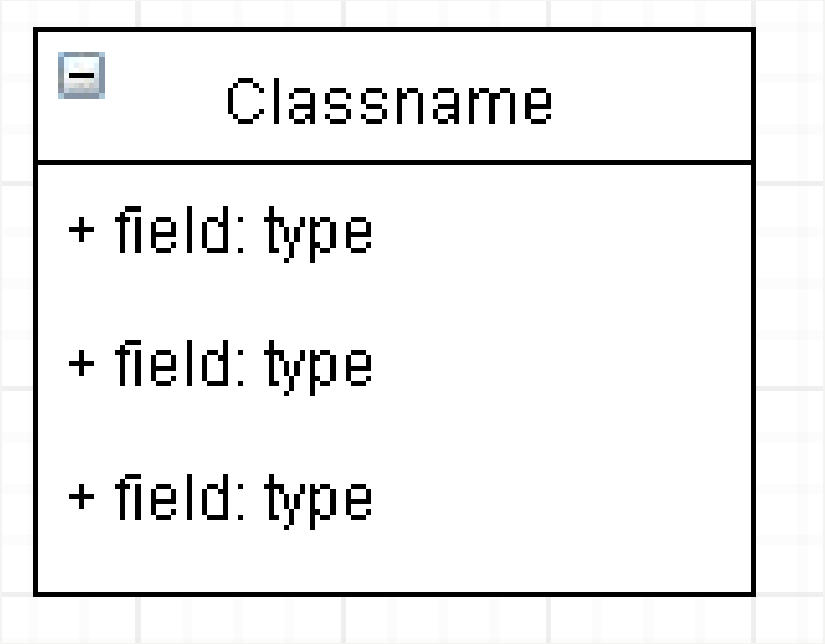
CONSTRUTORES E SOBRECARGA

PROF. ME. HÉLIO ESPERIDIÃO

NOME DE CLASSES

- Toda classe **deve** começar com uma letra maiúscula.
- Não pode conter letras não ASCII (caracteres de língua de origem latina, como caracteres acentuados)
- Nome composto por mais de uma palavra:
  - a primeira letra de cada palavra deve ser em maiúscula.



MÉTODOS

- Os *métodos* definem as ações a serem tomadas em diversos momentos da execução de um programa.
- Os métodos correspondem aos conceitos comuns de funções, procedimentos ou sub-rotinas.
- Os métodos que realizam todas as tarefas para as quais o programa foi escrito.
- Os métodos devem ser nomeados com a primeira letra minúscula e as demais palavras iniciando com letra maiúscula.

CLASSES

```
class [nome] {
  [atributos e métodos]
}
```

ESCOPO COM VARIÁVEIS

- Quando um variável é declarada no interior de uma estrutura seus valores e referências serão mantidos durante o momento em que a estrutura estiver sendo executada, depois disso essa variável não mais terá finalidade para o programa devendo ser destruída, caso o programador tente usar essa variável logo após a execução dessa estrutura o compilador irá detectar erro.

EXEMPLO DE ESCOPO

Y não existe no escopo
Y só existe dente da condição IF

```
int x=0;
if(x==0){
    x++;
    double y=33;
}
System.out.println(y);
```

O OPERADOR THIS

this faz referencia a métodos e atributos da própria classe

```
public class Pessoa {
    public int idade;
    public String nome;
    public void fazerAniversario(){
        this.idade++;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa();
        p.nome="Hélio";
        p.idade=31;
        p.fazerAniversario();
        System.out.println(p.idade);
    }

}
```

QUAL O VALOR DA IDADE EM P1 E P2?

```
public class Pessoa {
    public int idade;
    public String nome;
    public void fazerAniversario(){
        this.idade++;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa();
        p.nome="Hélio";
        p.idade=31;
        p.fazerAniversario();
        System.out.println(p.idade);


        Pessoa p2 =new Pessoa();
        p2.nome="Patricia";
        p2.idade=17;
        p2.fazerAniversario();
        System.out.println(p2.idade);
    }

}
```

CONSTRUTOR

- É um método que possui o mesmo nome da classe.
- Este método é executado antes de todos os outros quando a classe é instanciada.

```
public class Pessoa {
    public int idade;
    public String nome;
    Pessoa(){
        System.out.println("Oi. Eu sou o construtor");
    }
    public void fazerAniversario(){
        this.idade++;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa();
        p.nome="Hélio";
        p.idade=31;
        p.fazerAniversario();
        System.out.println(p.idade);



    }

}
```

CONSTRUTOR

- No exemplo abaixo a classe é instanciada, mas nenhum atributo recebe valor.
- Qual a saída desse programa?

```
public class Pessoa {
    public int idade;
    public String nome;

    public void fazerAniversario(){
        this.idade++;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa();
        System.out.println(p.nome);
        System.out.println(p.idade);



    }

}
```

EXEMPLO DE CONSTRUTOR

Método Construtor

O operador this
faz referência a própria classe ou seja ao atributo nome e não ao parâmetro nome

```
public class Pessoa {
    public int idade;
    public String nome;

    public void fazerAniversario(){
        this.idade++;
    }

    Pessoa(String nome,int idade){
        this.nome=nome;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa("Hélio",31);
        System.out.println(p.nome);
        System.out.println(p.idade);

    }

}
```

Parâmetros do método construtor.

Instância da classe pessoa, passa como parâmetros para o construtor o nome e a idade

O CONSTRUTOR OBRIGA

- Veja que existe apenas um construtor na classe e ele exige que sejam passados dois parâmetros(nome e idade). Caso os parâmetros não sejam passados é gerado erro de compilação

```
public class Pessoa {
    public int idade;
    public String nome;

    public void fazerAniversario(){
        this.idade++;
    }

   Pessoa(String nome,int idade){
        this.nome=nome;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa();
        System.out.println(p.nome);
        System.out.println(p.idade);

    }

}
```

OBJETO/CLASSE COMO ATRIBUTO

```
public class Pessoa {
    public int idade;
    public String nome;
    public Pessoa mae;
    public void fazerAniversario(){
        this.idade++;
    }

     Pessoa(String nome,int idade){
        this.nome=nome;
        this.idade=idade;
    }
    public static void main(String args[]){
        Pessoa p =new Pessoa("Ana",31);
        p.mae = new Pessoa("Mariana",65);
        p.mae.mae=new Pessoa("Geovana",90);

        System.out.println(p.nome);
        System.out.println(p.idade);

        System.out.println(p.mae.nome);
        System.out.println(p.mae.idade);

         System.out.println(p.mae.mae.nome);
    }
}
```

POLIMORFISMO

PROF.ME. HÉLIO ESPERIDIÃO

POLIMORFISMO

- Traduzindo, do grego, ao pé da letra, polimorfismo significa "muitas formas".
- Essas formas, em nosso contexto de programação, são as subclasses/objetos criados a partir de uma classe maior, mais geral, ou abstrata
- Polimorfismo é a capacidade da orientação a objeto que permite controlar todas as formas de uma maneira mais simples e geral, sem ter que se preocupar com cada objeto especificamente.

O POLIMORFISMO AD HOC

- O termo *ad hoc* tem origem no latim e significa "*para esta finalidade*".
- O termo é usado para tudo aquilo que foi projetado para atender a uma demanda pontual, não genérica.
- O polimorfismo do tipo ad-hoc é o polimorfismo que ocorre em tempo de compilação.
- Técnicas *ad hoc* serão sempre aquelas para resolver a necessidades pontuais.
- Esta categoria abrange duas técnicas
  - Coerção (também chamado de *casting*)
  - Sobrecarga.

POLIMORFISMO DE COERÇÃO (COERCION)

- Essa conversão acontece de maneira implícita, dizemos que existe polimorfismo por coerção já que você está "coergindo" ou convertendo o tipo de uma variável para outra.

- Isso só é possível porque int é 'menor' que double.
- O contrário não é possível na linguagem Java!

```
int a = 2;
double n = a;
```

UPCAST

- Converter sem perda de informação

```
int i = 20;
double x =2.5;
x = (double) i;
System.out.println(x);
```

DOWNCAST

CONVERTER COM PERCA DE INFORMAÇÃO

```
int i = 20;
double x = 2.5;
i = (int) x;
System.out.println(i);
```

SOBRECARGA

SOBRECARGA (OVERLOADING)

- Possuímos métodos com o mesmo nome
- Assinaturas são diferentes
- Esse é o conceito de polimorfismo de sobrecarga!
- É executado em tempo de compilação (Antes de executar o código você já sabe qual método será executado).
- Note os retornos dos métodos.
- Qual a saída?

```
public class Animal {
      public void comer() {
    System.out.println("Comendo...");
  }

  public void comer(String nomeComida) {
    System.out.println("Comendo " + nomeComida);
  }
  public static void main(String args[]){
      Animal a = new Animal();
      a.comer();
      a.comer("Batata");

  }
}
```

PONTO2D

- Análise o código ao lado.

- Torne este código funcional.
- Implemente conceitos de polimorfismo e encapsulamento de dados.
- Programe a classe com a maior quantidade de polimorfismo e encapsulamento possível.

```
public class Ponto2D {
    private double x;
    private double y;
    Ponto2D(){
        System.out.println("Construtor vazio");
    }
    Ponto2D(double x, double y){
        System.out.println("Construtor double");
    }
    Ponto2D(int x, int y){
        System.out.println("Construtor int");
    }
    Ponto2D(float x, float y){
        System.out.println("Construtor float");

    }
    public double CalcularDistancia(Ponto2D p1, Ponto2D p2){
        return 0;
    }
      public double CalcularDistancia(double x1, double x2,double y1, double y2){
        return 0;
    }
    public static void main(String args[]){
        Ponto2D p = new Ponto2D();
         p = new Ponto2D(1.1,1);
         p = new Ponto2D(1.2f,1);
    }
}
```

EXERCÍCIOS.

- Crie a Classe calculadora, esta classe deve ser capaz de realizar as operações básicas de matemática.

- Utilize conceitos de encapsulamento, construtores e polimorfismo.