Linguagem de Programação II

Prof. Marc Antonio Vieira de Queiroz

Ciência da Computação - UNIFIL
marc.queiroz@unifil.br

20 de agosto de 2015

Roteiro da aula I

1. 8.3.2 Sobreposição e ocultação

8.3.2 Sobreposição e ocultação I

Vimos nos exemplos mostrados neste capítulo que quando criamos classes estendidas através do mecanismo de herança, podemos definir novos campos e métodos que são específicos da classe herdeira (como visto nas listagens 8.7 - Classe Pessoa, 8.8 Classe Funcionário e 8.9 - Chefe de departamento) mas também redeclarar campos e métodos que tem a mesma assinatura de métodos declarados em classes ancestrais, mas com funcionalidade diferente (como nas listagens 8.10 - Automovel, 8.11 - Automovel Basico e 8.12 - Automovel de Luxo).

A declaração de métodos com a mesma assinatura que métodos de classes ancestrais chama-se sobreposição ou superposição. A razão de sobrepormos métodos é que métodos de classes herdeiras geralmente executam tarefas adicionais que os mesmos métodos das classes ancestrais não executam.

8.3.2 Sobreposição e ocultação II

O exemplo mais óbvio da necessidade de sobreposição é dado por métodos que fazem a impressão de campos: a classe ancestral somente imprime os campos que contém, então se adicionamos campos à classe descendente, o método de impressão desta não poderá ser o herdado, que não imprimirá os campos adicionais.

A declaração de campos em uma classe descendente com o mesmo nome de campos declarados na classe ancestral chama-se ocultação. Ao contrário da sobreposição de métodos, que é bastante útil e comum em classes herdeiras, a ocultação de campos não oferece muitas vantagens, e as poucas oferecidas podem facilmente ser implementadas através de métodos que retornam valores e são superpostos de acordo com a necessidade.

8.3.2 Sobreposição e ocultação III

As principais regras de sobreposição de métodos e ocultação de campos, e regras de uso de modificadores de acesso em classes herdadas são:

A sobreposição de um método em uma subclasse não elimina o acesso ao método de mesma assinatura na classe ancestral - este pode ser acessado, de dentro da classe herdeira, com a palavra-chave super, contanto que não tenha sido declarado como private. Veja, como exemplo, os vários métodos toString de classes herdeiras mostradas neste capítulo.

Métodos declarados em uma subclasse com o mesmo nome mas assinaturas diferentes (por exemplo, número de argumentos diferentes) dos métodos da superclasse não sobrepõem estes métodos. Por exemplo, se a classe Cor conter o método não-privado inicializa que recebe três valores do tipo float como argumentos, e a

8.3.2 Sobreposição e ocultação IV

classe CorCMYK conter o método com mesmo nome mas que recebe quatro valores do tipo float, esta classe poderá acessar diretamente o método inicializa da classe ancestral pois as assinaturas dos métodos de mesmo nome serão diferentes - em outras palavras, o método inicializa será herdado como qualquer outro.

Métodos podem ser sobrepostos com diferentes modificadores de acesso, contanto que os métodos sobrepostos tenham modificadores de acesso menos restritivos. Em outras palavras, podemos declarar um método na superclasse com o modificador de acesso private e sobrepor este método em uma subclasse com o modificador de acesso public, mas não podemos fazer o contrário.

Métodos estáticos declarados em classes ancestrais não podem ser sobrepostos em classes descendentes, nem mesmo se não forem declarados como estáticos.

8.3.2 Sobreposição e ocultação V

Se um campo é declarado em uma superclasse e oculto em subclasses, e métodos que acessam este campo são herdados, estes métodos farão referência ao campo da classe onde foram declarados.

Um exemplo é visto nas classes que representam diferentes tipos de automóveis, que pode ser visto neste capítulo. O campo NÚMEROMÁXIMODEPRESTAÇÕES e o método quantasPrestações que retorna o valor deste campo, foram declarados na classe Automovel (listagem 8.10) e herdados diretamente pela classe AutomovelBasico (listagem 8.11). O campo é redeclarado (oculto) pela classe AutomovelDeLuxo (listagem 8.12), que redeclara também o método quantasPrestações. Se este método não fosse redeclarado, sobrepondo o da classe ancestral, seria herdado, mas retornaria o valor do campo declarado na superclasse.

8.3.2 Sobreposição e ocultação VI

Em outras palavras, métodos herdados não podem acessar campos declarados em subclasses. Qualquer método da classe herdeira pode chamar qualquer método da classe ancestral que tenha sido declarado como public, protected ou sem declaração explícita de modificador. Métodos declarados como private não são acessíveis diretamente, mas podem ser chamados indiretamente a partir de métodos que não sejam private.

Por exemplo, consideremos a classe ComputadorDeBordo, que encapsula os campos e métodos necessários para implementar um computador de bordo de automóveis. Esta classe poderia ter o método privado calculaQuilometragemRodada e o método público mostraEstatísticasDeConsumo, com o método mostraEstatísticasDeConsumo executando o método calculaQuilometragemRodada como parte de seu processamento.

8.3.2 Sobreposição e ocultação VII

Se a classe ComputadorDeBordoDeCaminhao for criada como sendo herdeira da classe ComputadorDeBordo, ela não poderá executar o método calculaQuilometragemRodada diretamente pois este foi declarado como sendo privado da classe ComputadorDeBordo, mas o método será executado indiretamente através da chamada ao método público herdado mostraEstatísticasDeConsumo.

Métodos declarados como final são herdados por subclasses, mas não podem ser sobrepostos (a não ser que a sua assinatura seja diferente). Por exemplo, a classe ChefeDeDepartamento (listagem 8.9) não pode declarar um método qualSalário pois este foi declarado como final na classe ancestral Funcionario (listagem 8.8).

As regras de acesso e herança envolvendo o modificador protected serão vistas com mais detalhes no capítulo 10.
Classes inteiras podem ser declaradas como finais, na forma final

8.3.2 Sobreposição e ocultação VIII

class .... Se uma classe é declarada como final, todos os seus métodos serão finais, mas não os seus campos. A declaração de uma classe como final efetivamente impede o mecanismo de herança - o compilador não compilará uma classe declarada como herdeira de uma classe final.