

PROGRAMAÇÃO MODULAR

FUNDAMENTOS DE POO: MEMBROS DE CLASSE

PROF. JOÃO CARAM

ATRIBUTOS DE CLASSE

2

- Existem atributos que dizem respeito à toda coleção de objetos, e não a um objeto específico.

Ex: Classe *Aluno* → última matrícula gerada
 Classe *Livro* → conjunto de gêneros de livros
 Classe *Inimigo* → quantidade de inimigos vivos no jogo

ATRIBUTOS DE CLASSE

3

- Atributos de classe: compartilhados por todos os objetos daquela classe.
- Escopo local (delimitado pela visibilidade declarada).
- Tempo de vida global.
- Inicializados pelo primeiro objeto ou pelo carregamento da classe.

ATRIBUTOS DE CLASSE

4

- Úteis para:
 - Implementação de contadores e identificadores de autoincremento.
 - definição de constantes de validação.
- Economia de memória:
 - variável é compartilhada pelos objetos de uma classe.
- Declarados com a palavra chave static

EXEMPLO/EXERCÍCIO

5

- Um aluno de um curso livre é representado por seu nome e número de matrícula.
- O curso é identificado pelo código.
- A matrícula é atribuída automaticamente.
- O aluno faz 4 avaliações por semestre.

CLASSE ALUNO

6

- Nome e número de matrícula.
- Código do curso
- O aluno faz 4 avaliações por semestre.
- Matrícula atribuída automaticamente

Aluno
<pre>-matricula: int -nome: String -codCurso: int -notas: double[] -faltas: int</pre>
<pre>+lançarNota(valor:double): void +notaFinal(): double +aprovado(): bool</pre>

CLASSE ALUNO


```
class Aluno{  
    private int matricula;  
    private String nome;  
    private int codCurso;  
    private double[] notas;  
    ...  
}
```

← Atribuição automática
sem repetição?

Aluno
-matricula: int -nome: String -codCurso: int -notas: double[] -faltas: int
+lançarNota(valor:double): void +notaFinal(): double +aprovado(): bool

CLASSE ALUNO

```
class Aluno{  
    private static int proxMatricula;  
    private int matricula;  
    private String nome;  
    private int codCurso;  
    private double[] notas;  
    ...  
}
```



Atributo de classe

Aluno
-proxMatricula: int
-matricula: int
-nome: String
-codCurso: int
-notas: double[]
-faltas: int
+lançarNota(valor:double): void
+notaFinal(): double
+aprovado(): bool

CLASSE ALUNO

```
class Aluno{  
    static{  
        proxMatricula = 1;  
    }  
    public Aluno(int curso, String nome){  
        this.matricula = proxMatricula;  
        proxMatricula++;  
        notas = new double[4];  
        ...  
    }  
}
```

Aluno
-proxMatricula: int
-matricula: int
-nome: String
-codCurso: int
-notas: double[]
-faltas: int
+lançarNota(valor:double): void
+notaFinal(): double
+aprovado(): bool

0 SISTEMA EVOLUI...

10

- 0 aluno faz 4 avaliações por semestre e será aprovado se a nota final for maior ou igual a 60 e tiver frequência de no mínimo 75%
- Nota para aprovação?
- Frequência mínima?
- Quantidade de avaliações?
- Quantidade de aulas?

0 SISTEMA EVOLUI...

11

- ▣ 0 aluno faz 4 avaliações por semestre e será aprovado se a nota final for maior ou igual a 60 e tiver frequência de no mínimo 75%
- ▣ Nota para aprovação?
- ▣ Frequência mínima?
- ▣ Quantidade de avaliações?
- ▣ Quantidade de aulas?



ATRIBUTOS FINAIS

12

- Não podem mudar de valor após inicializados
- Podem ser atributos convencionais (pertencem ao objeto) ou de classe (estáticos)
 - Java: *final*
 - C#: *sealed*

CLASSE ALUNO

13

- 60 pontos para aprovação
- 75% de frequência
- 4 avaliações
- Número de aulas varia

Aluno
<pre>-proxMatricula: int -<u>NOTAAPROVACAO: double {60}</u> -<u>FREQMINIMA: double {0.75}</u> -matricula: int -nome: String -codCurso: int -notas: double[] -faltas: int</pre>
<pre>+lançarNota(valor:double): void +notaFinal(): double +aprovado(): bool +frequencia(qtAulas:int): double</pre>

CLASSE ALUNO

```
class Aluno{  
    private static int proxMatricula;  
    private static final int QTDAAVALIACOES;  
    private static final double NOTAAPROVACAO;  
    private static final double FREQMINIMA;  
    private int final matricula;  
    ...  
}
```

Aluno
-proxMatricula: int
-NOTAAPROVACAO: double {60}
-FREQMINIMA: double {0.75}
-matricula: int
-nome: String
-codCurso: int
-notas: double[]
-faltas: int
+lançarNota(valor:double): void
+notaFinal(): double
+aprovado(): bool
+frequencia(qtAulas:int): double

CLASSE ALUNO

```
class Aluno{  
    static{  
        proxMatricula = 1;  
        QTDVAVALIACOES = 4;  
        NOTAAPROVACAO = 60.0;  
        FREQMINIMA = 0.75;  
    }  
    ...  
}
```

Aluno
-proxMatricula: int
-NOTAAPROVACAO: double {60}
-FREQMINIMA: double {0.75}
-matricula: int
-nome: String
-codCurso: int
-notas: double[]
-faltas: int
+lançarNota(valor:double): void
+notaFinal(): double
+aprovado(): bool
+frequencia(qtAulas:int): double

CLASSE ALUNO

```
class Aluno{  
    public Aluno(int curso, String nome){  
        this.matricula = proxMatricula;  
        proxMatricula++;  
        notas = new double[QTDAAVALIACOES];  
        ...  
    }  
}
```

Aluno
-proxMatricula: int
-NOTAAPROVACAO: double {60}
-FREQMINIMA: double {0.75}
-matricula: int
-nome: String
-codCurso: int
-notas: double[]
-faltas: int
+lançarNota(valor:double): void
+notaFinal(): double
+aprovado(): bool
+frequencia(qtAulas:int): double

ATRIBUTOS DE CLASSE

17

- Caso sejam públicos, os atributos de classe são acessados externamente pelo nome da classe.



CLASSE ALUNO

```
class App{  
    public static void main (String args[] ) {  
        Aluno aluno1 = new Aluno("42", "Ada Lovelace");  
        System.out.print("Nota para aprovação: ");  
        System.out.println(Aluno.NOTAAPROVACAO);  
    }  
}
```

MÉTODOS DE CLASSE

19

- Também podemos ter métodos que não precisam ou não devem acessar dados particulares de objetos
- Ex: Classe Aluno
 - Número da última matrícula
 - Frequência mínima

CLASSE ALUNO

```
class Aluno{  
    private static int proxMatricula;  
    private static final int QTDVALIACOES;  
    private static final double NOTAAPROVACAO;  
    private static final double FREQMINIMA;  
    private int final matricula;  
    ...  
}
```

CLASSE ALUNO

```
class Aluno{  
    ...  
    public static int frequenciaMinima(){  
        return FREQMINIMA * 100;  
    }  
  
    public static int ultimaMatricula(){  
        return proxMatricula-1;  
    }  
}
```

APP PARA CLASSE ALUNO

```
class App{  
    public static void main (String args[] ) {  
        Aluno aluno1 = new Aluno(...);  
        (...)  
        System.out.print(aluno1.getNome());  
        System.out.println(" tem " + aluno1.getFrequencia());  
        System.out.print("Frequência mínima: ");  
        System.out.println(Aluno.frequenciaMinima());  
    }  
}
```

CLASSES ESTÁTICAS

23

- Classes que contêm apenas atributos e métodos de classe podem ser chamadas *classes estáticas*
- Ex:
 - *bibliotecas* de funções → System, Math...
 - manipulação de tipos → Integer.parseInt

CLASSE ESTÁTICA: CONVERSOR

```
class Conversor{  
    public static final double JARDA_EM_CM = 91.44;  
  
    public static double CelsiusParaFahrenheit(double tempC){  
        return (tempC * (9/5) + 32);  
    }  
  
    public static double JardasParaCentimetros(double jardas){  
        return (jardas * JARDA_EM_CM);  
    }  
}
```


APP E CONVERTOR

```
class App{  
    public static void main (String args[] ) {  
        double campoNFL;  
  
        campoNFL = Convertor.JardasParaCentimetros(120.0);  
        System.out.print("Uma jarda vale ");  
        System.out.print(Convertor.JARDA_EM_CM+ " cm");  
  
    }  
}
```

CLASSES ESTÁTICAS

26

- Uma classe estática não deve ser instanciada, mas pode ter um *inicializador estático*

OBRIGADO.

DÚVIDAS?