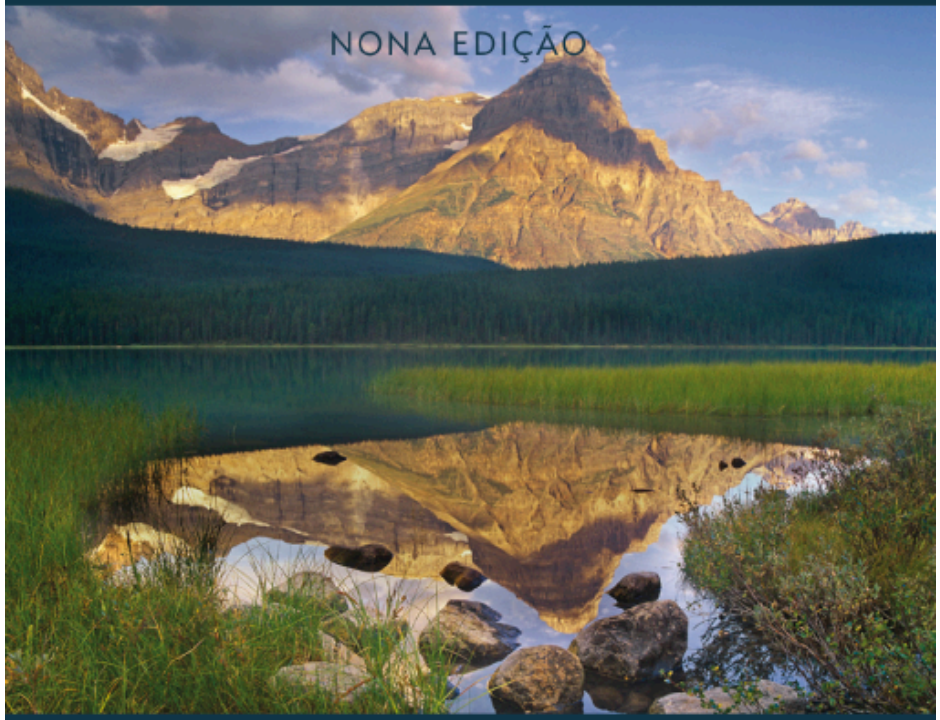
CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

NONA EDIÇÃO



ROBERT W. SEBESTA

bookman

Programação orientada a objeto

- Tipos de dados abstratos
- Herança
  - Herança é o tema central da programação orientada a a objetos e das linguagens que a suportam
- Polimorfismo

Herança

- Aumento de produtividade pode ocorrer com o reúso
  - Tipos de dados abstratos são difíceis de reusar − sempre precisam de mudanças
  - Definições de tipos de dados abstratos são todas independentes e no mesmo nível
- Herança permite novas classes definidas nos termos das já existentes

Conceitos de orientação a objetos

- Tipos de dados abstratos são geralmente chamados de *classes*
- Instâncias de classes são *objetos*
- Uma classe derivada por meio de herança de outra classe é uma *classe derivada* ou uma *subclasse*
- A classe da qual a nova classe é derivada é sua *classe pai* ou *superclasse*
- Subprogramas que definem as operações em objetos de uma classe são *métodos*

Conceitos de orientação a objetos (continuação)

- Chamadas aos métodos são *mensagens*
- A coleção completa de métodos de um objeto é chamada de *protocolo de mensagens* ou *interface de mensagens*
- Mensagens têm duas partes – um nome de método e o destino do objeto
- Se uma nova classe é uma subclasse de uma única classe pai, então o processo de derivação é chamado de herança simples

Conceitos de orientação a objetos (continuação)

- A herança pode ser complicada por controles de acesso às entidades encapsuladas
  - Uma classe pode esconder entidades de suas subclasses
  - Uma classe pode esconder entidades de seus clientes
  - Uma classe também pode ocultar entidades para seus clientes, mas permitir às suas subclasses vê-los
- Uma classe pode modificar um método herdado
  - O novo método *sobrescreve* o método herdado
  - Este, então, é chamado de *método sobrescrito*

Conceitos de orientação a objetos (continuação)

- Há dois tipos de variáveis em uma classe:
  - *Variáveis de classe*
  - *Variáveis de instância*
- Há dois tipos de métodos em uma classe:
  - *Métodos de classe*
  - *Métodos de instância*
- Herança simples × herança múltipla
- Uma desvantagem da herança como uma forma de aumentar a possibilidade de reúso é que ela cria dependências entre classes em uma hierarquia

Vinculação dinâmica

- Uma *variável polimórfica* pode ser definida em uma classe que é capaz de referenciar (ou apontar) os objetos da classe e objetos de qualquer dos seus descendentes
- Quando uma hierarquia de classe inclui as classes que sobrescrevem métodos e esses métodos são chamados por uma variável polimórfica, a ligação para o método correto será dinâmica

Conceitos de vinculação dinâmica

- Um *método abstrato* é um que não inclui uma definição (apenas define um protocolo)
- Uma *classe abstrata* inclui pelo menos um método virtual. Uma classe abstrata não pode ser instanciada

Questões de projeto

- A exclusividade dos objetos
- As subclasses são subtipos?
- Verificação de tipos e polimorfismo
- Herança simples e múltipla
- Alocação e liberação de objetos
- Vinculação estática e dinâmica
- Classes aninhadas
- Inicialização de objetos

A exclusividade dos objetos

- Tudo é um objeto
  - Vantagem - elegância e pureza
  - Desvantagem - operações lentas para objetos simples
- Adicionar objetos a um sistema de tipos completo
  - Vantagem - operações rápidas em objetos simples
  - Desvantagem - resulta em um sistema confuso (dois tipos de entidades)
- Incluir um estilo imperativo para tipos primitivos escalares, mas implementar todos os tipos estruturados como objetos
  - Vantagem - operações rápidas em objetos simples e um sistema de tipos relativamente pequeno
  - Desvantagem - ainda alguma confusão por causa dos dois sistemas de tipo

As subclasses são subtipos?

- Um relacionamento "é-um(a)" se mantém entre uma classe derivada e sua classe pai?
  - Se uma classe derivada é um(a) classe pai, então os objetos da classe derivada devem expor todos os membros que são expostos por objetos da classe pai
- Uma classe derivada é um subtipo se tiver um relacionamento "é-um(a)" com sua classe pai
  - Os métodos da subclasse que sobrescrevem métodos da classe pai devem ser compatíveis em relação ao tipo com seus métodos sobrescritos correspondentes

Verificação de tipos e polimorfismo

- O polimorfismo pode exigir uma verificação de tipo dinâmico de parâmetros e o valor de retorno
  - Verificação de tipos dinâmica custa tempo de execução e posterga a detecção de erros de tipo
- Se o método sobrescrevedor tiver o mesmo número de tipos de parâmetros e de retorno do que o método sobrescrito, a verificação pode ser estática

Herança simples e múltipla

- Herança múltipla permite uma nova classe herdar de duas ou mais classes
- Desvantagens de herança múltipla:
  - Complexidade de linguagem e implementação
  - Potencial ineficiência
- Vantagem:
  - Às vezes, é bastante conveniente

Alocação e liberação de objetos

- De onde são os objetos alocados?
  - Se eles se comportam como tipos de dados abstratos, então talvez eles possam ser alocados de qualquer lugar
    - Alocados da pilha de tempo de execução
    - Explicitamente criados no monte com um operador ou função, como `new`
  - Se eles são todos dinâmicos do monte, existe a vantagem de ter um método de criação e acesso uniforme por meio de ponteiros ou variáveis de referência
  - Se os objetos são dinâmicos da pilha, existe um problema relacionado aos subtipos
- A liberação é explícita ou implícita?

Vinculação estática e dinâmica

- Todas as vinculações de mensagens a métodos são dinâmicas?
  - Se nenhuma for, vai perder as vantagens de vinculação dinâmica
  - Se todos forem, é ineficiente

Classes aninhadas

- Se uma nova classe é necessária em apenas uma classe, não há razão para definir para que ele possa ser vista por outras classes
  - Pode a nova classe ser aninhada dentro da classe que a usa?
  - Em alguns casos, a nova classe está aninhada em um subprograma, em vez de em outra classe
- Questões de projeto:
  - Quais recursos da classe aninhadora são visíveis para a classe aninhada e vice-versa?

Inicialização de objetos

- Quando um objeto de uma subclasse é criado, a inicialização associada do membro herdado da classe pai é implícita ou o programador deve lidar explicitamente com ela?

Resumo

- A programação orientada a objetos envolve três conceitos fundamentais: tipos de dados abstratos, herança e vinculação dinâmica
- Questões de projeto: exclusividade de objetos, subclasses e subtipos, verificação de tipo e polimorfismo, herança simples e múltipla, vinculação dinâmica, liberação explícita ou implícita de objetos e classes aninhadas
- Smalltalk é uma linguagem orientada a objetos pura
- C++ tem dois sistemas de tipos (híbrida)
- Java não é uma linguagem híbrida como C++; suporta apenas programação orientada a objetos
- C# é baseada em C++ e Java
- Ruby é uma nova linguagem orientada a objetos pura
- JavaScript não é uma linguagem de programação orientada a objetos, mas fornece variações interessantes
- Implementar linguagens de programação orientada a objetos envolve novas estruturas de dados