generatrix: "+this.G);  
}  
  
double CalcularVolume( ){  
    return ((Math.PI)*(Math.sqrt(Math.pow(this.G, 2)- Math.pow((this.Rb-this.Rm),2)))*(Math.pow(this.Rb,2)+Math.pow(this.Rm,2) + (this.Rb*this.Rm)))/3;  
}
```