



Universidade Estadual de Maringá
Departamento de Informática



Elementos de uma classe Java

Conteúdo baseado nos materiais dos Professores:
Marcos Aurélio Domingues (DIN/UEM)
Edson Oliveira Junior (DIN/UEM)
Bruno Boniati (UFMS)

Prof.^a Juliana Keiko Yamaguchi
março de 2019

Objetivos

- Revisar conceitos de abstração, encapsulamento, objetos e classes.
- Aprender como criar classes em Java.
- Diferenciar classes de projeto e classes de implementação.

Introdução

Abstração na Computação

- Quando ocorre a abstração na computação?
- Como a abstração é representada no paradigma orientado a objetos?

Introdução

Abstração na Computação

- Abstração é a habilidade de se concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.
- No contexto da computação, é a capacidade de entender um problema da vida real, e propor uma solução computacional, por meio do mapeamento das entidades reais para um modelo.

Introdução

Abstração na Computação

- Na POO, as entidades abstratas são representadas por meio de classes/objetos.
- Qual a diferença entre classe e objeto?

Classes vs Objeto

- Uma classe é uma abstração para um conjunto de objetos.
- Uma classe pode representar uma entidade do mundo real.
- A classe é a unidade básica de trabalho em um programa orientado a objetos.
- A partir das classes, os objetos são instanciados, criados.

Classes vs Objeto

- Um objeto é a representação real da classe, com estado e apto a realizar ações dentro do sistema.
 - Ocupa espaço na memória e consome recursos do computador.

Classe em Java

- O código-fonte de uma classe em Java sempre está armazenado em um arquivo com a extensão .java.
- Um arquivo .java deve conter pelo menos uma classe.
 - Isso quer dizer que mais de uma classe pode ser declarada em um arquivo .java.

Classe em Java

Estrutura básica

- Cada classe deve conter pelo menos a sua assinatura mínima:

```
public class NomeDaClasse{  
  
    //código fonte da classe  
  
}
```

- O NomeDaClasse dá nome ao arquivo que a contém.
 - No exemplo, teríamos NomeDaClasse.java

Classe em Java

Estrutura básica

- Por convenção, o nome da classe inicia com a primeira letra maiúscula.

```
public class NomeDaClasse{  
  
    //código fonte da classe  
  
}
```

- Assim como em C, os blocos de instrução são delimitados pelos símbolos { e }.

Classe em Java

Elementos básicos

- Uma classe Java pode conter os seguintes elementos:
 - Atributos
 - Métodos
 - Construtor(es)
 - Obrigatório ter pelo menos um construtor.
 - Método main
 - Se for a classe principal da aplicação.

Classe em Java

Modificadores de acesso

- Os elementos de uma classe em Java, ou os membros de uma classe, são declarados com modificadores de acesso.
- Modificadores de acesso determinam as regras de escopo.
 - Regra de escopo refere-se ao grau de acesso que outras classes podem ter sobre determinado elemento.

Classe em Java

Modificadores de acesso

- Tipos de modificadores:

Modificador	Símbolo	Escopo
Público – public	+	Acesso permitido a todas as classes do sistema.
Privado – private	-	Acesso restrito somente à classe na qual o membro está declarado.
Protegido – protected	#	Acesso permitido somente por herança e por classes do mesmo pacote.
Padrão – default	~	Acesso permitido somente por classes do mesmo pacote.

Classe em Java

Atributos

- Atributos também são conhecidos como propriedades ou variáveis de instância.
 - Representam os dados que a classe manipula.
- Representação genérica:

```
public class NomeDaClasse{  
  
    modificador tipo nomeDoAtributo;  
  
}
```

Classe em Java

Atributos

- Os modificadores de acesso para atributos podem ser:
 - Público (+ *public*)
 - Privado (- *private*)
 - Protegido (# *protected*)
- Se não houver declaração do modificador, por padrão o modificador será *default*.

Classe em Java

Atributos

- Os atributos devem ter seu tipo declarado.
- Os tipos de dados em Java podem ser classificados em:
 - Tipos primitivos
 - boolean, char, byte, short, int, long, float, double.
 - Tipos Objeto
 - Representam as referências às classes que mantêm uma relação associativa.

Classe em Java

Métodos


- Os métodos representam as operações que os objetos da classe podem executar para manipular seus atributos ou se comunicar com outras classes.
- Um método sempre deve indicar o tipo de retorno:
 - Caso o método não retorne nenhum valor, a palavra reservada “void” deve ser usada.
- Um método pode conter ou não parâmetros.

Classe em Java

Métodos

- Representação genérica:

```
public class NomeDaClasse{  
  
    modificador tipo nomeDoAtributo;  
  
    modificador tipoRetorno nomeMetodo() {  
        //código fonte do método  
    }  
}
```



Lista de parâmetros é formado por zero, uma ou mais variáveis precedidas pelo seu tipo.

Classe em Java

Métodos

- Os modificadores de acesso para método podem ser:
 - Público (+ *public*)
 - Privado (- *private*)
 - Protegido (# *protected*)
- Se não houver declaração do modificador, por padrão o modificador será *default*.

Classe em Java

Construtor

- Método especial chamado automaticamente pelo ambiente de execução quando um objeto é criado.
- Comando para criar um objeto:

```
NomeDaClasse nomeObjeto = new NomeDaClasse();
```

- Exemplo:

```
Aluno aluno = new Aluno();
```

Classe em Java

Construtor

- Não é obrigatório declarar o construtor da classe.
- O próprio compilador provê um construtor padrão.
- Quando é necessário declarar um construtor da classe?

Classe em Java

Construtor

- Opcionalmente, pode-se declarar um construtor para criar um objeto quando deseja-se que seu estado inicializado.
 - Ou quando deseja-se que algum método seja acionado ao criar determinado objeto.

- Exemplo:

```
Aluno aluno = new Aluno(Maria) ;
```

- Nesse exemplo, o construtor padrão da classe Aluno foi sobrescrito para que o objeto seja criado com um nome.

Classe em Java

Construtor

- Regras para declarar (sobrescrever) construtores:
 - Uma classe pode ter um ou mais construtores.
 - SEMPRE tem o mesmo nome da classe e pode opcionalmente receber parâmetros.
 - Nunca pode ter um tipo de retorno (isso é implícito pois ele retorna uma referência para um objeto da classe em questão).

Classe em Java

Construtor – Exemplo

```
public class Hora {  
    private int hora;  
    private int minuto;  
    private int segundo;  
  
    public Hora() {  
        super();  
    }  
  
    public Hora(int h, int m, int s) {  
        setHorario(h, m, s);  
    }  
  
    public Hora(int h) {  
        setHorario(h, 0, 0);  
    }  
  
    public void setHorario(int h, int m, int s) { }  
    public void setHora(int h) { }  
    public void setMinutos(int m) { }  
    public void setSegundos(int s) { }  
  
    public String getHorario() { }  
}
```

Métodos construtores com assinaturas diferentes

Classe em Java

Construtor – Exemplo

- Em uma classe, podemos ter:

```
public class Aplicacao{  
  
    public static void main(String[] args){  
        Hora horario = new Hora();  
        Hora horaDaAula = new Hora(19, 30, 00);  
        Hora horaDoAlmoco = new Hora(12);  
    }  
}
```

Objeto em Java

- Precisa ser identificado enquanto entidade do código (uma variável, um elemento em uma lista, uma posição de um vetor, etc).
- O operador **new**:
 - Aloca memória para o novo objeto (a quantidade necessária é obtida a partir da definição da classe);
 - Chama o construtor da classe para inicializar o estado do novo objeto;
 - Retorna uma referência (um endereço de memória para o objeto recém criado).

Classe em Java

Método main()

- Permite que uma classe seja executada (normalmente pelo menos uma das classes da aplicação precisa ser executada).
- O método main() existe somente quando uma classe é considerada a classe principal de uma aplicação.
- É por meio desse método que uma aplicação entra em execução.

Classe em Java

Método main()

- Quando uma aplicação java é iniciada, a JVM localiza e chama o método main() que recebe por parâmetro um vetor de objetos String representando os parâmetros de linha de comando.
- Exemplo:

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

Classe em Java

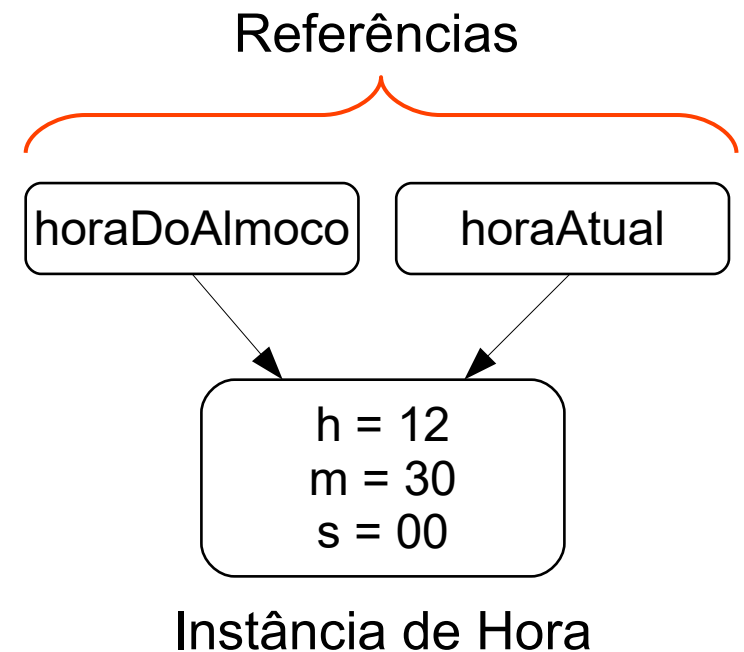
Método main()

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

- O modificador `public` deixará visível para todas as outras classes, subclasses e pacotes do projeto Java.
- A palavra-chave `static` significa:
 - Não precisa instanciar o objeto na memória;
 - Não precisa de construtor, basta usar o método `main`.
- O retorno `void` significa que o método não retorna nada.
- `String[] args` são argumentos passados por linha de comando.

Atribuição de referências

```
public class Aplicacao {  
  
    public static void main(String[] args) {  
        Hora horaDoAlmoco;  
  
        horaDoAlmoco = new Hora();  
        horaDoAlmoco.setHorario(12, 30, 00);  
  
        Hora horaAtual = new Hora();  
  
        horaAtual = horaDoAlmoco;  
    }  
}
```



- Quando atribuímos uma referência de um objeto para outra referência, temos ao final duas referências para um único objeto e não uma cópia do objeto (como seria com tipos primitivos).

Referência `this`

- Todos os métodos de instância recebem um parâmetro implícito chamado `this` que pode ser utilizado dentro do método de instância para se referir ao objeto corrente (cujo método foi invocado).

Referência `this`

- É normalmente utilizado em duas situações:
 - Quando o nome de uma variável de instância é igual ao nome de um parâmetro do método;
 - Quando o método precisa chamar outro método passando por parâmetro uma referência ao objeto atual.

Referência this

```
public class Hora {  
    private int hora;  
    private int minuto;  
    private int segundo;  
  
    public Hora() {  
        super();  
    }  
  
    public Hora(int h, int m, int s) {  
        setHorario(h, m, s);  
    }  
  
    public Hora(int h) {  
        setHorario(h, 0, 0);  
    }  
  
    public void setHorario(int hora, int minuto, int segundo) {  
        this.hora = hora;  
        this.minuto = minuto;  
        this.segundo = segundo;  
    }  
}
```

Neste caso this resolve a
ambiguidade de nomes
(parâmetros x atributos)

Exercício

- Crie as classes de projeto e de implementação para o seguinte cenário:
 - Deseja-se um sistema de controle de estoque de produtos de uma loja. Cada produto é adquirido de um fornecedor.
 - Deve-se armazenar no sistema:
 - Dados do produto: nome, preço, código;
 - Dados do fornecedor: nome, telefone, e-mail.

Atividade

- Implemente as classes para o seguinte cenário:
 - Um cinema exibe diversos filmes. Cada filme é exibido em uma determinada sala, em diferentes horários, denominados de sessão.
 - Deve-se considerar:
 - O título do filme;
 - O número da sala;
 - O horário da sessão;
 - Quantidade de assentos disponíveis cada sala.