PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
TÉCNICAS DE PROGRAMAÇÃO 1 – CMP 1046
PROF. MSC. ANIBAL SANTOS JUKEMURA

Classes

PUC GOIÁS

Agenda:

- Classes/Construtores
- Modificadores de acesso
- Exercício: implementação de classe.
- Relacionamento entre classes (Associação)
- Exercício: implementação de classe.

PUC GOIÁS

Construtores

- *Definição*

Também conhecidos pelo inglês **constructors**, os construtores são os responsáveis por **criar o objeto em memória**, ou seja, instanciar a classe que foi definida.

- *Construtor padrão*

Quando uma classe *não* declara um construtor, o compilador fornece um construtor padrão. Cada variável de instância recebe implicitamente o valor padrão (zero para tipos numéricos primitivos, false para valores boolean e null para referências).

PUC GOIÁS

Construtores

- *Declaração:* modificadores de acesso + nome da classe + parâmetros

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;

    testeAcesso()
    {
        hora=0;
        minuto=0;
        segundo=0;
    }
}
```

OU

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;

    public testeAcesso()
    {
        hora=0;
        minuto=0;
        segundo=0;
    }
}
```

Boa Prática!

PUC GOIÁS

Construtores

- *Nota sobre **Polimorfismo***

Polimorfismo significa "muitas formas", é o termo definido em linguagens orientadas a objeto, como por exemplo Java, C# e C++, que permite ao desenvolvedor usar o mesmo elemento de formas diferentes.

Polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem. No Polimorfismo temos dois tipos:

- **Polimorfismo Estático ou Sobrecarga**
- **Polimorfismo Dinâmico ou Sobreposição**

PUC GOIÁS

Construtores

- *Nota sobre Polimorfismo*

- **Polimorfismo Estático** ou **Sobrecarga** (caso do construtor)

  O Polimorfismo Estático se dá quando temos a mesma operação implementada várias vezes na mesma classe. A escolha de qual operação será chamada depende da assinatura dos métodos sobrecarregados.

- **Polimorfismo Dinâmico** ou **Sobreposição**

  O Polimorfismo Dinâmico acontece na herança, quando a subclasse sobrepõe o método original. Agora o método escolhido se dá em tempo de execução e não mais em tempo de compilação. A escolha de qual método será chamado depende do tipo do objeto que recebe a mensagem.

PUC GOIÁS

Construtores – sobrecarregados (polimorfismo estático)

- *Declaração:* modificadores de acesso + nome da classe + parâmetros

```
public testeAcesso()
{
    hora=0;
    minuto=0;
    segundo=0;
}
```

```
public testeAcesso(int h, int m, int s)
{
    hora=h;
    minuto=m;
    segundo=s;
}
```

PUC GOIÁS

Construtores – sobrecarregados (polimorfismo estático)

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;

    public testeAcesso()
    {
        hora=0;
        minuto=0;
        segundo=0;
    }

    public testeAcesso(int h, int m, int s)
    {
        hora=h;
        minuto=m;
        segundo=s;
    }
}
```

Classe com dois construtores

PUC GOIÁS

Construtores – sobrecarregados (polimorfismo estático)

```
public class JavaApplication14 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        testeAcesso objeto1 = new testeAcesso();
        testeAcesso objeto2 = new testeAcesso(10,30,15);

        System.out.println("Hora Certa: " + objeto1.getHora() + ":"
                           + objeto1.getMinuto() + ":" + objeto1.getSegundo());

        System.out.println("Hora Certa: " + objeto2.getHora() + ":"
                           + objeto2.getMinuto() + ":" + objeto2.getSegundo());
    }

}
```

Saída

```
run:
Hora Certa: 0:0:0
Hora Certa: 10:30:15
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

PUC GOIÁS

Destrutores em Java

- Não existe o conceito de destrutores em Java, mas você nem precisa se preocupar com isto, pois não tem como literalmente destruir um objeto, assim como você faz em C/C++. Isso porque não é garantido que o ***Garbage Collection*** irá destruir este objeto, já que ele o faz na hora que achar conveniente e o programador não tem nenhum controle sobre isso.

- A forma mais adequada de "tentar" destruir um objeto em Java é atribuir valores nulos a ele. Assim, quando o ***Garbage Collection*** for realizar o seu trabalho, verá que seu objeto não está sendo mais utilizado e o destruirá.

PUC GOIÁS

Modificadores de Acesso

- Os modificadores de acesso **public** e **private** controlam o acesso a variáveis e métodos de uma classe.

- **public** :
  - É o menos restritivo de todos. Atributos e métodos declarados como public em uma classe podem ser acessados pelos métodos da própria classe, por classes derivadas desta e por qualquer outra classe em qualquer outro pacote (veremos mais sobre pacotes e classes derivadas posteriormente).
  - O principal objetivo dos métodos **public** é apresentar aos clientes da classe uma visualização dos serviços que a classe fornece (isto é, a interface public dela).

PUC GOIÁS

Modificadores de Acesso

- **private** :
  - É o mais restritivo. Atributos e métodos declarados como private só podem ser acessados pelos métodos da própria classe.
  - As variáveis private e os métodos private da classe (isto é, seus *detalhes de implementação*) ***não*** **são acessíveis aos clientes** (**fora da classe**).
  - Os membros private de uma classe **só são acessíveis dentro da própria classe**.

- **protected** :
  - Atributos e métodos definidos como protected são acessíveis pelos métodos da própria classe e pelas classes derivadas.
  - Oferece um nível intermediário de acesso entre **public** e **private**.
  - Veremos isso na aula de Herança.

PUC GOIÁS

Modificadores de Acesso – Exemplo PUBLIC

```
public class testeAcesso {
    public int hora;
    public int minuto;
    public int segundo;
}
```

```
public static void main(String[] args) {
   testeAcesso objeto = new testeAcesso();

   objeto.hora=10;
   objeto.minuto = 20;
   objeto.segundo = 15;
    System.out.println("Hora Certa: " + objeto.hora + ":" + objeto.minuto + ":" + objeto.segundo);
}
```

Obs.: Apesar de permitir, não é uma boa prática de programação

PUC GOIÁS

Modificadores de Acesso – Exemplo PRIVATE

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;
}
```

```
public static void main(String[] args) {
    testeAcesso objeto = new testeAcesso();

    objeto.hora=10;
    objeto.minuto = 20;
    objeto.segundo = 15;
     System.out.println("Hora Certa: " + objeto.hora + ":" + objeto.minuto + ":" + objeto.segundo);
}
```

```
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - hora has private access in javaapplication14.testeAcesso
        at javaapplication14.JavaApplication14.main(JavaApplication14.java:20)
C:\Users\anibal.jukemura\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
FALHA NA CONSTRUÇÃO (tempo total: 1 segundo)
```

PUC GOIÁS

Modificadores de Acesso – Exemplo PRIVATE

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;
}
```

```
public static void main(String[] args) {
   testeAcesso objeto = new testeAcesso();

   objeto.setHora(10);
   objeto.setMinuto(20);
   objeto.setSegundo(15);
    System.out.println("Hora Certa: " + objeto.getHora() + ":"
                    + objeto.getMinuto() + ":" + objeto.getSegundo());
}
```

PUC GOIÁS

Modificadores de Acesso – Exemplo de método PRIVATE

```
public class testeAcesso {
    private int hora;
    private int minuto;
    private int segundo;
}
```

```
private void resetaHora()
{
    hora=12;
    minuto=0;
    segundo=0;
}

public testeAcesso()
{
    resetaHora();
}
```

Método private resetaHora usado em Construtor

PUC GOIÁS

Modificadores de Acesso: *Métodos* set *e* get versus *dados public*

- São chamados de **métodos de acesso** ou **métodos de consulta**

- Parece que fornecer as capacidades de *set* e *get* é essencialmente o mesmo que tornar public as variáveis de instância da classe.

- Essa é uma das sutilezas que torna o Java tão desejável para a engenharia de software.

- Uma variável de instância public pode ser lida ou gravada por qualquer método que tem uma referência que contém a variável. Se uma variável de instância for declarada private, um método *get* public certamente permitirá que outros métodos a acessem, mas o método *get* pode *controlar* como o cliente pode acessá-la.

- Embora os métodos *set* e *get* possam fornecer acesso a dados private, o acesso é restrito pela implementação dos métodos. Isso ajuda a promover uma boa engenharia de software.

PUC GOIÁS

Exercício

Escreva um programa com uma classe chamada **classeData** com os atributos **private** inteiros dia, mês e ano.
A classe terá dois construtores: o primeiro inicializa dia e mês com valores 1 e ano igual a 1900. O segundo, inicializa os atributos através de passagem de parâmetros.
Crie um método **public** chamado **verificaData**, que retornará um valor booleano ***true***, se a data é válida e ***false*** caso contrário. Para usar esse método, crie três métodos **private**: **checarDia**, **checarMes**, **checarAno** e chame-os no método **verificaData**.
Peça ao usuário entrar via teclado com o dia, mês e o ano.

Obs.: Não se esqueça dos anos bissextos!

PUC GOIÁS

Exercício – Dicas Técnicas

```
private boolean anoBissexto()
{
    if ((ano%4!=0))
        return false;
    else if ((ano%4==0) && (ano%100!=0))
          return true;
    else if ((ano%4==0) && (ano%100==0) && (ano%400!=0))
          return false;
    else if ((ano%4==0) && (ano%100==0) && (ano%400==0))
        return true;
    return false;
}
```

PUC GOIÁS

Exercício – Dicas Técnicas

```
import java.util.stream.IntStream;

private boolean checarDia()
{
    int[] mes30 = new int[]{1,3,4,5,6,7,8,9,10,11,12};
    int[] mes31 = new int[]{1,3,5,7,8,10,12};
    if (dia>=1 && dia<=28)
      return true;
    else if (dia==29 && mes!=2)
        return true;
    else if (dia==30 && (IntStream.of(mes30).anyMatch(x -> x == mes)))
        return true;
    else if (dia==31 && (IntStream.of(mes31).anyMatch(x -> x == mes)))
        return true;
    else if (dia==29 && mes==2 && anoBissexto())
      return true;
    return false;

}
```

PUC GOIÁS

Associação

- Uma classe pode ter referências a objetos de outras classes como membros. Isso é chamado **Associação** e, às vezes, é referido como um **relacionamento tem um**.

- Na **Associação** temos uma instância da classe existente sendo usada como componente da outra classe.

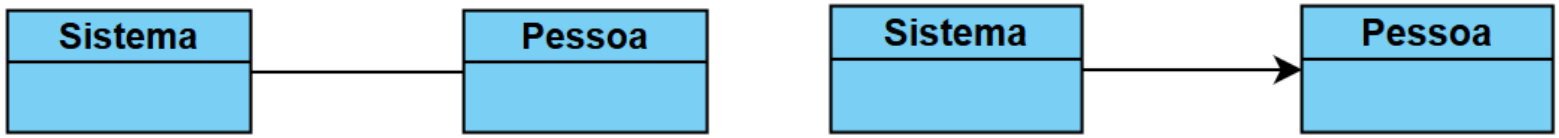
- Por exemplo, podemos dizer que um carro TEM UM pneu, neste exemplo temos a definição de uma **Associação** .

Obs.: Temos outros tipos de relacionamento entre classes: agregação e composição. O livro Java do Deitel trata uma associação como sendo uma composição.

PUC GOIÁS

Associação

UML: Diagrama de classe



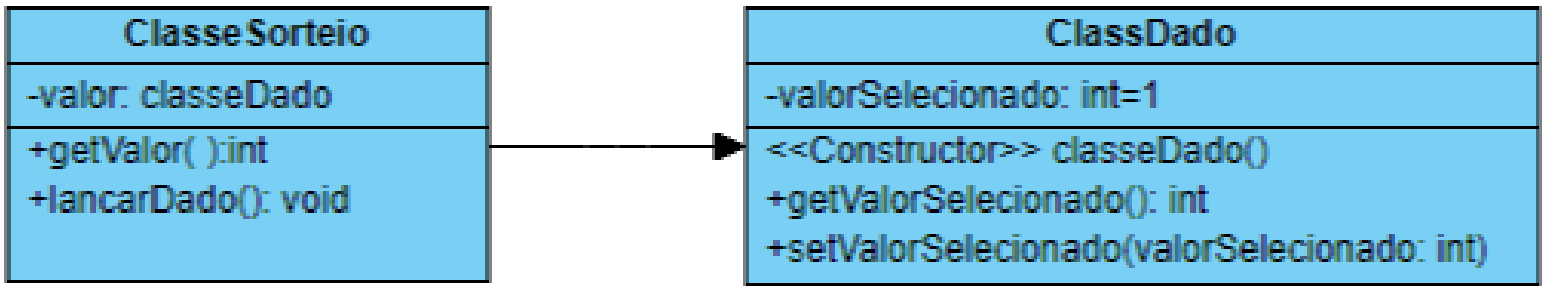

```
public class Sistema {

    Pessoa pessoa = new Pessoa();

}

public class Pessoa {

}
```

Representamos as associações por meio de retas que ligam as classes envolvidas, essas ligações podem ou não possuir setas nas extremidades indicando a navegabilidade da associação

PUC GOIÁS

Associação - Exemplo

EM SALA



PUC GOIÁS

Agregação

- É um **tipo especial de associação** onde tenta-se demonstrar que as informações de um objeto (chamado objeto-todo) precisam ser complementados pelas informações contidas em um ou mais objetos de outra classe (chamados objetos-parte); conhecemos como todo/parte.

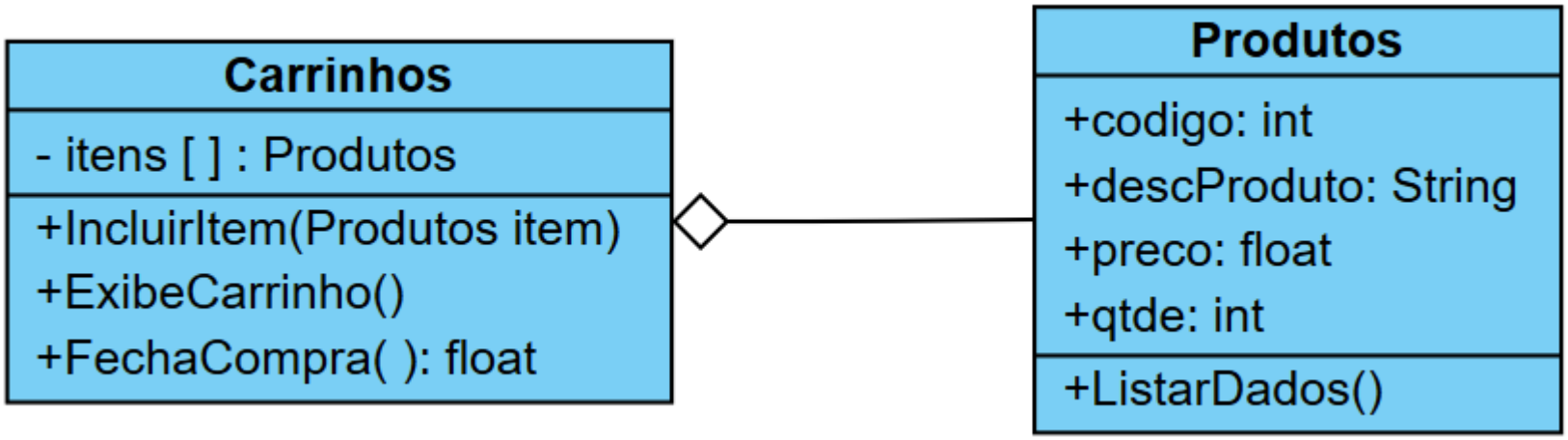
- O objeto-pai poderá usar as informações do objeto agregado.

- Nesta relação, um objeto poderá agregar uma ou mais instâncias de um outro objeto.

- Para agregar muitas instâncias, a forma mais simples é utilizando arrays. Criamos um array como atributo da classe, sendo que o papel deste array é armazenar inúmeras instâncias de uma outra classe

PUC GOIÁS

Agregação

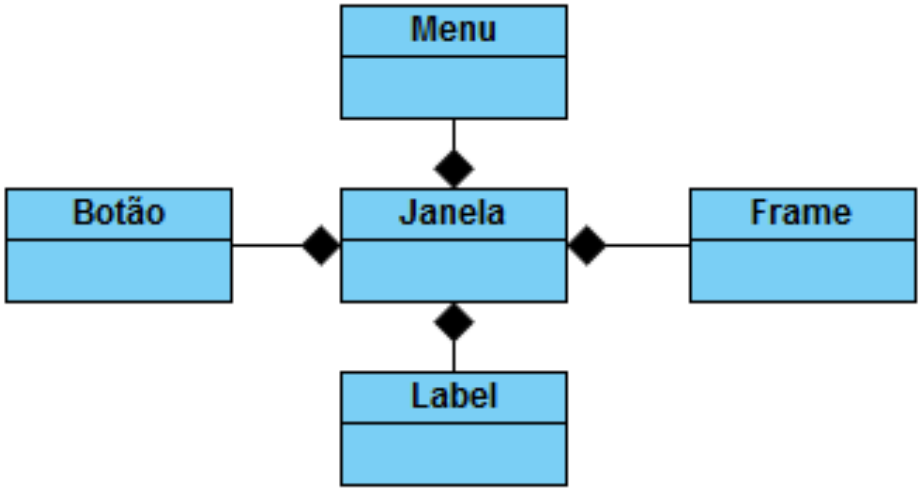


Carrinho de compras (classe Carrinhos) com vários itens do tipo produtos (classe Produtos). Para agregar os produtos ao carrinho, usa-se o método IncluirItem( ) na classe Carrinhos, que contém outro método chama ExibeCarrinho() responsável por listar todos os itens pedidos, por meio da listagem dos dados do produto -método ListarDados() da classe Produtos-, e um método FechaCompra( ) reponsável por efetuar a soma dos itens adicionados no carrinho apresentando ao final o preço a ser pago pelo cliente.

PUC GOIÁS

Composição

- Pode-se dizer que composição **é uma variação da agregação**.
- Uma composição tenta representar também uma relação todo - parte.
- No entanto, na composição o objeto-pai (todo) é responsável por criar e destruir suas partes. Em uma composição um mesmo objeto-parte não pode se associar a mais de um objeto-pai.



PUC GOIÁS

BOAS PRÁTICAS DE PROGRAMAÇÃO

Erro comum de programação 8.1

Uma tentativa por um método que não é membro de uma classe de acessar um membro `private` *dessa classe é um erro de compilação.*

Observação de engenharia de software 8.2

Lembre-se do que foi discutido no Capítulo 3, de que os métodos declarados com o modificador de acesso `private` *só podem ser chamados por outros métodos da classe em que os métodos* `private` *são declarados. Esses métodos são comumente chamados de* ***métodos utilitários*** *ou* ***métodos auxiliares*** *porque eles são tipicamente utilizados para suportar a operação dos outros métodos da classe.*

Observação de engenharia de software 8.3

Classes simplificam a programação porque o cliente só pode utilizar os métodos `public` *de uma classe . Normalmente, esses métodos são direcionados aos clientes em vez de à implementação. Os clientes não estão cientes de, nem envolvidos em, uma implementação da classe. Eles geralmente se preocupam com* o que *a classe faz, mas não* como *a classe faz isso.*

PUC GOIÁS

BOAS PRÁTICAS DE PROGRAMAÇÃO

Dica de prevenção de erro 8.2

Certifique-se de que você não inclui um tipo de retorno na definição de um construtor. O Java permite que outros métodos da classe além dos seus construtores tenham o mesmo nome da classe e especifiquem tipos de retorno. Esses métodos não *são construtores e* não *serão chamados quando um objeto da classe for instanciado.*

Erro comum de programação 8.3

Um erro de compilação ocorre se um programa tenta inicializar um objeto de uma classe passando o número ou tipo errado de argumentos para o construtor da classe.

Observação de engenharia de software 8.7

Se apropriado, forneça métodos `public` *para alterar e recuperar os valores de variáveis de instância* `private`*. Essa arquitetura ajuda a ocultar a implementação de uma classe dos seus clientes, o que aprimora a modificabilidade do programa.*

Dica de prevenção de erro 8.4

Usar métodos set *e* get *ajuda a criar classes que são mais fáceis de depurar e manter. Se apenas um método realizar uma tarefa particular, como configurar uma instância de variável em um objeto, é mais fácil depurar e manter a classe. Se a variável de instância não for configurada corretamente, o código que na verdade modifica a variável de instância estará localizado em um único método* set*. Seus esforços de depuração podem focalizar esse único método.*

Referência Bibliográfica Principal

• DEVMEDIA. Disponível em https://www.devmedia.com.br. Acessado em Julho de 2019.

• DEITEL, Harvey M. Java: Como Programar – 10 ed. Caps 03, 08. 2015.