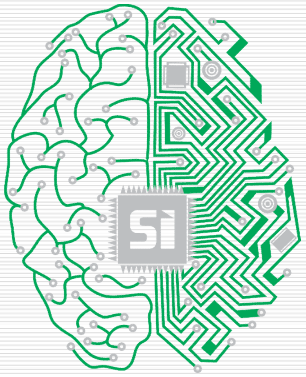
Herança de Classes

HERANÇA



1

Herança de Classes

- Sob um conceito genérico, entende-se Herança como sendo a extensão dos bens (ou outra coisa) de uma determinada pessoa/instituição para os seus “Herdeiros”.
- Por exemplo, um Pai divide suas posses aos seus Herdeiros imediatos.
- Ou seja, o que pertence ao Pai, pertence agora aos Filhos.

2

Herança de Classes

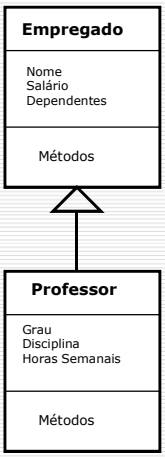
•Herança é um tipo de relacionamento que pode existir entre classes.
•Pode-se empregar o relacionamento de Herança entre classes quando uma classe for "***um tipo de***" outra classe.
•Costuma-se interpretar o relacionamento de herança entre duas classes como: "***é um tipo de***" ou "***é um***". Exemplo:

Professor "***é um tipo de***" Empregado ou
Professor "***é um***" Empregado.

3

Herança de Classes

Notação UML



4

Herança de Classes

•No relacionamento de Herança chama-se "**superclasse**" ou "**classe base**" a classe genérica e "**subclasse**" ou "**classe derivada**" a classe especializada.
•A maior característica desse tipo de relacionamento é como o próprio nome diz, a herança de métodos e atributos da superclasse pela subclasse.

5

Herança de Classes

•Portanto, no exemplo acima, a subclasse Professor herda todos os atributos e métodos da superclasse Empregado.
•Métodos **construtores NÃO** são herdados.
•Pode-se dizer que a subclasse Professor é uma **extensão** da superclasse Empregado.

6

Herança de Classes

Objetivo da Herança

•O objetivo principal da herança é a **reutilização de software**, já que novas classes são criadas a partir de outras já existentes, herdando seus atributos e métodos.

• A classe que herda ou estende outra classe tem a capacidade de ter novos atributos e métodos e também pode modificar, sobrescrever (*override*) os métodos herdados.

7

Herança de Classes

Objetivo da Herança

•A reutilização de software economiza tempo de desenvolvimento e manutenção de programas.

8

Herança de Classes

Outras Características

•Podem existir vários níveis de relacionamento de herança entre classes.

•Por exemplo: uma classe **Professor** herda de **Empregado** que herda de **Pessoa.**

•Chama-se **superclasse direta** de uma subclasse aquela que ela herda explicitamente e **superclasse indireta** é aquela herdada de dois ou mais níveis acima da hierarquia.

9

Herança de Classes

•Ao se instanciar um objeto da subclasse, pode-se "enxergar" ou acessar diretamente os membros com qualificador de acesso *public* ou *protected* da superclasse como parte da subclasse, por causa da herança.

10

Herança de Classes

•Os membros com modificador **protected** também podem ser acessados diretamente da subclasse por causa da herança.

•Um problema é que uma subclasse pode herdar métodos que ela não necessita ou não deveria ter.

• Quando um método da superclasse é impróprio para uma subclasse, ele pode ser **sobrescrito** (*override*) com uma implementação própria.

11

Herança de Classes

Construtores em Subclasses

•Quando um objeto de uma subclasse é instanciado, o construtor da superclasse deve ser chamado para inicializar os atributos superclasse.

•O construtor da superclasse deve ser chamado no construtor da subclasse na primeira linha deste método.

•A chamada deve ser explícita através da referência **super( )** .

12

Herança de Classes

Membros Protected

•Os membros **public** são acessíveis em qualquer classe faça referência a essa superclasse.

•Os membros **private** somente são acessíveis pelos métodos dessa superclasse. Isso somente acontece através de métodos (get e set).

•Os membros **protected** são um nível intermediário entre public e private, podendo ser acessados pela classe ou suas herdeiras.

13

Herança de Classes

Superclasses Abstratas e Classes Concretas

•Quando se pensa em uma classe, se imagina que os objetos dessa classe sempre serão instanciados.

•Entretanto, existem situações em que é útil definir classes para as quais ***nunca irá se querer instanciar objetos dessa classe***. Essas classes são chamadas **classes abstratas.**

14

Herança de Classes

•Como essas classes somente tem utilidade quando utilizadas em heranças como superclasse, elas são chamadas de **superclasses abstratas.**
•Nenhum objeto de uma superclasse abstrata pode ser instanciado.
•Tem o propósito de fornecer uma superclasse apropriada da qual as outras classes (subclasses) podem herdar atributos e métodos.
•As classes que permitem a instanciação de objetos são chamadas de **classes concretas.**

15

Herança de Classes

super e this
•Quando trabalhamos com herança, manipulamos membros que nem sempre estão na classe que estamos escrevendo.
•Não é incomum acessarmos métodos que estão escritos na **superclasse**.
•Não é incomum também, reescrevermos métodos nas classes derivadas(*override*).

16

Herança de Classes

super e this

•Para que não haja confusão quanto a "localização" dos membros, são utilizados os identificadores **super** e **this**.

•**super**: se usado de maneira isolada, **super()** irá invocar o construtor da classe base. Se usado com separador "." pode acessar qualquer membro *public* ou *protected* da classe base. Exemplo:

•**super.setNome("João");** `irá invocar o método setNome escrito na superclasse.`

17

Herança de Classes

super e this

•**this**: Usado com separador "." para identificar que o membro a ser utilizado está escrito na própria classe. Exemplo:

•**this.setCurso("Sistemas de Informação");** `irá invocar o método setCurso escrito na própria classe derivada.`

18