PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

Prof. Emanoel Barreiros

Herança

- Funciona da mesma forma que pensamos na “vida real”
- Algo pode ser deixado para você por herança (uma fortuna, por exemplo)
- Outra analogia seria o código DNA de cada um. Se você herda de uma classe, você vai ter também todas as propriedades da classe de quem você herdou
- A classe de quem herdamos é chamada de superclasse
- A classe que está herdando é chamada de subclasse

Herança

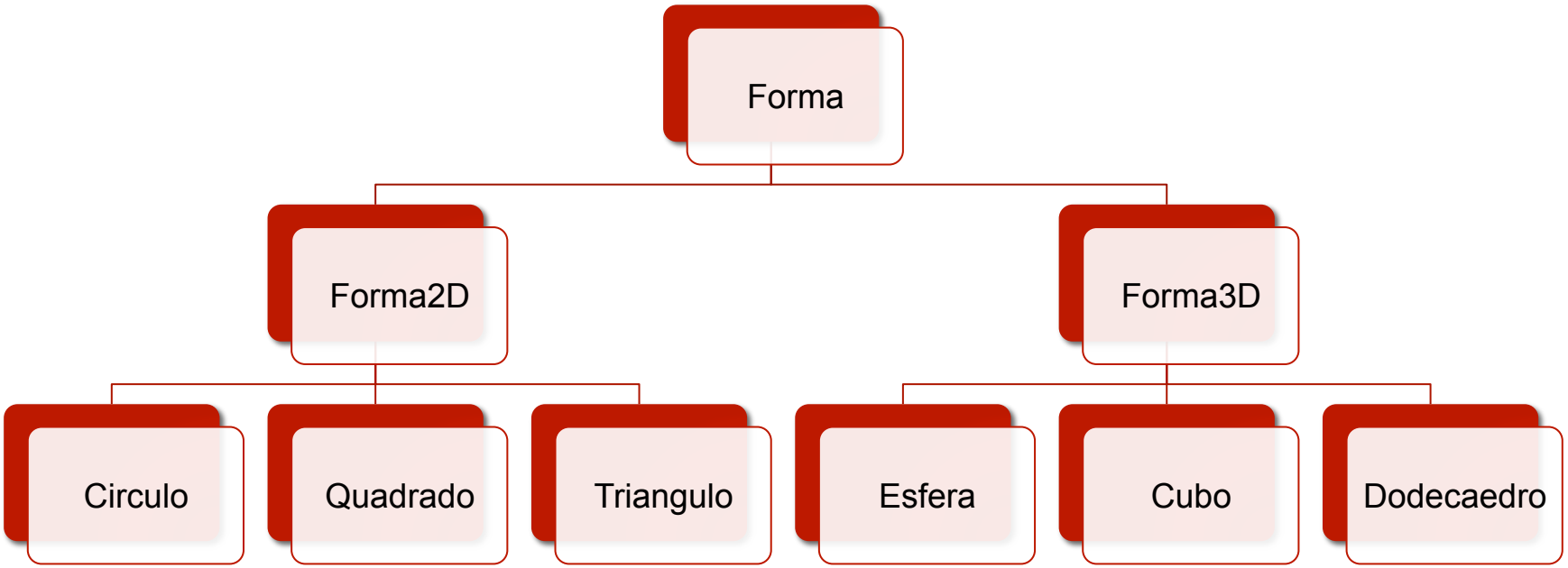
- Você pode criar uma hierarquia de herança tão longa quanto você ache necessário, mas não exagere
- Em Java, todos os objetos herdam, explícita ou implicitamente da classe java.lang.Object
- Java permite apenas herança simples, ou seja, podemos herdar de apenas uma classe por vez

Superclasses e subclasses

- Em geral, um objeto de uma classe *é um* objeto de outra classe também (quando há herança)
- Superclasses tendem a ser mais genéricas, enquanto subclasses tendem a ser mais específicas
- Exemplos de classes e subclasses:

Superclasse	Subclasses
Estudante	EstudanteGraduacao, EstudantePos
Forma	Circulo, Triangulo, Retangulo
Empregado	Docente, Administracao
Conta	ContaPoupanca, ContaCorrente

Herança das formas



Membros *protected*

- Atributos ou métodos declarados como private são visíveis apenas dentro da classe onde são declarados
- Atributos ou métodos declarados como public são visíveis por outras classes em qualquer lugar do sistema
- Atributos ou métodos declarados como protected são visíveis dentro da classe onde são definidos, por suas subclasses e outras classes quaisquer no mesmo pacote

Relacionamento entre superclasses e subclasses

- Exemplo no código!

Métodos importantes herdados de Object

- equals
  - O método equals é utilizado para determinar se um objeto é igual a outro. Para tipos por referência, não podemos usar o ==, precisamos implementar o equals para todas as classes nas quais queremos utilizá-lo. Object fornece uma implementação padrão, mas utilizá-la pode representar um erro de lógica, pois ela não representa a equalidade para todos os objetos.
- toString
  - É usado como uma representação String daquele objeto. Object fornece uma implementação padrão, mas ela não é muito útil.

CENAS DOS PRÓXIMOS CAPÍTULOS

Polimorfismo

Introdução ao Polimorfismo

- Suponha que desenvolvemos um programa que simule o movimento de diferentes animais
- Este sistema é usado para estudar peixes, sapos e pássaros. Para tal, você cria as classes Peixe, Sapo e Passaro
- Todas essas classes herdam da classe Animal, que possui um método mover() e atributos que mantém a posição de cada animal (x, y)
- Nosso sistema precisa manter um array de objetos do tipo Animal, e a cada segundo, executar o método mover() de cada um

Introdução ao Polimorfismo

- Um peixe pode nadar 1 metro, enquanto que um pássaro pode voar 5 metros e um salpo pode saltar 0,5 metros
- Cada objeto sabe como mover-se e atualizar a sua posição, mas o nosso sistema só deve chamar o método mover() de cada um
- O método mover é polimórfico, pois para cada objeto ele possui uma implementação diferente, inerente a cada tipo que a implementa
- Com polimorfismo podemos projetar e implementar sistemas facilmente extensíveis
  - Ex: se desejarmos criar uma nova classe Formiga, as outras classes de animais permanecem intocadas