PROGRAMAÇÃO ORIENTADA A OBJETOS

HERANÇA

AGENDA

1. Conceito de Herança.
2. Mecânica da Herança.
3. Objetivos da Herança.
4. Conversão de Tipos.
5. Como evitar a Herança.
6. Dicas para o Projeto de Herança.

O QUE É HERANÇA?

- **Encapsulamento** permite:
  - Escrever objetos definidos e independentes;
  - Permite que um objeto utilize outro;
- **Herança** permite:
  - Criar uma classe a partir da definição de outra classe.

O QUE É HERANÇA?

- Mecanismo que permite basear uma nova classe na definição de uma classe previamente existente;
- Quando uma classe **herda** de outra, todos os métodos e atributos que aparecem na interface da classe previamente existente apareceram automaticamente na interface da nova classe;
- A herança permite à classe que está sendo herdada redefinir qualquer comportamento que não goste.

O QUE É HERANÇA?

- Com identificar o tipo de relacionamento?
  - "É um": Herança
  - "Tem um": Associação
  - "Usa um": Dependência

O QUE É HERANÇA?

- **"É um":** descreve o relacionamento em que a classe que esta herdando é do mesmo tipo da classe herdada;
- **"Tem um":** descreve o relacionamento em que uma classe contém uma instância de outra classe;
- **"Usa um":** descreve o relacionamento em que uma classe dependente de outra classe.

HERANÇA

- No Java, a palavra-chave ***extends*** indica que está sendo criada uma nova classe que deriva de uma classe existente;
- A classe existente é chamada de **superclasse**, **classe base** ou **classe progenitora**;
- A nova classe é chamada de **subclasse**, **classe derivada** ou **classe filha**;
- As **subclasses** tem mais funcionalidades que suas **superclasses**.

HERANÇA

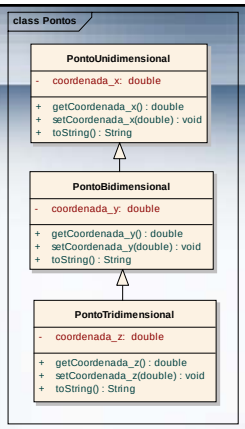
- A palavra-chave ***super*** sempre se refere a uma superclasse;
- Se o construtor da subclasse não chamar um construtor da superclasse explicitamente, então a superclasse usa seu **construtor padrão** (sem argumentos);
- Se a superclasse não tiver construtor padrão e o construtor da subclasse não chamar nenhum outro construtor da superclasse explicitamente, então o compilador Java vai informar um **erro**.

MECÂNICA DA HERANÇA

- Quando uma classe herda de outra, ela herda implementação, comportamentos e atributos;
- Uma classe pode ter três tipos de métodos e atributos:
  - **Sobrepostos**: a nova classe herda o método ou atributo, mas define uma nova "definição" para eles;
  - **Novos**: A nova classe adiciona um método ou atributo completamente novo;
  - **Recursivos**: a nova classe simplesmente herda um método ou atributo da progenitora;

SOBREPOSIÇÃO DE MÉTODOS

- É o processo de uma filha pegar um método que aparece na progenitora e reescrevê-lo para mudar o seu comportamento;
- Também conhecido como:
  - **Sobrescrita;**
  - **Overriding;**



SOBREPOSIÇÃO DE MÉTODOS E ATRIBUTOS

- Neste caso ocorre a sobreposição do método **toString()**;
- Diz-se que a subclasse **sobrescreve** a método definido pela superclasse;
- Chamamos este processo de **sobrescrita**.

MÉTODOS E ATRIBUTOS NOVOS

- É um método ou atributo que aparece na filha, mas não aparece na progenitora:
- Atributos novos:
  - **coordenada_y** em PontoBidimensional.
  - **coordenada_z** em PontoTridimensional.
- Métodos novos:
  - **getCoordenada_y** em PontoBidimensional.
  - **setCoordenada_y** em PontoBidimensional.
  - **getCoordenada_z** em PontoTridimensional.
  - **setCoordenada_z** em PontoTridimensional.

MÉTODOS E ATRIBUTOS RECURSIVOS

- É um método ou atributo que aparece na progenitora mas não aparece na filha;
- Atributos recursivos:
  - **coordenada_x** em PontoUnidimensional.
- Métodos recursivos:
  - **getCoordenada_x** em PontoUnidimensional.
  - **setCoordenada_x** em PontoUnidimensional.

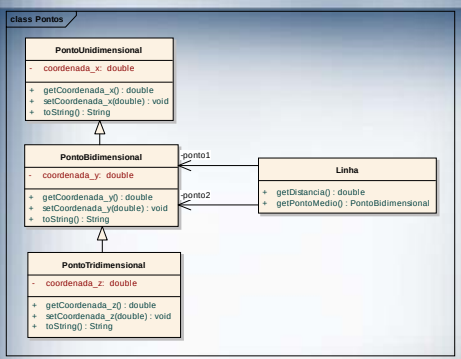
OBJETIVOS DA HERANÇA

1. Reutilização de implementação:
   - Obtida através de atributos e métodos recursivos;
2. Diferença:
   - Obtida através de atributos e métodos sobrescritos e novos.
3. Substituição de tipo:
   - Obtida através da capacidade utilizar subtipos no lugar de supertipos.

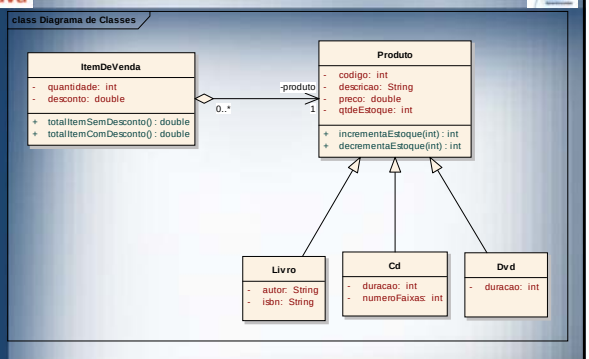
HERANÇA PARA SUBSTITUIÇÃO DE TIPO

- Permite que se descreva relacionamentos com capacidade de substituição;
- A capacidade de substituição aumenta a capacidade de reutilização.

HERANÇA PARA SUBSTITUIÇÃO DE TIPO



HERANÇA



HIERARQUIA DE HERANÇAS

- A herança não está limitada a derivar apenas uma camada de classes;
- A coleção de todas as classes que derivam de um ancestral em comum é chamada de **hierarquia de heranças**;
- O caminho de uma classe particular até o seu ancestral na hierarquia das heranças é chamado de **seqüência de heranças** (ou descendência).

CHAMADAS AO SUPER

- Chamadas ao ***super*** em **construtores**:
  - O construtor de uma subclasse sempre deve invocar o construtor da sua superclasse como sua primeira instrução;
  - Se o código fonte não incluir esta chamada, o Java inserirá a chamada automaticamente.

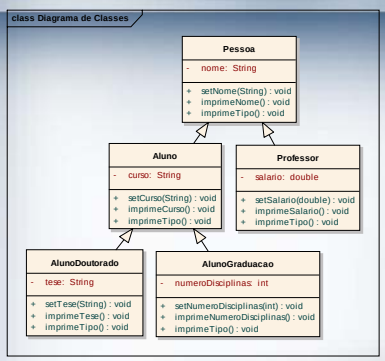
CHAMADAS AO SUPER

- Chamadas ao super em **métodos**:
  - Ao contrário das chamadas super nos construtores, o nome da superclasse é declarado explicitamente;
  - Ao contrário à regra para chamadas super nos construtores, a chamada nos métodos pode ocorrer em qualquer lugar dentro desse método;
  - Ao contrário às chamadas super nos construtores, nenhuma chamada super automática é gerada e nenhuma chamada super é requerida.

SUBCLASSES E SUBTIPOS

- O tipo definido por uma definição de **subclasse** é um **subtipo** do tipo definido pela superclasse correspondente;
- Variáveis podem armazenar objetos do seu tipo declarado ou de qualquer subtipo de seu tipo declarado;
- Os objetos de **subtipo** podem ser utilizados onde quer que os objetos de um **supertipo** sejam esperados.

SUBCLASSES E SUBTIPOS



CONVERSÃO DE TIPOS

- Embora um objeto de **subclasse** seja um objeto de **superclasse**, os dois objetos são diferentes;
- Objetos de **subclasses** podem ser tratados como se fossem objetos de **superclasses**;
- Entretanto, o oposto não é verdadeiro:
  - A subclasse pode ter membros adicionais.

CONVERSÃO DE TIPOS

- A conversão de um tipo de referência em outro tipo de referência é chamado de **casting**;
- A conversão de tipos é permitida apenas dentro em um relacionamento superclasse (supertipo) / subclasse (subtipo);
- Temos dois tipos de **casting**:
  - Uppercasting (widening)
  - Downcasting (narrowing)

CONVERSÃO DE TIPOS

- **Uppercasting**
  - Significa converter uma referência subtipo para uma referência supertipo;
  - Sempre é permitida;
  - Realizada implicitamente.

CONVERSÃO DE TIPOS

- **Downcasting**
  - Significa converter uma referência supertipo em uma referência subtipo;
  - Deve ser realizada explicitamente;
  - É realizada em tempo de execução.

COMO EVITAR HERANÇA E SOBRESCRITA

- As classes que não podem ser **classes-mãe** são chamada de **classes finais**;
- Usa-se o modificador ***final*** na definição da classe para indicar isto;

COMO EVITAR HERANÇA E SOBRESCRITA

- Pode-se também fazer ***final*** um método específico de uma classe;
- Nenhuma subclasse poderá sobrepor ou substituir esse método;
- Todos os métodos de uma classe final são automaticamente “finais”.

RECOMENDAÇÕES DE PROJETO DE HERANÇA

1. Coloque operações e campos comuns na superclasse;
2. Use herança para modelar uma relação de “é um tipo de”;
3. Não use herança a menos que todos os métodos herdados façam sentido.
4. Use polimorfismo, não informação de tipo.