POO: Classes, Objetos e TADs

Curso: Sistemas de Informação
Disciplina: POO

Prof. Fábio L. R. Cordeiro

SUMÁRIO

- Abstração de Dados
- Estruturas de Dados
- Estrutura Abstrata de Dados
- Tipos Abstratos de Dados (TADs)
  - Conceito, Condições e Exemplos
- Classes e Objetos

Motivação de Dados

- Abstração é o processo de identificar as características ou propriedades do problema no processo de modelagem.
- Através de um modelo abstrato, é possível selecionar as características mais relevantes na modelagem.
- Abstração é um exercício de raciocínio lógico.
- Exemplo de abstração: Objeto representando uma pessoa.

Estrutura Abstrata de Dados

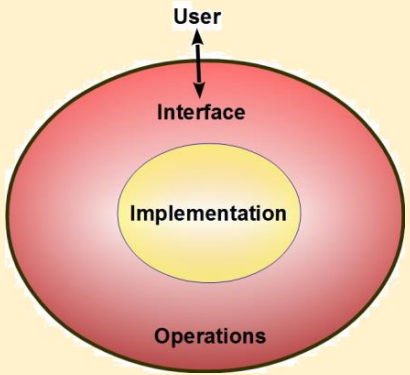
- Estrutura de dados abstrata, contém dados e métodos
- Dados estão ocultos dentro da estrutura e só podem ser acessados por meio dos métodos (encapsulamento)
- O nome da estrutura e sua interface são conhecidos, porem sua implementação é desconhecida (oculta).
- Métodos permanecem inalterados, mesmo que a implementação dos mesmos venha a ser alterada.

Tipos Abstratos de Dados (TADs)

- **Abstract Data Type (ADT)**

- TAD contém a definição da estrutura de dados abstrata
- TAD é um conjunto de estruturas de dados abstratas
- Uma particular estrutura de dados abstrata de um TAD é denominada de uma instância do TAD
- **TAD deve conter: - Nome - Dados - Métodos**
  Construtores – constrói e inicializa uma instância TAD
  Modificadores – Modifica o estado (dados) do TAD (sets)
  Acessores – Acessam e retornam estado do TAD (gets)
  Destrutores – Destrói uma instância do TAD

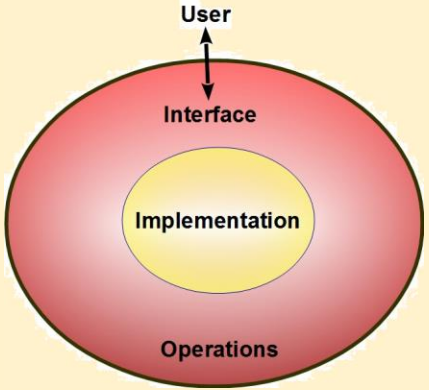
- A criação de uma classe é a definição de um TAD.



Tipos Abstratos de Dados (TADs)

- TAD deve ter as assinaturas das operações

- Assinatura contém as informações de:
  - Nome
  - Conjunto de entrada
    - Tipos, números e ordem dos parâmetros
  - Conjunto de saída
    - O tipo do valor de retorno



- Cada método deve conter
  - Pré-condições – aplicadas aos dados antes da utilização
  - Pós-condições – aplicadas depois da execução que indicam a modificação efetuadas

- Os tipos utilizados de TAD deve ser tipos válidos e conhecidos.

Círculo

Exemplo de um TAD (Círculo)

Nome: Circulo

Dados:

raio : número inteiro não negativo
xCentro : número inteiro que indica a coordenada x do ponto (x,y)
yCentro : número inteiro que indica a coordenada y do ponto (x,y)

Métodos:

Construtor

Valores Iniciais : O raio do círculo
Processo : Cria um círculo de raio informado e origem no ponto (0,0)

Área

Entrada : Nenhuma
Pré-condições : Nenhuma
Processo : Calcula área do círculo
Saída : A área do círculo
Pós-condições : Nenhuma

Escala

Entrada : Um número inteiro indicando o fator de escala
Pré-condições : O fator de escala deve ser maior que zero
Processo : Multiplica o raio pelo fator de escala e redesenha o círculo
Saída : Nenhuma
Pós-condições : O Valor do raio será o resultado da multiplicação do raio pelo fator de escala

TAD

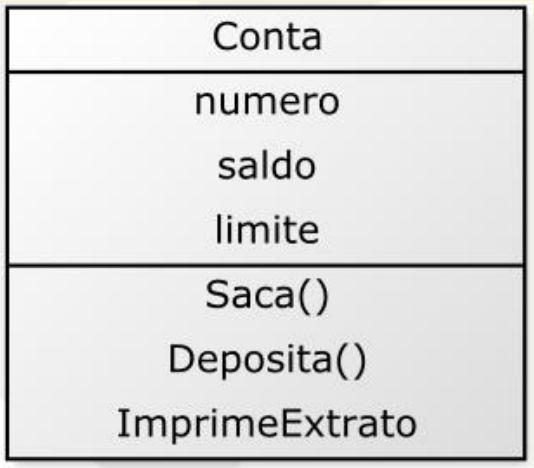
Implementação de Classes

- Nas linguagens orientadas a objetos (JAVA e C#) um ADT por uma classe, onde as operações são implementadas como métodos desta classe.
- Cada instância de uma classe é um objeto, criada a partir do operador new

Exemplo implementação da Classe conta:

```
class Conta {
    public double saldo;
    public double limite;
    public int numero;
}
```

Modelo Classe Conta em UML



Instanciando objetos da classe

- Objetos são criados a partir do ato de instanciar uma classe.
- É reservado um espaço (referência) na memória para aquele TAD implementado na Classe, feito de forma automática.
- Teste a classe instancie o objeto da classe Conta na Classe Principal.

```
1  using System;
2  namespace AulaTAD {
3      class MainClass {
4          public static void Main (string[] args) {
5              Conta conta = new Conta ();
6              conta.numero = 1;
7              conta.saldo = 1000.00;
8              conta.limite = 200.00;
9              Console.WriteLine ("Detalhes da conta nº: " + conta.numero);
10             Console.WriteLine ("Saldo da conta: " + conta.saldo);
11             Console.WriteLine ("Limite da conta é : " + conta.limite);
12         }
13     }
14 }
```

Estrutura Completa de uma Classe

- **modificador:** Possibilita alterar a forma como o método se comporta
  Ex: static; abstract; final; native; synchronized)
- **retorno:** Tipo de retorno do método, caso não exista, deve ser usado **void**.
  Pode ser básico (primitivo), um objeto da linguagem ou um Tipo Classe criada.
  Um pouco além:
  - **throws Exceptions** – Tipo de exceção que pode ser disparada pelo método
  - **Pacotes (Package ou Pastas)** – um conjunto de classes e interfaces, relacionadas entre si, normalmente ligadas a composição de bibliotecas. Garante o princípio da **Unicidade**, pois permite utilizar classes com mesmo nome em pacotes diferentes. Seguem a convenção: **com.company.package**

- Obs: Toda classe pertence a um pacote, quando não explicito, pertence ao **Default Package**.

Exercício de Modelagem

- Sua equipe de desenvolvimento foi alocada para participar de um Sistema de Gerenciamento de Biblioteca, trabalhando no planejamento, modelagem e implementação de um TAD que represente os Livros no sistema.
A Estrutura de dados inicialmente é:

```
struct livro {
    string titulo;
    string autor;
    string genero;
    int ano;
}
```

- As funções que devem ser exportadas pela interface TAD são:

a) Criar Livro que recebe por parâmetro o título, autor, gênero e ano de publicação e retorna um objeto do tipo Livro.
b) Quatro funções de manutenção básicas de registros do TAD livros:
Obter Autor, Título, Gênero e Ano.
c) Duas funções para verificar a Estilo literário, sendo que estás deverão receber um Objeto do Tipo Livro e retornar um Boleano com o resultado, sendo:
Modernismo de 1930 a 1945 e Barroco de 1601 a 1768.

- Deverá ser escrito o TAD com todos os elementos, seguindo o modelo visto em aula, seguindo boas práticas de nomenclatura e padrão CamelCase.

Continuação: Parte prática – mesmo grupo, em C# entregar via SGA

Exercício de Modelagem

a) Implemente o TAD Livro em uma classe com a definição de todas as funções definidas e suas implementações.

b) Escreva no módulo principal (Program.cs) um programa que inicialize um vetor do tipo Livros e usando as funções definidas na "interface" insira os registros:

"O Universo numa casca de Noz", "Stephen Hawking", "Física", 2001
"Cem Anos de Solidão", "Gabriel Garcia Matos", "Romance", 1967
"Ariana, a Mulher", "Vinicius de Morais", "Poesia", 1936
"Prosopopeia", "Bento Teixeira", "Poema", 1601
"O guia do mochileiro das galáxias", "Douglas Adams", "Ficção", 1981

c) No TAD Livro, escreva um método para ordenar o vetor de livros em ordem crescente, por ano de publicação.

d) Escreva um função que seja capaz de buscar pela existência de alguma obra de determinado autor que seja passado por parâmetro.

e) Escreva um método que seja capaz de verificar a ocorrência de um determinado gênero a quantidade de vez que ele foi encontrado.

Referências

CAELUM. C# e Orientação a Objetos. - Treinamentos. Caelum Ensino e Inovação, 180 p. 2019 - Disponível em: <https://www.caelum.com.br/>

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. Algoritmos: lógica para desenvolvimento de programação de computadores. 26. ed. rev. São Paulo: Érica, 2012. 328 p. ISBN 9788536502212 (Disponível no Acervo).

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos, c2010. xvi, 302 p. ISBN 9788521617501 (Disponível no Acervo).