

Análise e desenvolvimento de aplicações orientadas a objeto com Java SE

Aula 03 – Conceitos de Orientação a Objeto



Orientação a Objetos

- É uma técnica de desenvolvimento de softwares que consiste em representar os elementos do mundo real (que pertencem ao escopo da aplicação) dentro do software.
- Em tese, é uma forma mais natural de informatização, já que leva em consideração os elementos como eles realmente existem.



Conceitos da orientação a objetos

- Entenda “conceitos” como uma série de regras e convenções que padronizam as aplicações orientadas a objetos e que possibilitam o uso de todos os recursos inerentes a essa técnica.
- Vale ressaltar que a orientação a objetos não é exclusividade da linguagem Java. Outras linguagem C#, VB.net, PHP 5, entre outras, fazem uso de tais recursos.




Abstração

- Abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.
- Habilidade mental que permite aos seres humanos visualizarem os problemas do mundo real com vários graus de detalhe, dependendo do contexto corrente do problema.

Elementos básicos da programação orientada a objetos

■ Classe


- Na verdade, na linguagem Java, **tudo** está definido em classes.
- A estrutura da própria linguagem é organizada em classes.
- Porém, inicialmente vamos nos ater as duas representações básicas de classes em orientação a objetos:
 - A classe de modelagem (ou definição de tipo).
 - A classe que contém a inicialização da aplicação.



Elementos básicos da programação orientada a objetos

■ Classe de modelagem


- Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.
- A classe de modelagem pode ser entendida como um molde, uma forma, um modelo que define as características e as funcionalidades dos futuros objetos que serão criados a partir dela.



Elementos básicos da programação orientada a objetos

■ Objeto

- Em modelagem orientada a objetos, um objeto é uma instância de uma classe existente no sistema de software.
- Um objeto representa uma entidade do mundo real dentro da aplicação de forma individual, possuindo todas as informações e funcionalidades abstraídas na concepção da classe.

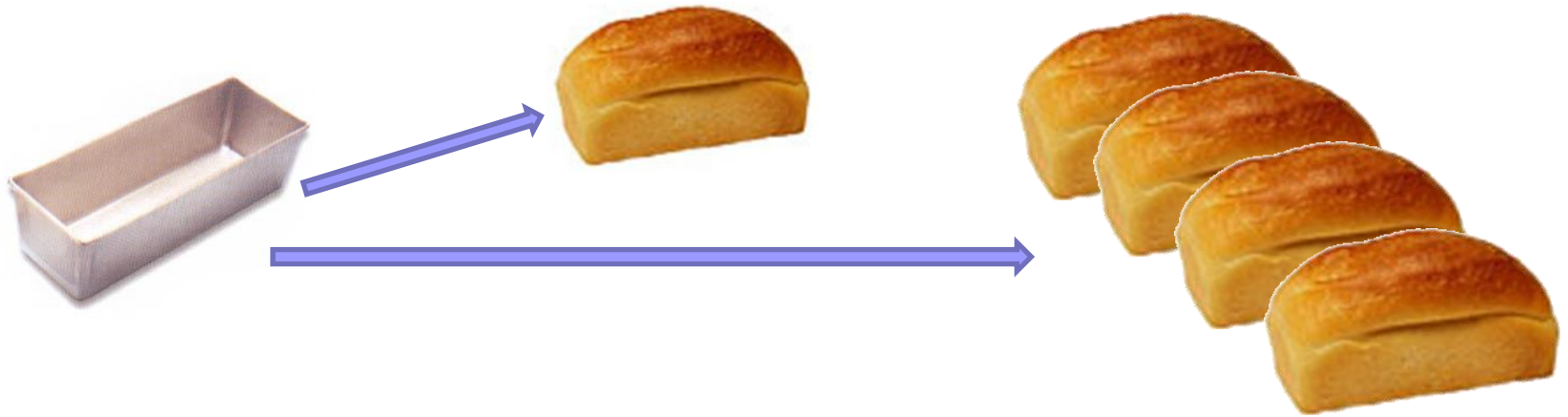



Elementos básicos da programação orientada a objetos

■ Classe de modelagem e Objetos

- Um objeto é criado (instanciado) a partir de uma classe e, portanto, possui todos os elementos definidos na classe.
- É possível criar quantos objetos forem necessários a partir de uma classe e todos terão as mesmas características e funcionalidades.

Analogias de Classe X Objeto






Estrutura básica e uma classe de modelagem

- Atributos ou variáveis de instância

- São as informações, os dados, que serão armazenados nos objetos.

- Métodos

- São as ações, as regras , as funcionalidades que serão executadas pelos objetos.



Estrutura básica e uma classe de modelagem

- Recapitulando

- Classe de modelagem (definição de tipo)

- Atributos (dados)

- Métodos (ações)

- Objetos (instâncias)

UML

- A partir do momento em que os elementos básicos da orientação a objetos são assimilados, podemos modelar classes nas especificações corretas utilizando a principal ferramenta de modelagem e documentação de aplicações orientadas a objeto existente no mercado:
- A UML (Unified Modeling Language ou Linguagem unificada de modelagem).

UML

- A UML não é uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objetos.
- Permite que desenvolvedores visualizem os produtos de seu trabalho em diagramas padronizados.
- Os objetivos da UML são: especificação, documentação, e estruturação para sub-visualização e maior visualização lógica de um total desenvolvimento de um sistema de informação.

Classe de modelagem: Usuario

- Abstração e modelagem da classe de usuário de uma aplicação.
 - Informações armazenadas em cada usuário (objeto).
 - nome
 - email
 - login
 - senha
 - Ações exercidas pelos (objetos do tipo) usuários na aplicação.
 - provar existência.

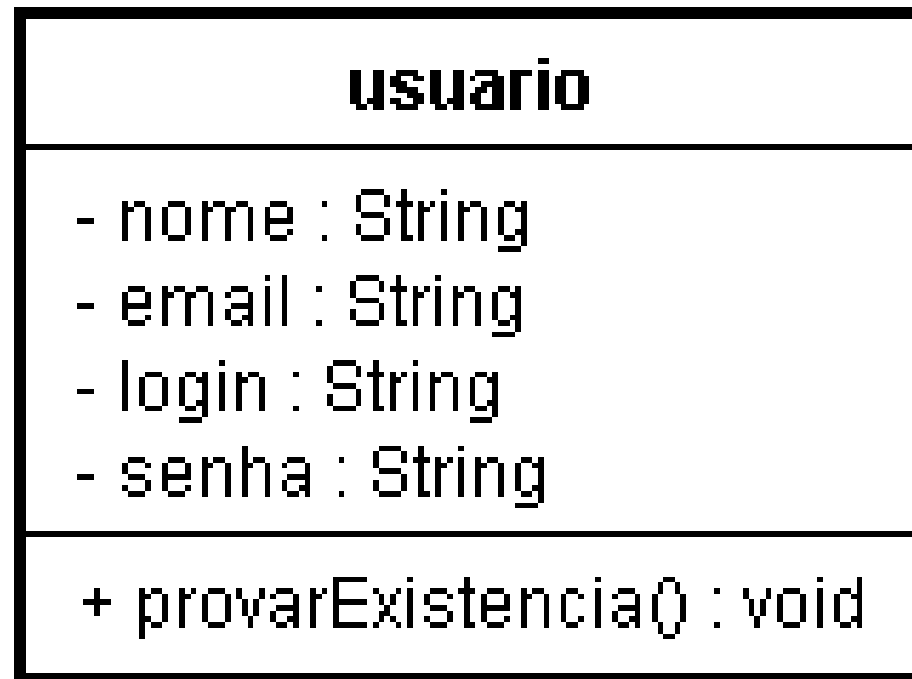


Classe de modelagem: Usuario

- Nome da classe
 - Usuario
- Atributos
 - nome, email, login e senha
- Métodos
 - provarExistencia

Classe de modelagem: Usuario

■ Em UML:





Classe de modelagem: Usuario

- Modificadores de acesso

- Private

- Só fica visível dentro da classe em que foi implementado

- Public

- Fica visível em toda a aplicação

- Protected

- Fica visível na classe em que foi implementado e em suas sub-classe (veremos em breve).

Codificação da classe Usuario

Atributos

```
public class Usuario {
```

```
    // Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;
```

Construtores

```
    // Construtores
    // Inicializa os atributos vazios
    public Usuario() {
        this("", "", "", "");
    }
    // Inicializa os atributos com valores passados por parametro
    public Usuario(String email, String login, String nome, String senha) {
        this.email = email;
        this.login = login;
        this.nome = nome;
        this.senha = senha;
    }
}
```

Getters e Setters

```
    // Getters e Setters
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    // Implementação dos demais getters e setters

    // Métodos específicos da classe
    public void provarExistencia() {
        System.out.println("Oi, eu existo!");
    }
}
```



Encapsulamento

- Consiste na proteção dos atributos de uma classe (e posteriormente dos objetos) de acessos externos.
- Considerando que todas as regras referentes a classe estão contidas na própria classe (e nunca em outra parte da aplicação), o acesso aos atributos deverão ser feitos de modo a garantir que tais regras sejam cumpridas.

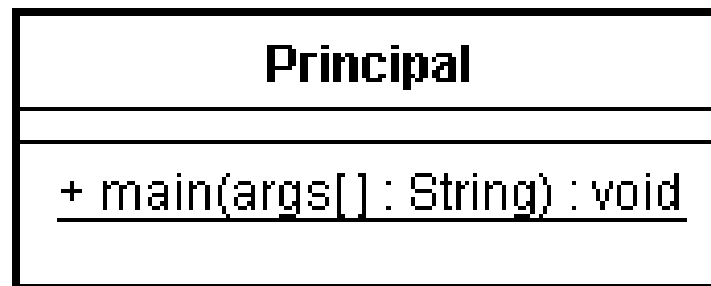


Encapsulamento

- Métodos getters e setters.
- Cada atributo tem seus próprios métodos públicos getter e setter.
 - Getter: Lê o conteúdo de um atributo e retorna seu valor.
 - Setter: Recebe um valor por parâmetro e altera (escreve) tal valor no referente atributo.

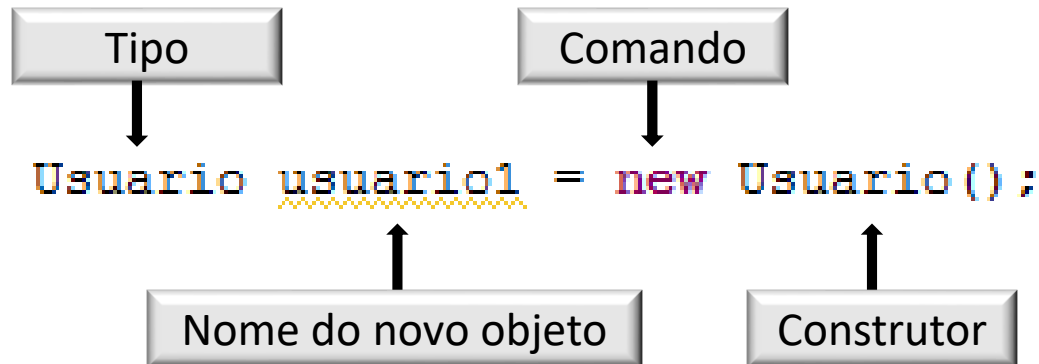
Classe: Principal

- Neste exemplo, a classe principal será usada somente para conter o método main



Como instanciar um objeto

```
public class Principal {  
  
    public static void main(String[] args) {
```



```
}
```

```
}
```

Como realizar uma chamada a método

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Usuario usuario1 = new Usuario();  
  
        usuario1.provarExistencia();  
  
        ↑           ↑  
Nome do novo objeto Método  
  
    }  
  
}
```

Codificação da classe Usuario

```
public class Usuario {  
  
    // Atributos  
    private String nome;  
    private String email;  
    private String login;  
    private String senha;  
  
    // Construtores  
    // Inicializa os atributos vazios  
    public Usuario() {  
        this("", "", "", "");  
    }  
    // Inicializa os atributos com valores passados por parametro  
    public Usuario(String email, String login, String nome, String senha) {  
        this.email = email;  
        this.login = login;  
        this.nome = nome;  
        this.senha = senha;  
    }  
  
    // Getters e Setters  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    // Implementação dos demais getters e setters  
  
    // Métodos específicos da classe  
    public void provarExistencia(){  
        System.out.println("Oi, eu existo!");  
    }  
}
```

Codificação da classe Principal

```
public class Principal {  
  
    public static void main(String[] args) {  
  
        Usuario usuario1 = new Usuario();  
  
        usuario1.provarExistencia();  
  
    }  
}
```


Objeto usuario1

Restante da
aplicação

Restante da
aplicação

