

Disciplina: DIM0437 — Linguagens de Programação: Conceitos e Paradigmas

Docente: Umberto Souza da Costa

Discentes: Dogival Ferreira da Silva Junior

Felipe Cortez de Sá

Gabriel Sebastian von Conta

Phellipe Albert Volkmer

Vinícius Araújo Petch

Subproblema 8 — Abstrações

1 Problema

1.1 Produto do problema

Definição de abstrações de larga escala presentes na linguagem de programação a ser definida pelo grupo, considerando o contexto de aplicação dos programas nela escritos e características principais da mesma.

1.2 Questões

1. Qual o conceito de abstração?
2. Quais serão as formas de abstrações presentes na linguagem proposta:
 - (a) Tipos abstratos de dados?
 - (b) Classes de objetos?
 - (c) Pacotes?
 - (d) Módulos?
3. O que caracteriza a categorização de abstrações mostrada acima? Quais são as semelhanças e diferenças dos conceitos definidos na questão anterior?
4. Como os conceitos discutidos poderiam ser implementados em sua linguagem?
5. Além dessas características, como sua linguagem poderia incluir a noção de genericidade (para tipos, classes, pacotes ou módulos)?
6. Como sua linguagem lidará com construções de encapsulamento e com o encapsulamento de nomes?

2 Resoluções

1. Uma **abstração** é uma simplificação útil para o programador, uma vez que permite representar uma entidade, filtrando as informações e operações, aproveitando apenas o que é relevante.

Por exemplo, uma entidade **grupo de lpcp** possui os atributos gerais componentes e nota e trabalhos, mas também possui características únicas a certos grupos, como por exemplo **possui gordos**. Criamos então um objeto **G2** que precisa do atributo específico **possui gordos**, mas não **componentes**, **nota** e **trabalhos**, herdados da entidade **grupo de lpcp**.

2. (a) No caso de tipos abstratos de dados:

Um *tipo abstrato de dado* tem a representação do tipo dos objetos ocultas para as unidades de programa que utilizam esse tipo. Dessa forma as únicas operações possíveis nesses objetos são aquelas que estão na definição do objeto.

Além disso as operações do tipo e as operações definidas nos objetos desse tipo, que fornecem a interface desse tipo, são contidas em uma única unidade sintática.

Assim, a interface do tipo não depende da representação dos objetos nem da definição de suas operações. Além disso outras unidades de programa podem usar variáveis do tipo definido.

Na nossa linguagem, utilizaremos tipos abstratos de dados, presentes tanto no tipo float, quanto em tipos criados pelo usuário.

- (b) No caso de pacotes:

Um *pacote* é como a implementação de módulo é chamada na linguagem Ada. Nela, os cabeçalhos e corpos dos módulos podem ser separados.

Nós utilizaremos pacotes (packages). A ideia de pacotes será implementada semelhante à utilização de módulos em Ada, onde cabeçalho e corpo podem ser separados. Porém, na nossa implementação cabeçalho e corpo terão que estar no mesmo arquivo.

- (c) No caso de classes:

Uma *classe* é um *template* extensível para a criação de objetos. As classes podem ser instanciadas através de uma chamada para o método construtor, uma subrotina responsável pela inicialização de dados de objetos. As entidades declaradas como públicas podem ser acessadas através dessas instâncias. De acordo com Sebesta, esta maneira é mais limpa e direta que o encapsulamento em Ada.

Classes possuem membros de dados, que armazenam estado e informação, e membros de subrotina, implementações de comportamento, também chamados de métodos em linguagens orientadas a objeto. Além disso, membros de dados podem ser categorizados em membros de classe, independentes de instanciação e membros de instância, que podem assumir valores diferentes por cada objeto. Para organizar o código e aumentar a legibilidade, C++ permite separar a definição de uma classe e sua implementação.

A diferença entre as classes das linguagens orientadas a objetos e pacotes em Ada é que classes geram tipos.

3. Todas são encapsulamentos que permitem a abstração de dados e funções do código. No caso do tipo de dados, temos a noção de uma implementação de um grupo de funções, das quais o usuário só conhece aquelas que estão na definição do grupo. Classes de objetos são uma implementação em cima dessa noção criada a partir de TAD.

No caso de pacotes e módulos, ambos tem a função tanto de organização do código e separação da compilação. A diferença se encontra na implementação de ambos.

4. Em relação aos pacotes, como foi dito anteriormente, eles serão implementados no mesmo estilo da Ada, porém com cabeçalho e corpo no mesmo arquivo. Para chamar um pacote em um arquivo, será utilizado o comando **import <nome_do_arquivo>**

- Criação de pacotes - porém, códigos monolíticos não são muito problema

5. - Tipos e pacotes mais genéricos, tipos parametrizados
6. - Quais são as estruturas que permitem que você guarde e oculte - namespaces (encapsulamento de nomes)