FATEC – Votorantim
Profº Rodrigo Diver

Aula 3 – Revisão

- **UML – Linguagem de Modelagem Simplificada.**
  - *Conceitos e aplicação.*
  - *Padronização de Classes, Métodos e Atributos.*
  - *Representação dos Modificadores de Acesso.*

- Palavra reservada “this”.
  - *Ambiguidade em nome de atributos.*
  - *Recursão em construtores.*
  - *Auto referência.*

UML – Linguagem de Modelagem Unificada.

"A UML permite modelar de forma mais concisa, consistente e sem ambiguidades softwares orientados a objeto:"

- *A UML é uma combinação de várias notações orientadas a objetos: design orientado a objetos, técnica de modelagem de objetos e engenharia de software orientada a objetos.*
- *Modela os aspectos estruturais (estáticos) e comportamentais (dinâmicos) do sistema.*
- *É uma linguagem que permite descrever os atributos e métodos de uma classe, além dos relacionamentos entre classes.*
- *Oferece padrões de notação para modelar todos os tipos de sistemas de computação.*

UML – Linguagem de Modelagem Unificada.

Nome da Classe

- A primeira letra de todos os nome serão maiúsculas o restante minúscula.
- Exemplo: Pessoa, Aluno, AlunoColegio, AlunoFaculdade;

Visibilidade

(+) Publico
(-) Privado
(#) Restrito

Tipo das variáveis

- Poderá ser utilizado todas as variáveis primitivas ou compostas. Se houver restrições ou padrões a serem respeitados, deverão ser descritos dentro de {....}.

Nome das Variáveis e Métodos.

- Inicia com letra minúscula, se houver outros nomes a primeira letra dos próximos nomes serão maiúscula.
- Exemplo: nomeMae, getNomeMae, setNomePai.

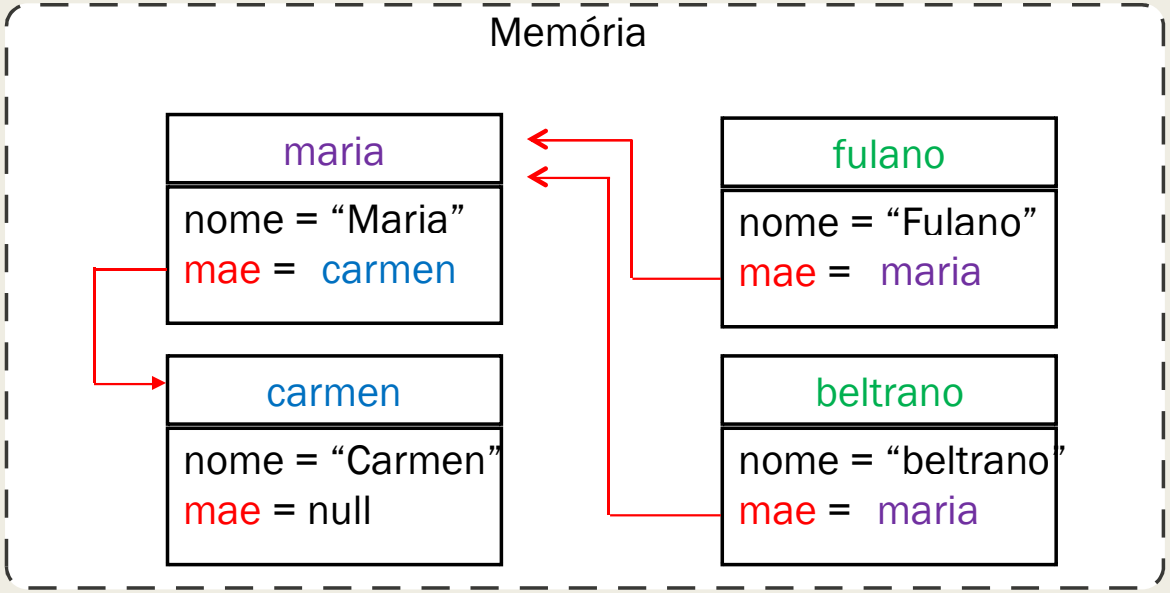
Nome da Classe
Atributos (visibilidade) nome variável: tipo variável + idNumber: String - peso: int { >0 } - CPF: String {###.###.###-##}
Métodos (visibilidade) nome método: tipo retorno (visibilidade) nome método (tipo parâmetro) + getNomePai: String + getIdade: int + setNomePai (String): + iniciar()

Palavra reservada “this” em Java

1º Forma de utilização: “*resolver ambiguidade de nomes entre variáveis declaradas em escopos diferentes,* ***por exemplo:*** *variáveis locais com o mesmo nome de variáveis de instancia.*”

```
==== Classe principal ====
{
    Pessoa fulano = new.....;
    Pessoa beltrano = new.....;
    Pessoa maria = new.....;
    fulano.setMae(maria);
    beltrano.setMae(maria);
    Pessoa carmen = new.....;
    maria.setMae(carmen);
..................................... }
Public Pessoa{
        String nome;
        Pessoa mae;

        public void setMae (Pessoa mae){
              this.mae = mae;       }
```



beltrano.mae.nome => “Maria”
fulano.mae.nome = “Mae do Fulano”;
beltrano.mae.nome => “Mae do Fulano”
beltrano.mae.mae.nome => “Carmen”

Palavra reservada “this” em Java

2º Forma de utilização: *“permite eliminar duplicidade de códigos dentro de construtores, chamando os construtores sobrecarregados recursivamente.”*

```
==== Classe principal ====
{
    1Pessoa mae = new Pessoa(“Maria”);
    2Pessoa pai = new Pessoa (“Jóse”, 58);
    3Pessoa fulano = new Pessoa(“João”,18, mae, pai);
    ..... }
    public Pessoa{
        String nome; int idade;
        Pessoa mae, pai;

    1public Pessoa(String nome){
        this.nome=nome;    }

    2public Pessoa(String nome, int idade){
        this(nome);
        this.idade=idade;    }

    public Pessoa(String nome, int idade, Pessoa mae){
        this(nome, idade);
        this.mae=mae;      }

    3public Pessoa(String nome, int idade, Pessoa mae,
    Pessoa pai){
        this(nome, idade,mae);
        this.pai=pai;   }
```

- *A recursividade dos construtores, permite o reaproveitamento do código, e facilita a manutenção, uma vez que as variáveis são atribuídas em apenas um local do código.*
- *Construtores sobrecarregados recursivamente, também podem ser utilizados para atribuir valores padrões aos atributos do objeto.*

Palavra reservada “this” em Java

2º Forma de utilização: “*Construtores sobrecarregados recursivamente permitem atribuir valores padrões aos atributos do objeto.*”

```
==== Classe principal ====
{     Pessoa alguem = new Pessoa(“”);
      Pessoa mae = new Pessoa(“Maria”);
      Pessoa pai = new Pessoa (“Jóse”, 58);
      Pessoa fulano = new Pessoa(“João”,18, mae, pai);
      ..................................... }
      public Pessoa{
            String nome; int idade;
            Pessoa mae, pai;

      public Pessoa(){
            this( “sem nome”); }

      public Pessoa(String nome){
            this(nome, 0 );  }

      public Pessoa(String nome, int idade){
            this(nome, idade, new Pessoa() );      }

      public Pessoa(String nome, int idade, Pessoa mae){
            this(nome, idade, mae, new Pessoa());      }

      public Pessoa(String nome, int idade, Pessoa mae,
      Pessoa pai){
            this.nome=nome;
            this.idade = idade;
            this.mae = mae;
            this.pai = pai; }
```

Palavra reservada “this” em Java

3º Forma de utilização: “Utilizado para atribuir o próprio objeto instanciado como referência a um método dentro do objeto”. **Exemplo**: adicione o objeto Filho a lista de filhos do objeto Mãe, que foi passado como parâmetro para o objeto Filho.

```
Pessoa fulano = new Pessoa(“João”, 18, mae, pai);
==================================
//************* Construtores **************
        public Pessoa(String nome){
            this.nome=nome;
        }
        public Pessoa(String nome, int idade) {
            this(nome);
            this.idade=idade;
        }
        public Pessoa(String nome, int idade, Pessoa mae){
            this(nome,idade);
            setMaternidade(mae);
    }
//*****************************************
```

```
// ********* Métodos Internos ***********
    private void setMaternidade(Pessoa mae){
        this.mae=mae;
        mae.addFilho(this);
    }
    private void addFilho(Pessoa filho){
        filhos.add(filho);
    }
// ***********************************
```

Objeto mae.addFilho(este objeto)

Exercício

1. Implemente um método que seja chamado no construtor, e permita adicionar os filhos do objeto ***Mae*** recebidos como parâmetro pelo objeto ***Filho***, na lista de irmãos do objeto Filho.

2. Modifique a classe principal para utilizar os métodos *getters* e *setters* e implemente os métodos necessários para responder as perguntas?
    1. Quantos filhos o objeto pessoa possui ?
    2. Qual o nome desses filhos ?
    3. Quantos irmãos o objeto pessoa possui ?
    4. Qual o nome desses irmãos ?

3. Modifique o método ***setMae*** da classe ***Pessoa***, para remover o objeto ***Filho*** da lista de filhos do objeto ***Mãe*** atual, antes de atualizar o valor da variável. Realize o mesmo procedimento para a lista de irmãos de todos os objetos envolvidos.

Bibliografia

- **Java Básico e Orientação a Objeto (livro);**
  - Clayton Escouper das Chagas; Cássia Blondet Baruque; Lúcia Blondet Baruque.
  - Fundação CECIERJ, 2010: https://canal.cecierj.edu.br/012016/d7d8367338445d5a49b4d5a49f6ad2b9.pdf
- **Java e Orientação a Objetos (apostila);**
  - Caelum, 2023: https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf
- **Modificadores de acesso em Java(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/modificadores-de-acesso-em-java/18822
- **Get e Set - Métodos Acessores em Java(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/get-e-set-metodos-acessores-em-java/29241
- **Sobrecarga e sobreposição de métodos em orientação a objetos(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/sobrecarga-e-sobreposicao-de-metodos-em-orientacao-a-objetos/33066
- **Análise e modelagem de sistemas com a UML: com dicas e exercícios resolvidos (livro);**
  - Luiz Antônio Pereira, 2011: https://luizantoniopereira.com.br/downloads/publicacoes/AnaliseEModelagemComUML.pdf