**Programação Orientada a Objetos – Resumo Prova 1**

**01 - Introdução à programação orientada a objetos**

**Paradigmas de programação**

Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Paradigma orientado a objetos**

* Surgiu na década de 60.
* Primeiras linguagens comerciais surgiram na década de 90.
* É uma visão contemporânea que utiliza a perspectiva de objetos.
* Capaz de ser usado em qualquer tipo de sistema.

| Principais objetivos são: | Em contrapartida: |
| --- | --- |
| – Melhorar a compreensão do sistema. | – Maior curva de aprendizagem. |
| – Alto grau de reutilização. | – Programas maiores. |
| – Facilidade de manutenção. | – Não recomendável para qualquer tipo de problema. |
| – Facilidade de evolução. |  |
| – Maior qualidade. |  |
| – Maior produtividade e menor custo. |  |

**Exemplo sobre Classes, Objetos, Métodos e Atributos:**

Tela de computador com imagem de cachorro

Descrição gerada automaticamente

**02 - Diagrama de classes**

**UML**

A UML (Unified Modeling Language – linguagem de modelagem unificada) é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de orientação a objetos.

**Diagrama de Classes**

* O diagrama de classes mostra um conjunto de classes (entre outros elementos) e seus relacionamentos.
* O diagrama de classes é usado para modelar uma visão estática de um software.

Diagrama

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

**Criação de Classes, Atributos ou Operações (Métodos)**

**Classes**

* São substantivos ou expressões breves, definidos a partir do vocabulário do sistema.
* Utilizam o primeiro caractere em letra maiúscula para cada palavra existente no nome da classe (estilo UpperCamelCase)
* Exemplos de nomes de classes:
  + Cliente.
  + Estudante.
  + Venda.
  + ContaBancaria.
  + SensorTemperatura.

**Atributos**

* Geralmente é um substantivo ou expressão que representa alguma propriedade da classe.
* Utilizam o primeiro caractere em letra maiúscula para cada palavra existente no nome, exceto para a primeira letra (estilo lowerCamelCase).
* Todo atributo possui um tipo de dado, que delimita os possíveis valores para o atributo. Os principais tipos em Java, são:
  + boolean: Indica um tipo lógico, com dois valores possíveis (verdadeiro ou falso).
  + int: Permite representar números inteiros.
  + double: Permite representar números decimais.
  + String: Permite representar texto.

**Operações**

* O nome é um verbo ou locução verbal, representando algum comportamento (ação) da classe.
* Utilizam o estilo lowerCamelCase.
* Contém um par de parênteses após o nome.
* Podem ser de dois tipos:
  + Mudam o estado do objeto.
  + A partir do estado do objeto, realizam alguma operação, resultando num valor.

**03 - Implementação - Classes e Objetos**

**Utilização da POO**

Diagrama

Descrição gerada automaticamente

**Declaração de Classe**

Diagrama

Descrição gerada automaticamente

**Métodos: Devolvem um valor**

Interface gráfica do usuário, Diagrama, Aplicativo

Descrição gerada automaticamente

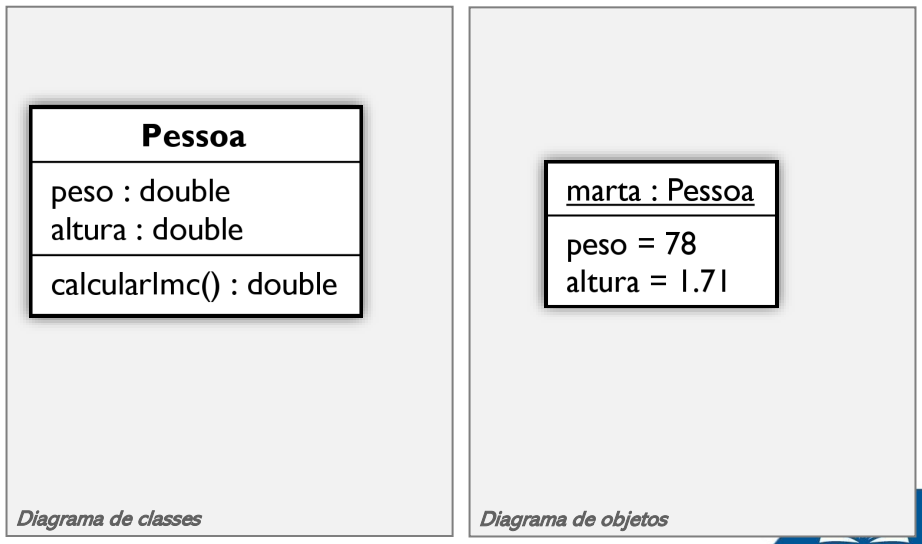
**Métodos: Alterando estado do objeto**

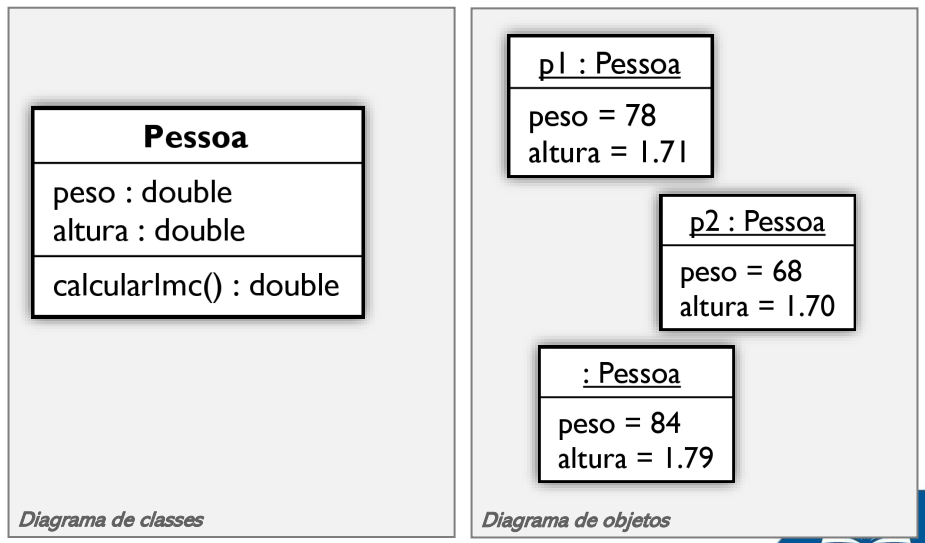
Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**05 - Diagrama de Objetos**

* É um diagrama que mostra uma fotografia do estado detalhado de um sistema, num determinado instante do tempo.
* É um diagrama da UML que contém somente objetos (não contém classes).
* Fornece uma perspectiva concreta de objetos e seus relacionamentos.
* Uso limitado, pois apresenta somente estruturas de dados.





**06 - Escopo de Variáveis**

* O escopo de uma variável denota sua visibilidade no programa, isto é, onde a variável é acessível.
* Fora deste contexto, o identificador não pode ser utilizado, isto é, está fora do escopo da variável.

Diagrama

Descrição gerada automaticamente

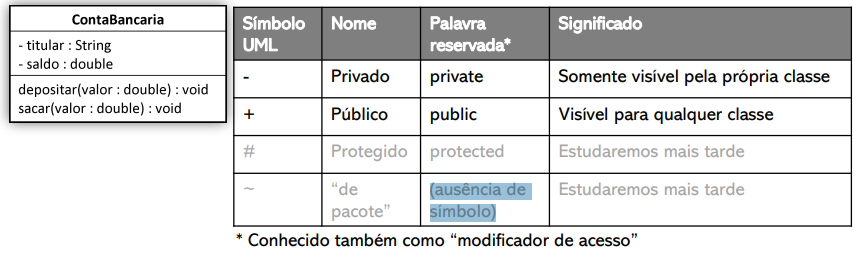
**Inicialização das Variáveis**

* As variáveis declaradas num método ou num bloco não possuem valor inicial.
  + Somente é possível ler o valor da variável depois de atribuir explicitamente um valor para a variável.
* As variáveis de instância têm valor padrão. Java automaticamente inicializa (atribui um valor) para estas variáveis:
  + Variáveis numéricas são inicializadas com 0 (zero).
  + Variáveis booleanas são inicializadas com false.
  + Variáveis de referência são inicializadas com null.

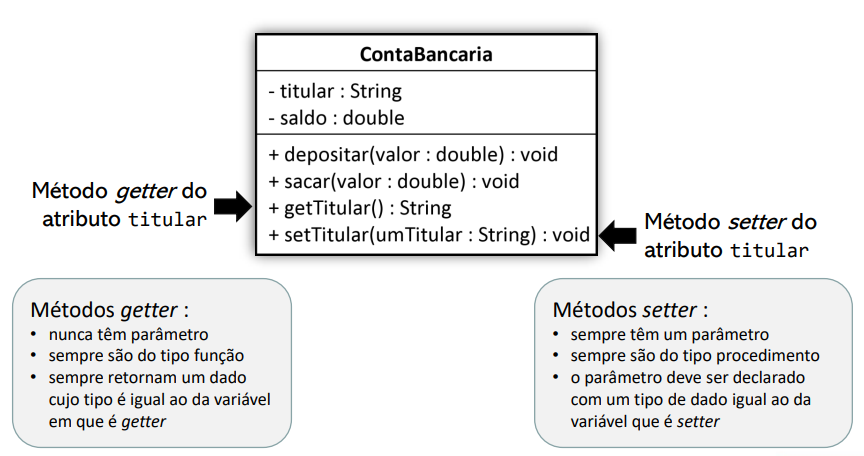
**07 – Encapsulamento**

**Motivação Encapsulamento**

* O acesso ao atributo deve ser “controlado”, para garantir integridade dos dados – Isto é, o estado do objeto precisa ser “controlado”.
* Somente o próprio objeto deveria manipular o valor de seus atributos.
* Esta técnica se chama encapsulamento de dados.
* Em Java, para aplicar o encapsulamento é preciso tornar o atributo privado.



* Todos os atributos de um objeto deveriam ser encapsulados.
* Os atributos que precisam ser acessados por outras classes poderão ser acessados por meio de métodos de acesso.
* Os métodos de acesso são métodos, geralmente públicos.Dividem-se em:
  + Getters: métodos usados para recuperar valor de atributos de um objeto da classe. O nome de um método getter deve ser escrito com o prefixo get, seguido do nome do atributo com a inicial maiúscula. *Exceção: se o atributo for lógico (booleano), usar o prefixo “is”.*
  + Setters: métodos usados para atribuir valor de atributos da classe. O nome de um método setter deve ser escrito com o prefixo set, seguido do nome do atributo com a letra inicial maiúscula.



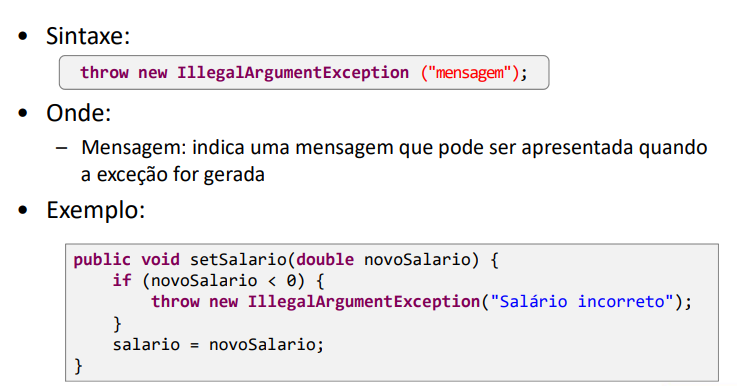
**This**

* “this” refere-se ao objeto corrente – o objeto na qual o método foi chamado.
* O principal motivo em usar a palavra this é quando há algum parâmetro de método que possui o mesmo nome de uma variável de instância.
  + Quando há dois identificadores com o mesmo nome, por padrão Java sempre reconhece que o identificador utilizado no comando é aquele com menor escopo.

**08 - Lançamento de Exceções**

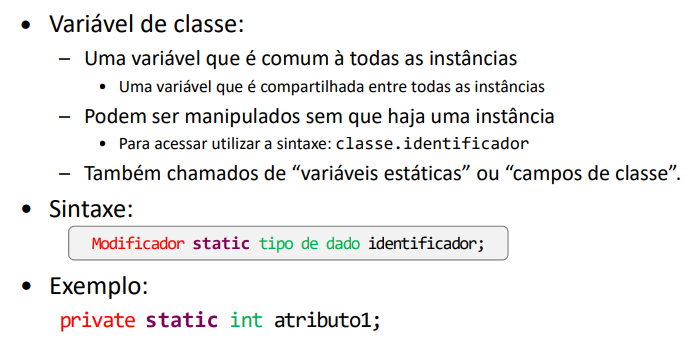
**Exceções**

* Uma exceção é um evento, que ocorre durante a execução do programa, que interrompe o fluxo normal de execução.
* Quando uma operação incorreta é identificada dentro de um método, o método pode criar um objeto de uma classe que caracteriza o erro e notificá-lo ao sistema.
  + Este objeto é denominado de “objeto de exceção” – O objeto de exceção contém informação sobre o erro.
  + Esta operação (criar objeto e notificar o sistema) é conhecida como “lançamento de exceção”.
* O efeito de uma exceção lançada é (por enquanto) abortar a execução do programa.
  + Não poderá existir comandos, no mesmo método, após a instrução que lança a exceção.



**09 - Membros de classe, exceções e construtores**

**Variáveis de Classe**

****

**Sobrecarga de Métodos**

* A linguagem Java suporta a sobrecarga de métodos, isto é, implementação de vários métodos com o mesmo nome.
* Os métodos devem ter assinaturas diferentes:
  + Métodos podem ter o mesmo nome se a lista de parâmetros for diferente.
  + O compilador não considera o tipo de retorno para diferenciar o método. Por isso, dois métodos com a mesma assinatura mas retornos distintos não podem ser implementados na mesma classe.
* Deve ser utilizado com moderação pois pode tornar o código menos legível.

**12 - Qualidade de Teste**

**•** Conceito de **“qualidade”** usado pela indústria: – “Um produto com qualidade é um produto que cumpre com sua especificação”

**•** Teste de software – Teste é a execução do software de maneira controlada para avaliar se ele se comporta ou não conforme o especificado.

Um plano de testes é um documento para registrar formalmente o planejamento

• O plano de testes pode conter: propósito, identificação, itens a serem testados, critérios de aceite, documentos produzidos (como logs, relatórios, etc.), ambiente a ser testado.

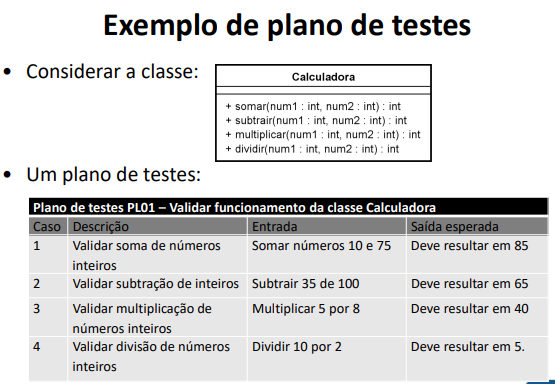
• Um plano de testes contém um conjunto de “casos de testes” – Um caso de teste possui uma descrição de um teste particular – Cada caso possui um conjunto de dados de entrada diferente.

Testes de Sistema

– Execução do sistema sob o ponto de vista do usuário, indo em busca de falhas em relação aos requisitos propostos.

Testes de Unidade (ou testes unitários) – Testam-se unidades, isto é, partes individuais (funcionalidades pequenas e específicas) de software.

• Em OOP, a “unidade” é o método



13 - **JUnit**

JUnit é um framework para execução de teste de unidade em Java

• Criado em 1997 por Erich Gamma e Kent Beck

• Hospedado no Source Forge

• Distribuído sob a licença IBM Common Public Lisence

• Modelo adotado por outras linguagens

• Possui três implementações (versão 3, versão 4 e versão 5)

• Importar as classes do framework de testes:

– import org.junit.\*

– import static org.junit.Assert.\*

Para cada “caso de teste”, criar um método e introduzir a anotação **@Test**

– O método deve ser público e não pode ter parâmetros

– O método não pode retornar dados (void)

– Dentro deste método, deve-se utilizar um comando assert, para validar uma situação

