# Classes e Métodos – Parte 2

Prof. Bruno Nogueira

# Modificadores de acesso

- Public e private
  - Public já vimos que é acessível a todos
  - Private torna o método ou atributo acessível somente aos métodos da classe em que são declarados

```
1  public class Turma{
2      private String nomeCurso; // Nome do curso
3
4      // Método para configurar o valor do nome do curso
5      public void setNomeCurso(String nome){
6          nomeCurso = nome;
7      }
8
9      // Método para retornar o valor do nome do curso
10     public String getnomeCurso(){
11         return nomeCurso;
12     }
13
14     // Método para exibição da mensagem de boas vindas ao curso
15     public void exibeMensagem(){
16         System.out.printf("Bem vindo ao curso %s!\n", nomeCurso);
17     }
18 }
```

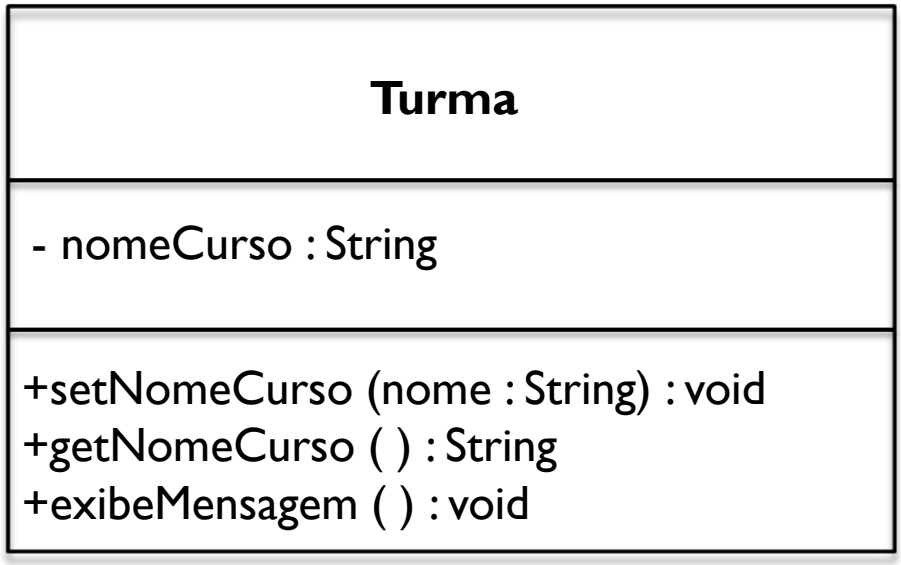
# Modificadores de acesso

- Public e private
  - Public já vimos que é acessível a todos
  - Private torna o método ou atributo acessível somente aos métodos da classe em que são declarados

```
1  public class Turma{
2      private String nomeCurso; // Nome do curso
3
4      // Método para configurar o valor do nome do cu
5      public void setNomeCurso(String nome){
6          nomeCurso = nome;
7      }
8
9      // Método para retornar o valor do nome do curso
10     public String getnomeCurso(){
11         return nomeCurso;
12     }
13
14     // Método para exibição da mensagem de boas vindas ao curso
15     public void exibeMensagem(){
16         System.out.printf("Bem vindo ao curso %s!\n", nomeCurso);
17     }
18 }
```

Acessível somente aos métodos setNomeCurso, getNomeCurso e exibeMensagem

# Modificadores de acesso




- No diagrama UML, modificadores privados são indicados pelo símbolo (-)

# Métodos get e set

- Como as variáveis privadas não são acessíveis por outros métodos de outras classes, cabe ao programador manter métodos de acesso adequados e controlados
  - Métodos get e set
    - Métodos get: retornam o valor da variável
    - Métodos set: atribuem valor à variável

```
4       // Método para configurar o valor do nome do curso
5       public void setNomeCurso(String nome){
6           nomeCurso = nome;
7       }
8
9       // Método para retornar o valor do nome do curso
10      public String getnomeCurso(){
11          return nomeCurso;
12      }
```

# Métodos construtores

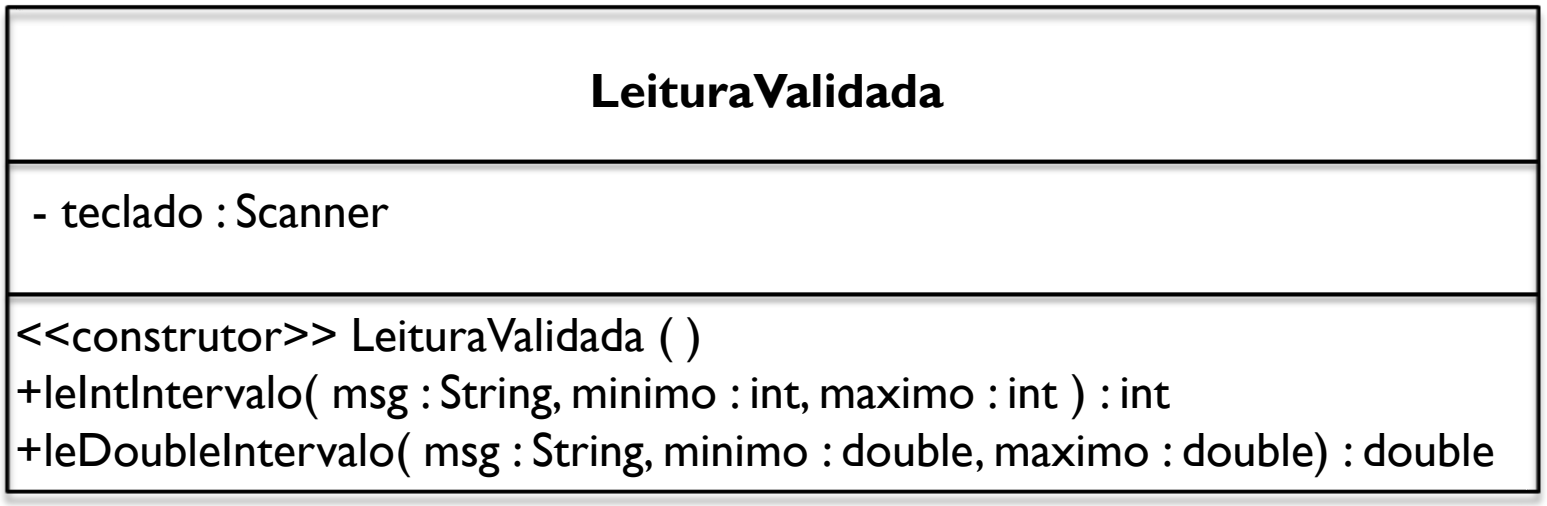
- Já discutimos antes: a combinação de **new** com o **nome da classe** seguido de parênteses chama o **construtor da classe**
  - Método especial chamado apenas quando da inicialização da classe

```
Turma novaTurma = new Turma ();
```

  - Por padrão, Java fornece um construtor padrão, sem qualquer parâmetro
    - Variáveis de instância são inicializadas para o valor padrão
    - String é inicializada com null; int com 0; double com 0.0; etc...
  - Exercício: fazer a classe TurmaTeste, chamar o método getNomeCurso antes e depois de setNomeCurso

# Exemplo...

- Vamos considerar uma classe que tenha a seguinte modelagem:



```java
import java.util.Scanner;
public class LeituraValidada{

    private Scanner teclado;

    // Construtor da classe
    public LeituraValidada()
    {
        teclado = new Scanner(System.in);
    }

    //  Lê um valor inteiro e verifica se o mesmo está em um intervalo predefinido
    //  Mensagem se refere à mensagem a ser exibida ao usuário quando da requisição da
    // entrada
    public int leIntIntervalo(String mensagem, int minimo, int maximo){

        int valor;
        System.out.println(mensagem);
        valor = teclado.nextInt();

        while(valor < minimo || valor > maximo)
        {
            System.out.printf("O valor digitado deve estar entre %d e %d." +
                              " Por favor, digite novamente: ", minimo, maximo);
            valor = teclado.nextInt();
        }

        return valor;
    }
```

# Exemplo

```
30
31      //  Lê um valor double e verifica se o mesmo está em um intervalo predefinido
32      //  Mensagem se refere à mensagem a ser exibida ao usuário quando da requisição da
33      // entrada
34      public double leDoubleIntervalo(String mensagem, double minimo, double maximo){
35
36          double valor;
37          System.out.println(mensagem);
38          valor = teclado.nextDouble();
39
40          while(valor < minimo || valor > maximo)
41          {
42              System.out.printf("O valor digitado deve estar entre %d e %d." +
43                                " Por favor, digite novamente: ", minimo, maximo);
44              valor = teclado.nextDouble();
45          }
46          return valor;
47      }
48  }
```

# Métodos construtores

- Podemos declarar nossos próprios métodos construtores

```
6       // Construtor da classe
7       public LeituraValidada()
8       {
9           teclado = new Scanner(System.in);
10      }
```

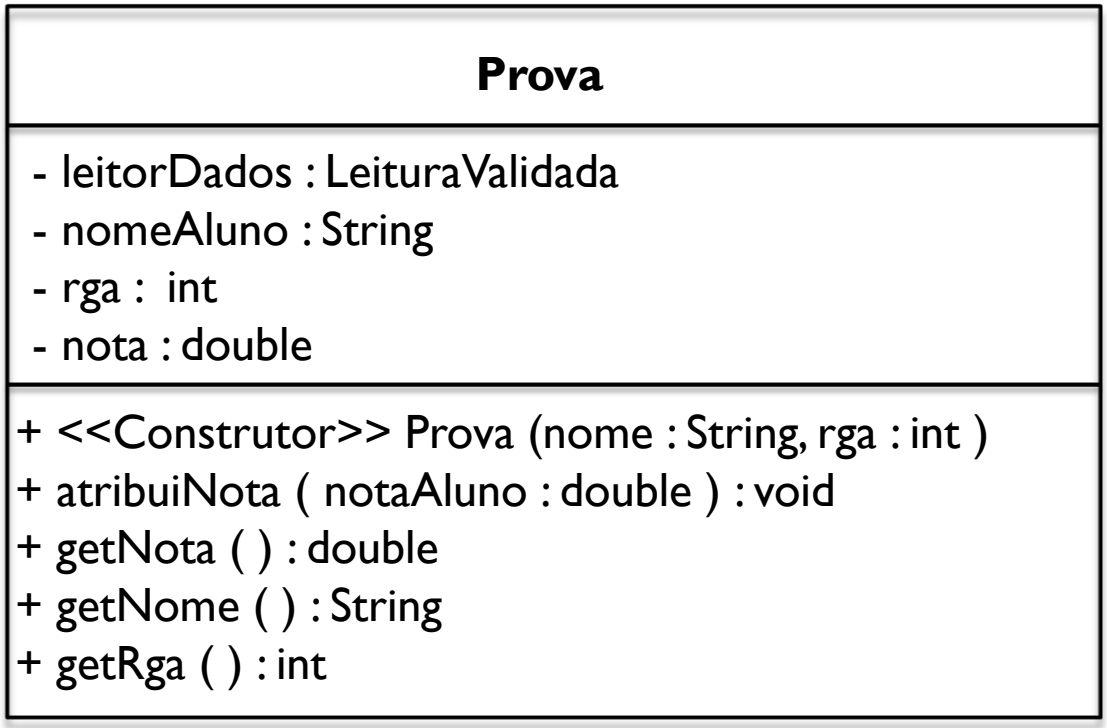
  - Sintaxe

```
public NomeClasse( LISTA_DE_PARÂMETROS ){ ... }
```

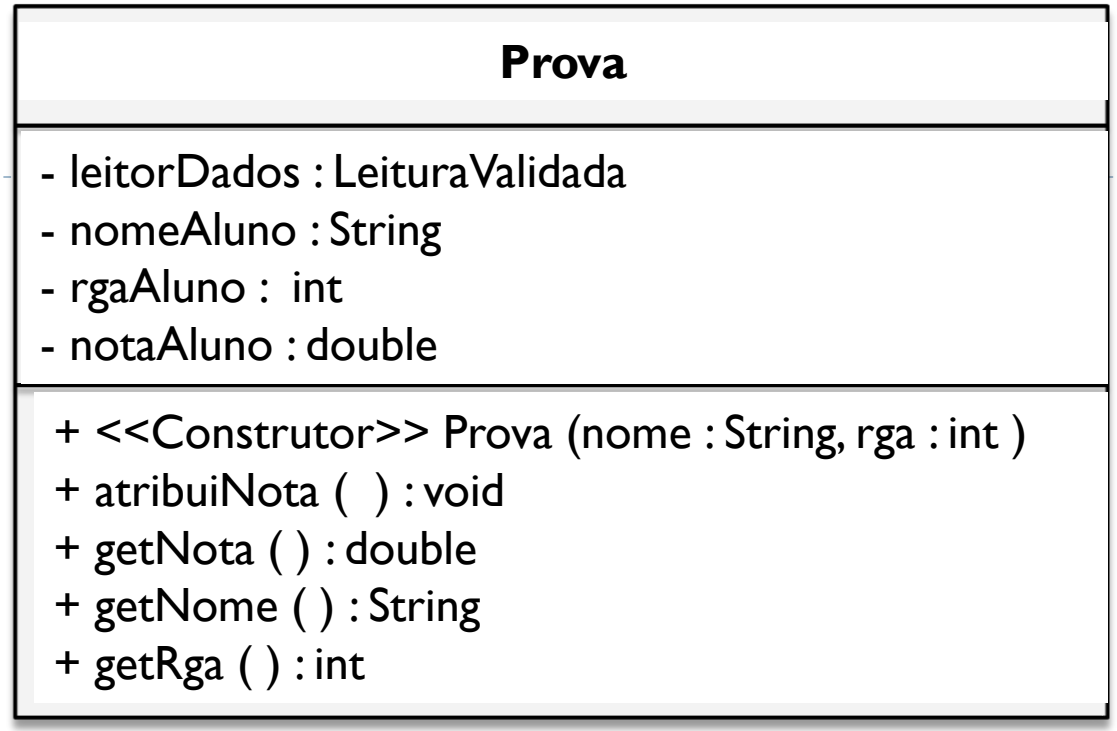
  - Dentro destes métodos, inicializamos os nossos objetos de maneira a garantir o funcionamento adequado do mesmo

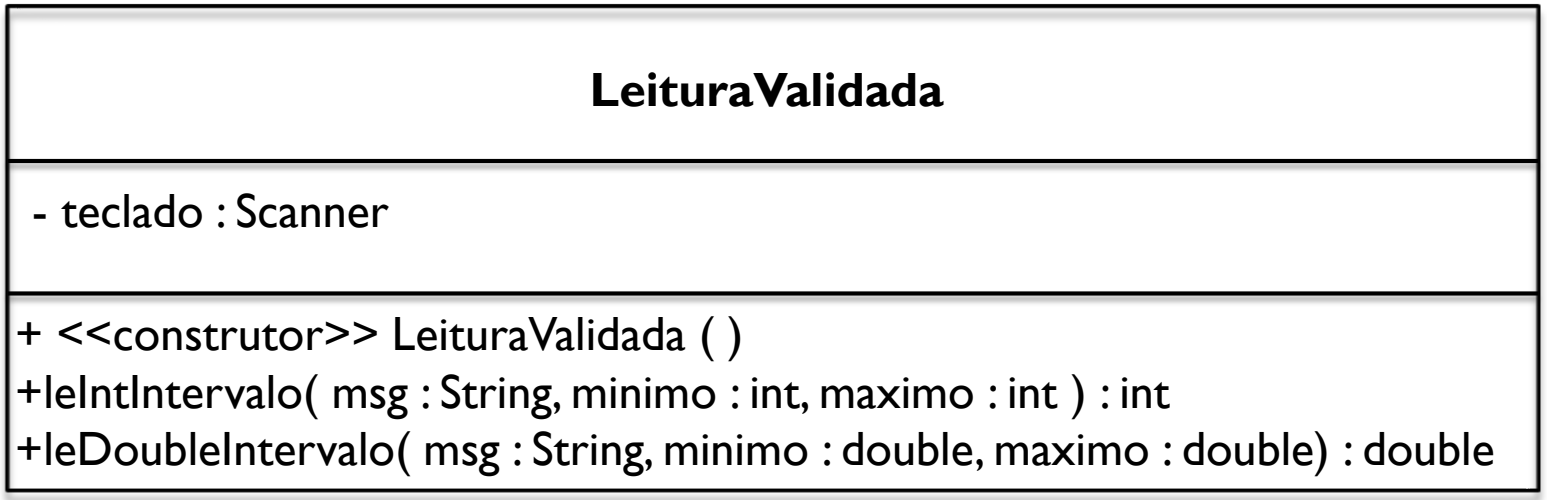
# Exercício

- Crie uma classe que atenda ao seguinte diagrama. As notas devem ser validadas utilizando a classe LeituraValidada, devendo estar no intervalo [0, 10]



# Exercício


Prova
- leitorDados : LeituraValidada
- nomeAluno : String
- rgaAluno : int
- notaAluno : double
+ <<Construtor>> Prova (nome : String, rga : int )
+ atribuiNota ( ) : void
+ getNota ( ) : double
+ getNome ( ) : String
+ getRga ( ) : int


LeituraValidada
- teclado : Scanner
+ <<construtor>> LeituraValidada ( )
+leIntIntervalo( msg : String, minimo : int, maximo : int ) : int
+leDoubleIntervalo( msg : String, minimo : double, maximo : double) : double