Curso de Extensão

Tecnologias Microsoft

UNICAMP

INF-0990

Programação em C#
Aula 2

Prof. Dr. Ricardo Ribeiro Gudwin

gudwin@unicamp.br

03 de Setembro de 2022

Programação Estruturada x Programação Orientada a Objetos

Programação Estruturada x Orientada a Objetos

- Engenharia de Software
  - Conjunto de boas práticas de programação que tornam o desenvolvimento de software mais produtivo e eficiente
  - **Modularidade**: Dividir o programa em módulos mais simples, de modo recursivo e incremental
    - Módulos mais simples tornam sua compreensão mais simples, evitando erros e facilitando o processo de manutenção e correção de defeitos
    - Design do software torna-se um processo de refinamento sucessivo, onde o detalhamento vai sendo feito à medida em que aprofunda-se a compreensão de como deve ser o funcionamento de cada módulo
    - Facilita o reuso, pois módulos utilizados em programas anteriores podem ser reutilizados em novos programas com funcionalidades equivalentes
- Programação Estruturada
  - Muito popular, a partir da década de 50
  - Principais Técnicas:
    - Re-estruturação do código visando o uso de estruturas de fluxo de controle de seleção e repetição, ao invés de *go-tos*
    - Estruturas de Blocos e Subrotinas

Programação Estruturada x Orientada a Objetos

- Programação Estruturada
  - Dados Globais x Dados Locais
    - Em programas mais complexos, o número de variáveis (dados) pode ser bastante grande
    - Variáveis com o mesmo nome, mas com dados (e as vezes tipos) distintos podem gerar confusão e mal uso dos dados
    - Ao invés de **dados globais**, disponíveis para uso em qualquer lugar, torna-se mais conveniente usar **dados locais,** que têm validade somente dentro de um módulo (ou subrotina)
    - Quando o programa encontra-se distribuído em diversos arquivos, o uso de dados com validade global demandam um esforço adicional, para serem conhecidos em outros arquivos, onde não foram originalmente definidos
  - Ocultamento de Informações (Information Hiding)
    - **Variáveis locais:** variáveis com nome e validade somente dentro do escopo do módulo
    - Torna um programa mais compreensível e menos suscetível a erros
    - **Modos de acesso**: público ou privado

Programação Estruturada x Orientada a Objetos

- Programação Estruturada
  - Abstração
    - Subrotinas complexas podem ser decompostas em diversas subrotinas mais simples
      - → Árvore de Dependência entre subrotinas
      - → Organização de um grande número de subrotinas, orientada ao reuso pode se tornar uma tarefa bastante complexa
        - Como escolher as subrotinas que devem ser copiadas, quando se deseja reutilizar partes de um programa ?
  - Acoplamento
    - Dependência que uma sub-rotina tem com outras subrotinas, quando se deseja reutilizá-las
    - Acoplamento pode ser circular, em algumas situações, causando um aumento de complexidade na compreensão do programa
    - Dependência de um certo número de variáveis globais que precisam existir e ter seus valores definidos para que as sub-rotinas funcionem adequadamente
  - Coesão
    - Algumas subrotinas dizem respeito a um mesmo tema, e existe grande chance de que possam ser reutilizadas em conjunto
    - Da mesma forma, alguns dados podem ser necessários, simultaneamente, para o funcionamento de uma subrotina
  - Desejável: **Baixo Acoplamento** e **Alta Coesão**

Programação Estruturada x Orientada a Objetos

- Encapsulamento de Dados
  - Tipos de Dados Abstratos
    - Composição de diversas variáveis em uma única variável composta
- Criando Módulos de Alta Coesão
  - Definindo subrotinas tematicamente relacionadas em um mesmo arquivo
    - Melhora a possibilidade de reutilização
  - Utilização de tipos de dados abstratos para reunir variáveis que precisam ser acessadas simultaneamente para que uma dada funcionalidade seja implementada
- De Módulos para Objetos
  - Módulos → Módulos de Alta Coesão → Tipos de Dados Abstatos → Ocultamento de Informação → Objetos
  - Objetos: Instância de um tipo de dados abstrato (encapsulamento de dados) que ao mesmo tempo proporciona encapsulamento de procedimentos, ocultamento de informações e um mecanismo de herança, para promover sua reutilização

Programação Estruturada x Orientada a Objetos

- Herança
  - Novos tipos de dados podem ser definidos por meio da extensão de tipos previamente definidos, ao invés de uma construção do zero
- Classe
  - Tipo de dados abstrato que suporta herança
  - ***Fields*** (ou campos) são variáveis encapsuladas dentro da classe
    - Fields podem ser públicos ou privados
      - → Fields públicos podem ser acessados a partir de uma instância da classe
      - → Fiels privados só podem ser acessados internamente pela própria classe
  - **Métodos** são procedimentos (ou subrotinas) encapsuladas dentro da classe, e têm acesso privilegiado aos fields da classe
    - Métodos podem ser públicos ou privados
      - → Métodos públicos podem ser acessados a partir de uma instância da classe
      - → Métodos privados só podem ser acessados internamente pela própria classe
- Objeto
  - É uma instância de uma classe
  - Funciona como um tipo sofisticado de variável, que pode ter múltiplos ***fields***, e encapsula um conjunto de **métodos**.

Programação Orientada a Objetos

Programação Orientada a Objetos

- Objetos
  - Têm esse nome, por sua similaridade com os objetos do cotidiano
  - Um objeto (software) é similar a um objeto do cotidiano, dotado de múltiplas propriedades (cada uma representada por um campo), e para o qual pode ser invocado um conjunto de métodos, ou operações que o objeto é capaz de realizar
- Classes
  - Formam uma hierarquia baseada em herança, onde uma classe A que herda de outra classe B, possui em si todos os campos e métodos de B, mais todos aqueles adicionais que possam ser definidos somente para a classe A.
  - Dessa forma, é possível dizer que uma classe A que extende outra classe B, é um tipo de B.
    - A é, neste caso, uma sub-classe de B
    - Além de tudo de B, possui algumas outras peculiaridades exclusivas
    - Por exemplo a classe dos **cachorros**, é um tipo de **animal**, pois os **cachorros** possuem tudo que um **animal** tem, mais algumas peculiaridade que só os **cachorros** possuem
  - **Métodos Abstratos**
    - Algumas classes podem possuir métodos abstratos, ou seja, que não são definidos na própria classe, mas somente em suas sub-classe. Tais métodos são chamados de métodos abstratos
    - Exemplo: As classes **Vídeo** e **Áudio** são duas possíveis sub-classes da classe **MediaType**. A classe **MediaType** possui um método abstrato **Play**, que precisa ser implementado de maneira individual, dependendo se a sub-classe é um **Video** ou um **Áudio**. As implementações podem ser completamente diferentes
  - Classe Abstrata
    - Uma classe que possua um ou mais métodos abstratos é chamada de uma **classe abstrata**
    - Classes abstratas não podem ser instanciadas, mas somente suas sub-classes

Programação Orientada a Objetos

- Fields Estáticos
  - São fields de uma classe (ao invés de uma instância), e podem ser acessados sem que seja necessário criar uma instância da classe
- Métodos Estáticos
  - São métodos de uma classe (ao invés de uma instância), e podem ser acessados sem que seja necessário criar uma instância da classe
- Classes Estáticas
  - São aquelas em que todos os fields e métodos são estáticos
  - Não é possível criar um objeto de uma classe estática
    - Essas classes não foram feitas para serem instanciadas

Programação Orientada a Objetos

- Polimorfismo
  - Habilidade de um objeto de ser visto como instância de qualquer uma das classes da hierarquia de classes à qual pertence
  - Um objeto pode ser visto como um bloco de memória que pode ser interpretado sob diferentes pespectivas
- Overriding (Sobreescrita ou Reescrita)
  - Quando uma sub-classe **redefine** um método de sua classe pai, dando-lhe outra implementação
  - Uma instância de uma classe filha irá executar o código redefinido e não o código original definido na classe pai
- Dynamic Binding (Ligação Dinâmica)
  - Quando um método é overrided por uma subclasse, esse é o mecanismo por meio do qual o método correto a ser executado é dinamicamente determinado, dependendo da classe que está sendo considerada para o objeto polimórfico
- Overloading (Sobrecarga)
  - Métodos com o mesmo nome, em uma mesma classe, mas com diferentes parâmetros de entrada e saída
  - Corresponde a diferentes “versões” do método
  - É diferente do override, pois este corresponde a um método com exatamente o mesmo nome, mas definido em uma classe filha.

Programação Orientada a Objetos

- Construtor de uma Classe
  - Permite a criação de uma instância a partir de um conjunto mínimo de informações necessárias para a construção dessa instância
  - Uma classe pode ter diversos construtores diferentes (por overload do construtor)
- Desconstrutor de uma Classe
  - A partir de um objeto (instância de uma classe), o desconstrutor pode recortar as informações internas do objeto e restaurar os parâmetros originais que poderiam reconstruir um clone do objeto
- Propriedades (sets e gets)
  - Campos privados de um objeto podem ser definidos for funções do tipo set e recuperados por funções do tipo get
  - Esses campos são chamados de propriedades do objeto

Programação Orientada a Objetos

- Interface
  - Especificação de um conjunto de métodos e seus parâmetros, sem o código de sua implementação
  - Funcionam como um tipo de classe abstrata, sem a definição de campos
  - Foram criadas para resolver problemas de herança múltipla e campos com nomes idênticos
  - Classes podem implementar uma interface
  - Objetos polimórficos podem ser considerados como do tipo de interfaces que implementam
- Ponto de Entrada
  - Todo programa orientado a objetos precisa ter pelo menos um **ponto de entrada**
  - Ponto de entrada, usualmente é um método estático com nome padronizado (e.g. Main ou main)
  - Algumas linguagens permitem múltiplos pontos de entrada
    - Nesse caso, a linguagem precisa indicar qual deles deve ser utilizado durante a execução do programa
  - Algumas linguagens, como o C# 10.0, podem criar automaticamente um ponto de entrada, caso não seja especificado um

Programação Orientada a Objetos

- Programação “Orientada a Objetos”
  - Envolve uma mudança paradigmática no processo de programação
  - Ao invés de uma sequência de operações para realizar um processamento, deve-se pensar na montagem de um repertório de objetos que, interagindo entre si, devem realizar uma tarefa
  - Os objetos interagem entre si, mandando mensagens de um para o outro, onde cada mensagem corresponde a uma invocação de método
  - Os objetos são instâncias de diferentes classes, que precisam ser definidas no código do programa
  - Programar = Montar “Equipe” de objetos que, interagindo entre si resolvam o problema em questão
    - O foco deixa de ser no processo (sequência de ações) e passa para a organização do time de objetos, e suas competências
    - Atribuição de responsabilidades aos diferentes objetos, de modo coerente
    - Programa primeiro cria os objetos necessários e depois inicia a interação entre eles
      - → Novos objetos podem ir sendo criados, dinamicamente, à medida em que tornem-se necessários