Suporte para Programação Orientada a Objetos

Prof. Lucas Ismaily

Sumário

- Tipos abstratos de dados
- Conceitos de orientação a objetos
- Questões de projeto de LPs OO

Tipos abstratos de dados

- Abstração: omissão de detalhes que não são importantes em um dado contexto
  - Reduz a complexidade da programação
- Abstração de processos: subprogramas
- Abstração de dados: encapsulamento

Tipos abstratos de dados

- Encapsulamento
  - Agrupamento de subprogramas e dos dados que eles manipulam em módulos
  - Vantagens
    - Organização
    - Compilação separada
  - Para que existe reuso, é importante que a organização em módulo produza pouca dependência de outros módulos

Tipos abstratos de dados

- Encapsulamento
  - Ocultação da informação: disponibilizar apenas o que for importante para módulos clientes
  - Ex.: linguagem C
    - Conjunto de funções e dados relacionados podem ser colocados em arquivos separados
    - A compilação pode ser separada
    - As informações que devem ser visíveis para os módulos “clientes” são colocadas no arquivo de cabeçalho

Tipos abstratos de dados

- Encapsulamento
  - Ex.: tipo abstrato PILHA

    (independe da implementação)

```
cria(pilha) – aloca e inicializa
destroi(pilha) – libera memória
vazia(pilha) – testa se pilha vazia
empilha(pilha, elemento) – coloca o elemento no topo
                           da pilha
desempilha(pilha) – remove elemento do topo da pilha
topo(pilha) – retorna elemento no topo da pilha
```

Programação OO

- Muitas LPs orientadas a objetos (OO)
  - Algumas suportam programação procedural (ADA 5, C++)
  - Algumas suportam programação funcional (CLOS, Python)
  - LPs mais novas não suportam outros paradigmas, mas utilizam estruturas de LPs imperativas mais antigas (Java, C#)
  - Algumas são puramente OO (Smalltalk, Ruby)

Programação OO

- Principais características
  - Uso intenso da abstração de dados
  - Herança
    - Tema central da programação OO e das LPs que a suportam
    - Aumenta reuso
  - Polimorfismo

Herança

- Aumento de produtividade através do reuso
  - Todos os TADs são independentes e em um mesmo nível
  - Não é fácil criar novo TAD complementando um existente
- Permite definir novas classes em termos de classes existentes
- Resolve os problemas dos TADs:
  - Reuso de TADs após pequenas modificações
  - Definidas como uma hierarquia de classes

Conceitos de OO

- TADs são geralmente chamadas de classes
- Instâncias de classes são chamados objetos
- Uma classe que herda é uma classe derivada, ou subclasse
- Uma classe herdada por outra é uma classe pai, ou superclasse
- Subprogramas que definem operações sobre objetos são chamados de métodos

Conceitos de OO

- Chamadas de métodos são chamadas mensagens
- O conjunto de métodos de uma classe é chamado interface de mensagens
- Mensagens têm duas partes: nome do método e objeto de destino
- No caso mais simples, uma classe herda todos os métodos e dados da classe pai

Conceitos de OO

- Podemos ocultar informação através de controle de acesso
  - Pode ocultar dados/métodos das subclasses
  - Pode ocultar dados/métodos dos clientes
  - Pode ocultar do cliente, mas permitir acesso às subclasses
- A subclasse pode modificar métodos herdados
  - O novo sobrescreve o herdado

Conceitos de OO

- Dois tipos de variáveis em uma classe:
  - Variáveis de classe: uma por classe
  - Variáveis de instância: uma por objeto
- Dois tipos de métodos em uma classe:
  - Métodos de classe: mensagens para a classe
  - Métodos de instância: mensagens para o objeto
- Herança simples versus múltipla
- Desvantagem da herança: cria dependência entre classes, complicando manutenção

Vinculação Dinâmica

- Uma variável polimórfica é definida em uma classe, mas referenciar objetos de classes decendentes
- Quando o método da subclasse é sobrecarregado em uma variável polimórfica, a vinculação do método correto é dinâmica
- Facilita a extensão do sistema durante o desenvolvimento e manutenção

Vinculação Dinâmica

- Um método abstrato não inclui sua definição (corpo)
- Uma classe abstrata inclui pelo menos um método abstrato
- Uma classe abstrata não pode ser instanciada

Questões de projeto de LPs OO

- Exclusividade de objetos
- Subclasses são subtipos
- Verificação de tipos e polimorfismo
- Herança simples ou múltipla
- Alocação de desalocação de objetos
- Classes aninhadas

Exclusividade de Objetos

- Tudo é um objeto
  - Vantagem: elegância e pureza
  - Desvantagens: operação lenta de objetos simples (ex números)(tudo exige passagem de mensagens)
- Separar os tipos primitivos
  - Vantagem: eficiência
  - Desvantagens: confusão (tratamento diferenciado), combinação de primitivos com objetos exige “wrappers”

Subclasses são subtipos

- Existe uma relação “é um” entre um objeto da classe pai e um da classe filha
  - Neste caso podemos usar o objeto da classe filha no lugar do objeto da classe pai (mesmo comportamente)
- Todas as modificações na classe filha conservam as características da classe pai

Verificação de tipos e polimorfismo

- Polimorfismo pode requerer verificação dinâmica de tipo de parâmetros e retornos
  - Verificação dinâmica é custosa e atrasa a detecção de erros
- Se métodos sobrecarregados são restritos a ter o mesmo tipo de parâmetro e retorno, então a verificação pode ser estática

Herança simples e múltipla

- Herança múltipla permite que uma classe herde de duas ou mais classes
- Desvantagens da herança múltipla:
  - Complexidade (deixa de ser uma árvore..)
  - Vinculação dinâmica é um pouco mais lenta
- Vantagem:
  - s vezes é conveniente

Alocação e desalocação de objetos

- Onde os objetos são alocados
  - Como TADs
    - Alocado na pilha em tempo de execução
    - Pode ser criado explicitamente na heap (ex.: new)
  - Se são dinâmicos na heap, temos uniformidade na referência (facilita atribuição)
  - Se são dinâmicos na pilha, existe problema ao armazenar subtipos
- A desalocação é implícita ou explícita

Classes aninhadas

- Se uma classe é utilizada apenas dentro uma outra, ela não precisa ser visível às outras
  - Podemos declarar uma dentro da outra
  - Em alguns casos podemos querer definir dentro de um subprograma
- Outras questões:
  - O que deve ser visível à classe aninhada