



FACULDADE IMPACTA

CURSOS

INTRODUÇÃO PROGRAMAÇÃO ORIENTADO A OBJETO



ALEX SOUSA

SÃO PAULO - 10/2023



SUMÁRIO

SUMÁRIO.....	2
Programação Orientada a Objetos (POO).....	4
Objetos.....	5
Objetos computacionais.....	5
Objetos visuais.....	5
Objetos de domínio de trabalho.....	6
Objetos com tarefa relacionada.....	6
Objetos multimídia.....	6
Concepção de um sistema orientado a objeto.....	7
Análise.....	7
Programação.....	7
Vantagens.....	7
Objetos.....	8
Atributos.....	9
Operações e métodos.....	9
Operações.....	9
Métodos.....	10
Mensagens.....	10
Classes.....	11
Instanciação.....	11
Classes Abstratas.....	11
Herança.....	11
Herança simples.....	12
Herança múltipla.....	12
Persistência.....	12
Abstração.....	13
Encapsulamento.....	13
Polimorfismo.....	14
Compartilhamento.....	14
Notações Gráficas de Classes e Instâncias.....	14
Modelo de objetos.....	15
Diagramas de classes.....	15
Diagramas de instâncias.....	16
Estruturas e Relacionamentos.....	16
Generalização e herança.....	16
Agregação.....	18
Conexões entre objetos.....	18
Conexão de ocorrência.....	18
Conexão de mensagem.....	19
Ligações e associações.....	19
Ligações.....	19
Associações.....	19
O que é um software.....	20

Fase 1 – Conceito.....	20
Fase 2 – Design.....	20
Fase 3 – Desenvolvimento.....	20
Fase 4 – Testes.....	20
Fase 5 – Implantação.....	20
Tipos de software.....	21
Interface de usuário (User Interface ou UI).....	21
Componentes.....	21
Serviços.....	21
Web services.....	21
Linguagens de programação.....	22
Bancos de dados.....	22
Banco de dados de arquivo.....	22
Banco de dados de servidor.....	22
Tecnologias e ferramentas.....	22
Java.....	23
Plataforma .NET.....	23
Frameworks.....	23
Metodologias de desenvolvimento.....	23
Programação estruturada.....	23
RUP.....	24
XP.....	24
Scrum.....	24

Programação Orientada a Objetos (POO)

Modelos de Programação

- POO
- Estruturada - um comando atrás do outro
- Orientada a Eventos
- Orientada a Aspectos

Conceitos POO

- Herança
- Campos
- Elementos
- Métodos
- Associação
- Encapsulamento
- Propriedades
- Interface
- Polimorfismo
- Agregação Atributos
- Eventos
- Dependências

POO é um estilo de programação que utiliza **classes** como elemento principal para a construção de um software.

Em comparação à visão tradicional de programação, em que o programa pode ser visto como uma lista de instruções para o computador, na POO, cada um dos objetos relaciona-se com os demais.

Essa habilidade é demonstrada em diferentes características, como:

- Processamento de dados;
- Recebimento de mensagens de outros objetos;
- Envio de mensagens para outros objetos da aplicação.

Uma das vantagens de utilizar a programação orientada a objetos é que ela permite alterar ou substituir partes de um sistema sem que haja riscos de ocorrência de erros. Além disso, ela oferece uma visão mais natural, uma forte arquitetura e um alto grau de reusabilidade do código.

Os sistemas orientados a objetos são definidos estruturalmente como duas partes separadas, as funcionalidades são nada mais que conjuntos de objetos interativos. Esses objetos geralmente possuem ações que interferem no comportamento de um sistema e informações que armazenam dados. Em termos de modelagem, os objetos são representados por classes.

Classe é uma estrutura de programação que descreve as características e comportamentos de um **objeto**.

Uma classe é composta de um conjunto de elementos que compartilham uma estrutura (atributos ou valores) e procedimentos comuns (métodos).

Desse modo, a classe representa um conjunto de objetos com características semelhantes.

O processo de criar um objeto é chamado **instanciação**.

Objetos

A orientação a objetos visa representar, de maneira idêntica, as situações do mundo real nos sistemas computacionais. Para ela, os sistemas operacionais não devem ser vistos como uma coleção estruturada de processos, mas sim como uma coleção de objetos que interagem uns com os outros organizadamente, assim como ocorre no mundo real.


Em uma concepção abrangente, um objeto pode ser entendido como sendo um elemento físico, por exemplo, uma pedra, uma cadeira ou um animal, ou como um elemento abstrato, como uma conta bancária.

Dentro do ambiente de TI, os objetos computacionais são objetos que estão localizados dentro de sistemas de computador. A intenção ao especificar esses objetos é retratar as mesmas características dos objetos do mundo real e programar comportamentos que se aproximam dos encontrados na realidade. Como o comportamento desses objetos já está programado, um programador não precisa entender o funcionamento interno desse comportamento para interagir com ele. Para isso, basta ativar os comportamentos programados.

Um exemplo comum desse processo são os aplicativos para desenvolvimento de softwares que utilizam recursos de arrastar e soltar. O programador pode criar uma tela arrastando e soltando botões, caixas de textos, listas, conexões com banco de dados para uma tela, sem se preocupar com a forma como esses objetos funcionam internamente.

Curso POO Curso em Vídeo - [Curso em Vídeo - POO](#)

Parei na aula - Curso POO Teoria #05a - Exemplo Prático com Objetos

Classes, Objetos, Atributos e Método  [Entenda Classe, Objeto, Atributo e Método \(POO\)](#)

Resumo dos principais conceitos de POO - [Fundação Bradesco](#)

Objetos computacionais

A representação dos objetos computacionais no computador retrata apenas uma imagem dos objetos do mundo real, ou seja, são modelos abstratos dos objetos reais. No entanto, o comportamento desses modelos pode ser igual ao da realidade. Para que ocorra a interação dentro de um sistema de computador, os objetos computacionais devem saber como reagir diante de uma ação.

Conforme mencionado anteriormente, o comportamento desses objetos já está previamente programado, portanto, eles possuem a informação necessária para saber como reagir. Isso significa que o seu comportamento não depende do sistema.

Para facilitar a compreensão acerca da relação entre os objetos computacionais e o comportamento associado a eles, consideremos, como exemplo, quatro tipos de objetos existentes no ambiente computacional:

Objetos visuais

Interagem diretamente com o usuário, permitindo que ele aperte botões, mova peças e selecione itens dentro de caixas. Neste caso, a interação do usuário com esses objetos ocorre em um monitor, com a utilização de um mouse ou teclado. Mesmo assim, existe um nível de abstração e interação muito bom em relação ao mundo real.

Um fator importante e que exige a nossa máxima atenção é a maneira como utilizaremos esses objetos, pois cada objeto já possui o conhecimento de como deve funcionar, sem que o programador escreva um código.

Alguns exemplos de objetos visuais são: Menus, Caixas de Texto, Botões e Listas

A calculadora do Windows utiliza diversos objetos visuais:



É possível identificar nesta imagem da calculadora: um menu, uma caixa de texto e diversos botões.

Objetos de domínio de trabalho

São objetos que estão envolvidos ou são criados durante o desenvolvimento de um sistema de computador. Além disso, eles se relacionam diretamente com o trabalho desenvolvido e, muitas vezes, não são percebidos pelos olhos do usuário final. Outra característica dos objetos de domínio de trabalho é que eles estão envolvidos diretamente com o sistema de informações.

Por exemplo, um software controlador de estoque pode conter alguns objetos internos, como Cliente, Produto, Fornecedor, Venda ou Compra

Objetos com tarefa relacionada

São arquivos que também possuem comportamentos inclusos, como objetos do tipo documentos e arquivos multimídia. Apesar de apresentarem esses comportamentos, em muitos casos é necessário ter softwares aplicativos para abri-los.

Por exemplo, um software de compra de ingressos para o cinema pode ter uma funcionalidade que permite ao usuário ver o trailer de um filme. Se esse trailer for um arquivo do tipo MOV, um aplicativo chamado QuickTime, da Apple, deverá estar instalado no computador, além do aplicativo que é responsável pela exibição correta de imagem e som.

Objetos multimídia

A maioria das pessoas já está familiarizada com os objetos multimídia, os quais podem conter som, imagem, animação ou vídeo. Esses objetos sabem de que forma devem se comportar diante de uma situação.

Por representarem os objetos reais de forma bastante idêntica, os objetos multimídia são de fácil utilização.

Concepção de um sistema orientado a objeto

O conceito de orientação a objetos consiste em criar um sistema computacional como um sistema orgânico, que seja formado por objetos que interajam uns com os outros.

Esse conceito pode ser empregado na análise de sistemas e na programação, fazendo com que uma das principais vantagens da orientação a objetos seja a possibilidade de utilizar a mesma técnica, tanto para a definição lógica do sistema quanto para a sua implementação.

Análise

A análise orientada a objetos consiste em definir os objetos que pertencem a um sistema e a maneira como eles se comportam. O processo dessa análise consiste em identificar os seguintes elementos:

- Os objetos que existem dentro do ambiente que desejamos automatizar;
- Os atributos desses objetos, ou seja, que tipos de informação esses objetos devem conter;
- As ações que esses objetos podem executar.

Programação

Consiste em utilizar estruturas de dados que façam uma simulação do comportamento dos objetos. Dessa forma, a programação é realizada com mais facilidade com a utilização de uma única técnica tanto para a análise quanto para a programação.

Na programação orientada a objetos, os objetos identificados pelo analista em seu diagrama podem ser implementados exatamente da forma que foram projetados. Por outro lado, na análise estruturada é necessário traduzir os diagramas da análise para estruturas de dados e de programação.

Depois de fazer uma análise orientada a objetos, é possível implementá-la em uma linguagem tradicional e, também, implementar sistemas analisados com outras técnicas, utilizando linguagens orientadas a objetos.

Vantagens

A seguir veremos algumas vantagens da utilização da técnica de orientação a objetos:

• Organização

A partir do momento que os desenvolvedores de sistemas começam a utilizar a técnica da orientação a objetos, eles passam a usufruir de diversas vantagens, dentre as quais a principal é a possibilidade de reunir em uma mesma estrutura os dados e os processos que são executados sobre esses dados. Consequentemente, há um aumento no nível de organização e simplicidade do programa.

• Produtividade

Outra vantagem é o ganho de produtividade, pois, quando um desenvolvedor cria um objeto para realizar uma tarefa difícil ou altera a maneira como esse objeto a executa internamente, os demais desenvolvedores que venham a interagir com esse objeto precisarão apenas acessá-lo para executar as mesmas tarefas, sem a necessidade de saber de que forma ocorre a execução e sem precisar alterar seus programas para continuar tendo acesso ao novo comportamento.

• Redução do custo

Além das vantagens mencionadas anteriormente, também há a redução no custo de desenvolvimento e no risco de ocorrência de erros. Isso ocorre porque, depois de criar uma estrutura eficaz de objetos, será possível incluir módulos adicionais no sistema, que reutilizem funcionalidades já desenvolvidas, ou seja, utilizem o mesmo código em sistemas diferentes. Consequentemente, não há necessidade de reprogramação.

- **Reaproveitamento**

Mais uma vantagem encontrada é a possibilidade de transformar objetos semelhantes, com a finalidade de aproveitá-los em novos sistemas. Isso significa que podemos fazer pequenas alterações em objetos que possuem características parecidas com as especificações necessárias de um novo objeto, a fim de que esse novo objeto resultante da alteração seja utilizado em um novo sistema.

- **Facilidade de manutenção**

Quando é preciso realizar alguma alteração em um sistema, a flexibilidade oferecida pela técnica de orientação a objetos permite que o desenvolvedor adapte, exclua ou inclua novos objetos no sistema rapidamente, sem comprometer o funcionamento do mesmo.

- **Trabalho em equipe**

Usando orientação a objetos, é fácil dividir tarefas entre diversas equipes de programação. Uma parte da equipe pode trabalhar no código relacionado aos bancos de dados e outra parte pode trabalhar na interação com o usuário. Esse tipo de programação é conhecido como Programação em Camadas.

Enfim, como a programação orientada a objetos lida com objetos pré-prontos, ela não apenas simplifica a criação de novos sistemas como tem a finalidade de maximizar a reutilização dos objetos.

Objetos

Para entendermos melhor o conceito de objeto, podemos considerá-lo como uma representação do mundo real. Em termos gerais, entendemos por objetos quaisquer elementos da natureza aos quais é possível atribuir comportamentos e características. No entanto, em termos de computação, entendemos por objetos os elementos capazes de representar uma entidade que esteja no domínio de interesse do problema analisado. As entidades representadas por objetos podem ser concretas ou abstratas.

As linguagens de programação que são orientadas a objetos possuem mecanismos que, a partir do conceito de classes, permitem determinar os tipos de objetos que serão destinados a cada aplicação. Os objetos que apresentam semelhanças entre si são agrupados em classes.

Os objetos, quando são criados, ocupam um espaço na memória que é utilizado para o armazenamento de seu estado e de um grupo de operações, as quais podemos aplicar ao objeto. Mais especificamente, o estado de um objeto refere-se aos valores de seu conjunto de atributos, e seu grupo de operações refere-se ao conjunto de métodos. A classe é responsável por definir ambos, atributos e métodos.

Tendo em vista que os objetos são instâncias de classe, para que eles realizem suas tarefas é preciso que as instâncias sejam criadas, uma vez que, por meio de sua manipulação, as tarefas dos objetos podem ser efetuadas. Uma vez realizadas essas tarefas, podemos excluir os objetos.

Ainda em relação aos objetos, é importante notar alguns aspectos:

- Todos os objetos são distinguíveis e possuem uma identidade própria;

- Os objetos possuem duas finalidades: promover o entendimento do mundo real e dar suporte a uma base prática para uma implementação computacional;
- A decomposição de um problema em objetos depende da natureza do problema e do julgamento do projetista, ou seja, não existe uma maneira correta ou única de fazer esta decomposição;
- Um objeto é constituído por atributos e métodos.

Podemos citar como exemplos de objetos: Nota Fiscal, Cliente, Produto, Boletim, Orçamento.

Atributos

Os valores de dados assumidos pelos objetos de uma classe são chamados de atributos. Para cada instância do objeto, um atributo possui um valor, que pode ser o mesmo para instâncias diferentes. Para exemplificar, o objeto Pessoa pode possuir os atributos Nome e Altura, e o objeto Produto pode possuir os atributos Nome, Preço e EstoqueAtual, ou seja, os atributos podem ser entendidos como variáveis ou campos utilizados para armazenar os diferentes valores que as características dos objetos podem conter.

Dentro de uma classe, os nomes de cada atributo devem ser exclusivos, porém, podemos ter atributos com o mesmo nome em classes diferentes.

Os atributos dos objetos só podem ter seus valores alterados por meio de estímulos internos ou externos. Para modificar esses valores, é necessário disparar eventos que provoquem a transição de estados no objeto.

Os dois itens a seguir garantem a independência completa de qualquer objeto em relação aos demais, além da possibilidade de alteração dos atributos e do código interno dos métodos do objeto, sem o risco de afetar outros objetos internamente:

- Cada objeto é responsável pela mudança de seus próprios atributos;
- Salvo por meio da solicitação de serviços, nenhum objeto possui a capacidade de interferir nos atributos de outro objeto.

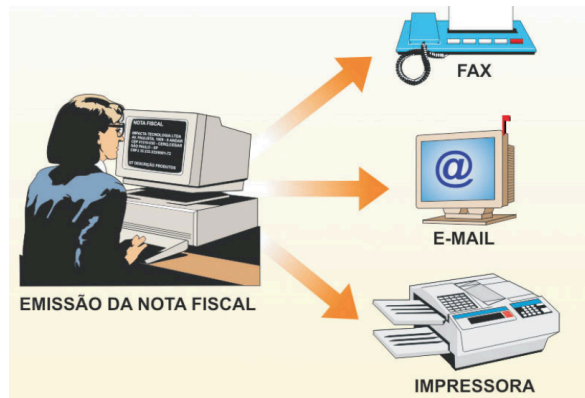
Operações e métodos

Vejam os que são as operações e os métodos:

Operações

Operação é uma transformação, ou função, que pode ser aplicada a objetos de uma classe ou por esses objetos. Dentro de uma classe, as mesmas operações são compartilhadas por todos os objetos.

Toda operação possui, como argumento implícito, um objeto-alvo, cuja classe determinará o comportamento da operação. O objeto conhece a classe à qual pertence e por isso é possível escolher a implementação correta da operação.



As operações que se aplicam a diversas classes diferentes são chamadas de polimórficas, pois podem assumir formas diferentes em cada uma das classes.

Métodos

Os métodos permitem que um objeto se manifeste e interaja com outros objetos. Método é uma implementação de uma operação para uma classe específica.

Um método possui uma assinatura constituída pelo nome de uma operação, o número, tipo e ordem de seus argumentos e pelo valor de retorno. Como estratégia de desenvolvimento, é recomendável manter, além de um comportamento consistente entre as implementações, assinaturas que sejam coerentes para os métodos que implementam uma determinada operação.

Como exemplo, podemos citar que o objeto Conta pode conter os métodos Retirar e Depositar.

Mensagens

A mensagem, ou seja, o meio de comunicação entre os objetos, é uma chamada feita a um objeto com o objetivo de invocar um de seus métodos, ativando um comportamento descrito por sua classe. A mensagem é uma requisição de ação junto com argumentos necessários para a execução da tarefa solicitada.



Vejamos um exemplo:

MinhaConta.Depositar(100)

O objeto MinhaConta está enviando a mensagem Depositar e passando o parâmetro 100. Esse valor será usado para executar a ação corretamente (depositar 100 reais na conta representada pelo objeto (MinhaConta)).

Classes

As classes criam representações computacionais a partir de entidades do mundo real. São elementos fundamentais no desenvolvimento de softwares orientados a objetos e podem ser definidas como descrições coletivas ou genéricas do mundo real. Assim, em um sistema, a definição das classes deve procurar inspiração nas entidades mundanas.

Computacionalmente, podemos definir classe como uma abstração de um conjunto de objetos, os quais são agrupados por possuírem similaridades em termos de comportamento (operações) e características (atributos). Sendo assim, as propriedades ou os atributos de um objeto são descritos a partir da definição de uma classe.

É importante notar que os objetos são criados pela classe, ou seja, não criamos os objetos, mas sim definimos, na classe, os atributos e métodos necessários para essa criação. O ato de criar um objeto é chamado de **Instanciação**. Instanciar um objeto é criar uma cópia de uma classe na memória, para uso no programa.

Instanciação

Na teoria de orientação a objetos, um objeto é definido como uma instância de uma classe. A instanciação é a criação de um objeto por uma classe, em que esta funciona como um gabarito, ou modelo, para essa criação.

Classes Abstratas

Entendemos por classes abstratas as classes a partir das quais não é possível realizar qualquer tipo de instância. São classes feitas especialmente para serem modelos para suas classes derivadas. As classes derivadas, via de regra, deverão sobrescrever os métodos para realizar a implementação dos mesmos. As classes derivadas das classes abstratas são conhecidas como **Classes Concretas**.

Como medidas de segurança, as classes abstratas podem ser somente estendidas e a criação de um objeto a partir da mesma é um procedimento evitado. Além disso, caso um ou mais métodos abstratos estejam presentes nessa classe abstrata, a classe filha será, então, forçada a definir tais métodos, pois, caso contrário, a classe filha também se tornará abstrata.

A funcionalidade dos métodos abstratos que são herdados pelas classes filha normalmente é atribuída de acordo com o objetivo ou o propósito dessas classes. É possível, porém, não atribuímos uma funcionalidade a esses métodos abstratos. Nesse caso, faz-se necessário, pelo menos, declará-los.

Os métodos abstratos estão presentes somente em classes abstratas, e são aqueles que não possuem implementação.

Herança

A herança possibilita que as classes compartilhem seus atributos, métodos e outros membros da classe entre si. Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente.

Na herança, temos dois tipos principais de classe:

- **Classe base:** A classe que concede as características a uma outra classe;
- **Classe derivada:** A classe que herda as características da classe base.

O fato de as classes derivadas herdarem atributos das classes base assegura que programas orientados a objetos cresçam de forma linear, e não geométrica, em complexidade.

Cada nova classe derivada não possui interações imprevisíveis em relação ao restante do código do sistema.

Com o uso da herança, uma classe derivada geralmente é uma implementação específica de um caso mais geral. A classe derivada deve apenas definir as características que a tornam única.

Por exemplo, uma classe base, que serviria como um modelo genérico, poderia ser a classe Produto com os campos Nome e Preço. Já uma classe derivada poderia ser a Livro, com os campos Nome e Preço herdados de Produto e acrescida do campo Numero de Paginas.

De maneira natural, as pessoas visualizam o mundo como sendo formado de objetos relacionados entre si hierarquicamente. Vejamos a seguir a relação entre animais, mamíferos e cachorros.

Os animais, sob uma descrição abstrata, apresentam atributos, tais como tamanho, inteligência e estrutura óssea, além de aspectos comportamentais, como mover-se, dormir, respirar, alimentar-se, entre outros. Os atributos e aspectos comportamentais descritos definem a classe dos animais.

Se analisarmos os mamíferos, que estão dentro da classe de animais, notaremos atributos mais particulares, como tipo de dente, pelos e glândulas mamárias. Os mamíferos são classificados como uma classe derivada dos animais, que por sua vez são uma classe base de mamíferos.

Pela chamada hierárquica de classes, a classe derivada mamíferos recebe todos os atributos de animais, partindo do princípio que uma classe derivada recebe por herança todos os atributos de seus ancestrais.

Herança simples

A herança simples determina que uma classe herdará características de apenas uma superclasse (classe base).

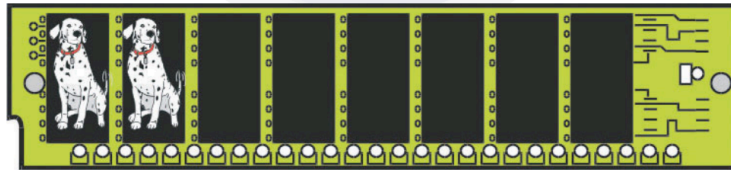
Herança múltipla

A herança múltipla determina que uma classe herdará características de duas ou mais superclasses, como é o caso das classes derivadas, quando herdam duas ou mais classes base.

Quando utilizamos a herança múltipla, é importante observarmos a manipulação de nomes de membros duplicados nas classes base. Para especificar a qual membro a classe derivada se refere, utilizamos o mecanismo chamado qualificação. A qualificação consiste na prefixação do nome do membro com o nome da classe base a que ele faz referência. Os nomes do membro (atributo ou método) e da classe base são separados por um duplo dois-pontos (::).

Persistência

A persistência de um objeto é o tempo que este permanece armazenado na memória RAM (principal) ou auxiliar (um meio magnético, por exemplo).



Um objeto torna-se persistente a partir do momento em que os valores de seus atributos e as informações necessárias para a manutenção de suas conexões com outros objetos são armazenados em uma mídia qualquer.

A persistência refere-se mais a um conceito do que a uma técnica de análise ou de desenvolvimento.

Os objetos criados e destruídos sem que sejam salvos de alguma maneira são chamados objetos não persistentes. Esses objetos são temporários, já que os valores de seus atributos são transientes e, por isso, não existe a necessidade ou o interesse de que sejam gravados.

Abstração

A abstração ocorre quando nos focamos nos aspectos essenciais de uma entidade e ignoramos propriedades secundárias. No desenvolvimento de sistemas, a abstração consiste em nos concentrarmos no que o objeto é e executa, para somente depois tomar decisões sobre sua implementação.

Por meio da abstração, isolamos o objeto que desejamos representar de um ambiente complexo e representamos nele apenas as características relevantes, de acordo com o problema atual.

Assim, ao utilizarmos os conceitos de abstração, as decisões relacionadas ao desenvolvimento e à implementação de objetos podem ser tomadas quando entendemos melhor o problema a ser resolvido. Portanto, a abstração é um dos elementos mais importantes da programação orientada a objetos.

Podemos dizer que a abstração é um processo por meio do qual separamos os fatos relevantes dos detalhes que não têm grande importância. É um conceito bastante adequado em situações de grande complexidade.

A utilização apropriada da abstração permite utilizar em todas as fases de desenvolvimento do sistema (desde a sua análise até a sua documentação) um mesmo modelo conceitual (orientação a objetos).

Encapsulamento

Diferentemente do que ocorre na programação tradicional, na programação orientada a objetos, no modo de utilização dos atributos e métodos, os dados e as operações realizadas com esses dados são encapsulados em uma única entidade. Assim, a única forma de conhecer ou alterar os atributos de um objeto é por meio de seus métodos. A lista a seguir descreve as vantagens do encapsulamento:

O objeto é disponibilizado com toda a sua funcionalidade, sem a necessidade de conhecermos seu funcionamento ou armazenamento interno;

- É possível modificar um objeto internamente, acrescentando métodos, sem que isso afete os outros componentes do sistema que utilizam o objeto modificado;

- O processo de desenvolvimento de sistemas é acelerado e simplificado, já que os usuários dos objetos não precisam necessariamente saber como eles são constituídos internamente;
- A implementação de um comportamento pode ser modificada radicalmente sem que haja impacto no resto do programa. Isso é possível porque o código que utiliza o objeto não depende da maneira como ele é implementado.

Para que dois objetos possam interagir, é necessário que um conheça o conjunto de operações disponíveis no outro (interface), e vice-versa, para que possam enviar e receber informações, além de ordenarem a realização de procedimentos.

Podemos definir como interface o contrato entre a classe e o mundo exterior. Quando uma classe implementa uma interface, se compromete a fornecer o comportamento publicado por essa interface.

Polimorfismo

Definimos polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de forma específica para cada uma das classes derivadas.

O polimorfismo é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

Compartilhamento

Na orientação a objetos, existem técnicas que possibilitam o compartilhamento em vários níveis. Comportamento e herança de estrutura de dados permitem o compartilhamento de estruturas comuns entre classes similares diversas, sem redundância. A utilização da herança no compartilhamento do código traz duas vantagens:

- Economia de código;
- Redução dos casos distintos para análise, já que há uma maior clareza conceitual a partir do reconhecimento de que operações diferentes estão relacionadas ao mesmo elemento.

O desenvolvimento orientado a objetos permite, além do compartilhamento de informação dentro de um projeto, o reaproveitamento de códigos (e até mesmo de projetos) em projetos futuros. A metodologia disponibiliza ferramentas que possibilitam o compartilhamento, como a abstração, a encapsulação e a herança. A reutilização entre projetos pode utilizar como estratégia a criação de bibliotecas de elementos reutilizáveis, além do pensamento do desenvolvedor voltado para termos genéricos, ou seja, não focado apenas na aplicação atual.

Notações Gráficas de Classes e Instâncias

Abordaremos, nesta leitura complementar, as notações gráficas do Modelo de Objetos que faz parte da OMT, ou Object Modeling Technique (Técnica de Modelagem de Objetos), desenvolvida por Rumbaugh e utilizada principalmente por desenvolvedores de sistemas e softwares que suportam um ciclo completo de desenvolvimento, objetivando implementações orientadas a objetos.

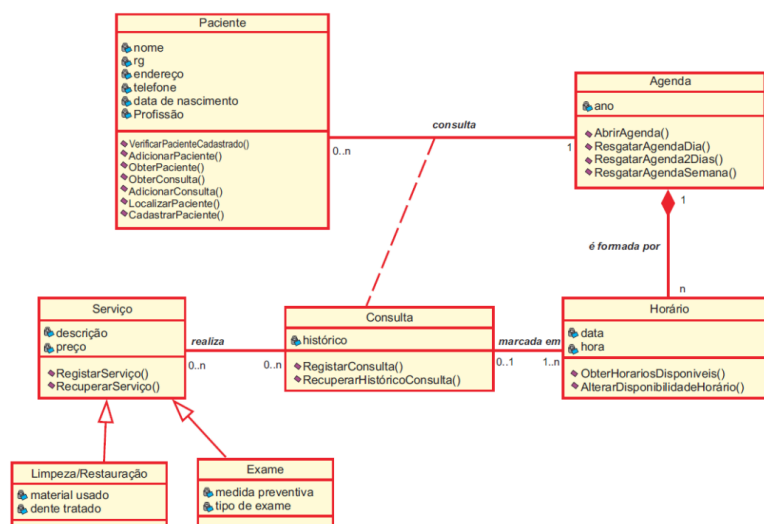
A OMT é uma das técnicas de desenvolvimento orientadas a objeto mais populares atualmente e possui uma notação principal simples, o que faz com que seja facilmente compreendida, desenhada e utilizada.

Modelo de objetos

A construção do Modelo de Objetos é utilizada para capturar, do mundo real, os conceitos que são importantes para uma aplicação. O Modelo de Objetos descreve a estrutura estática de objetos de um sistema, que inclui:

- A identidade de um objeto;
- Os relacionamentos de um objeto com outros objetos;
- Os atributos de um objeto;
- As operações de um objeto.

O Modelo de Objetos é composto pelo diagrama do modelo de objetos e pelo dicionário de dados, que descreve os atributos (é utilizado para evitar que os atributos sejam explicitados graficamente e para garantir que os diagramas sejam mais administráveis visualmente).



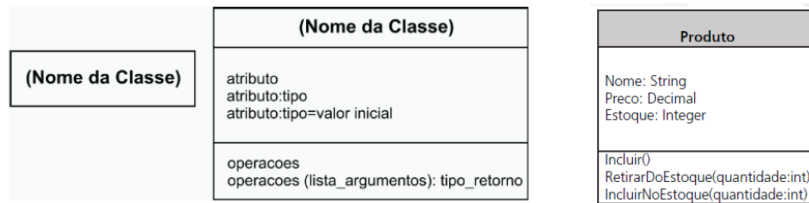
Há dois tipos de diagramas de objetos:

- Diagrama de classes: Descreve o caso geral da modelagem;
- Diagrama de instâncias: Utilizado para exemplificar.

Diagramas de classes

Uma classe é um grupo de objetos que são semelhantes em seus atributos (propriedades), operações (comportamento), relacionamentos com outros objetos e semântica. Cada classe possui um grupo de atributos e operações.

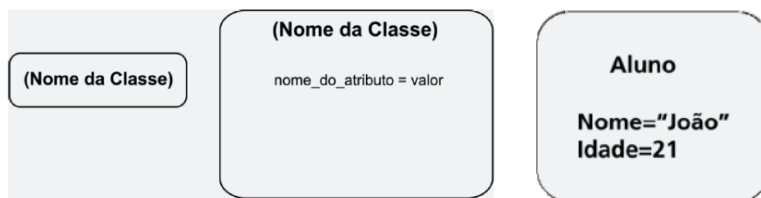
A notação OMT para uma classe é um retângulo contendo o nome da classe em negrito e seções opcionais para os atributos e operações (essas seções devem ser separadas por linhas horizontais):



Diagramas de instâncias

Utilizados na documentação de testes e na apresentação de resultados, os diagramas de instâncias, também conhecidos como diagramas de objetos, descrevem os relacionamentos de um grupo particular de objetos.

A notação OMT para instâncias é um retângulo de cantos arredondados, que inclui o nome da classe em negrito e entre parênteses. Opcionalmente, pode incluir os valores dos atributos separados do nome da classe por uma linha horizontal:



Estruturas e Relacionamentos

Nesta leitura complementar, apresentaremos uma parte do Modelo de Objetos (um subconjunto do modelo OMT), que retrata o relacionamento entre objetos. Esse modelo pode ser introduzido a partir do estabelecimento dos principais conceitos e definições da abordagem de orientação a objetos.

Generalização e herança

A generalização e a herança são abstrações que permitem que classes compartilhem similaridades ao mesmo tempo em que preservam as características que as diferem.

O relacionamento de uma classe com suas versões refinadas, ou seja, especializadas, é chamado de generalização. Nesse ponto, é importante definir dois conceitos:

- **Superclasse ou classe base:** É a classe por meio da qual podemos gerar subclasses; também é conhecida como classe pai;
- **Subclasse ou classe derivada:** É a versão refinada de uma superclasse.

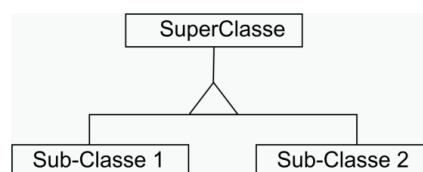
Vejamos um exemplo: pode haver uma classe base chamada Pessoa, com características comuns a todas as pessoas: Nome, Email, Telefone. Uma classe derivada poderia ser a classe Funcionario, que, além de ter os campos Nome, Email e Telefone, teria também os campos Salário e Cargo. Uma outra classe derivada poderia ser a Cliente, que, além dos campos herdados da classe base, teria o campo Nome do Contato ou Celular.



A generalização também pode ser chamada de relacionamento is-a (ou seja, é-um), já que toda instância de classe derivada também é uma instância de classe base. As características da classe base são herdadas pela classe derivada, e as operações e os atributos comuns a um conjunto de classes derivadas são colocados como atributos e operações da classe base.



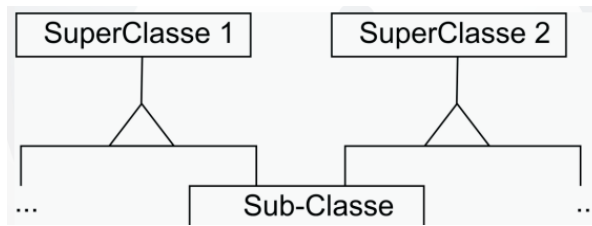
A imagem seguinte exibe a notação diagramática de OMT utilizada para representar a generalização: um triângulo cujo vértice aponta para a classe base.



Cada associação do tipo generalização pode possuir um discriminador associado. O discriminador é um atributo (do tipo enumeração) utilizado para indicar qual é a propriedade do objeto que é abstraída pelo relacionamento de generalização. Trata-se de um nome para a base de generalização.

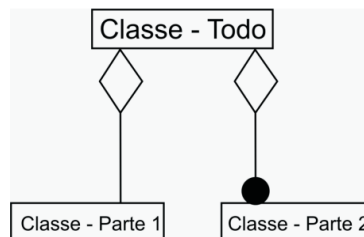
Uma característica de classe base pode ser sobreposta por uma classe derivada, caso esta defina uma característica própria com o mesmo nome. A característica própria da classe derivada refinará e substituirá a característica da classe base.

Temos a herança múltipla representada graficamente na imagem a seguir:



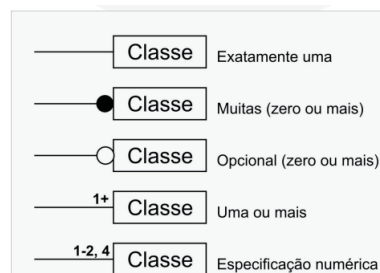
Agregação

Na agregação, há certa coesão entre as partes, porém, elas não são totalmente dependentes. Para exemplificar, podemos imaginar a criação de uma nova classe, Família, constituída de vários membros da classe Pessoa, porém, os membros da classe Pessoa podem existir fora da classe Família.



A agregação define um relacionamento do tipo uma-parte-de. Nesse tipo de relacionamento, alguns objetos (parte) representam componentes de outro objeto (todo), porém, um não contém o outro, o todo não contém a parte, não havendo relação de posse exclusiva. Em uma agregação, um componente que faz parte de outro pode existir isoladamente.

Isso é diferente do que acontece na composição, como em uma nota fiscal, por exemplo, em que o objeto que representa um produto comprado pertence a essa nota e apenas a ela, não fazendo sentido esse objeto existir se não for dentro dessa nota fiscal.



Conexões entre objetos

Conexões de ocorrência e conexões de mensagens são tipos de conexão entre objetos que não caracterizam um tipo de hierarquia ou de estrutura.

Conexão de ocorrência

Este tipo de conexão ocorre quando um atributo de um objeto faz referência a outro objeto. As conexões de ocorrência normalmente são criadas quando identificamos que atributos redundantes de um objeto são parte de

outro objeto. As conexões de ocorrência possibilitam também o registro do número de vezes que um objeto faz referência a outro ou é referenciado, o que também é chamado de cardinalidade.

Conexão de mensagem

Este tipo de conexão ocorre quando um objeto envia uma mensagem para outro objeto. No contexto de uma conexão de mensagem, existem os seguintes elementos:

- **Objeto transmissor:** O objeto que transmite a mensagem para um outro objeto;
- **Mensagem:** Procedimento que dispara um método específico para que o objeto receptor execute um comportamento determinado. Pode ser uma solicitação de informações ou uma solicitação para a realização de alguma ação no objeto;
- **Objeto receptor:** O objeto que recebe a mensagem e retorna ou não uma resposta para o objeto transmissor.

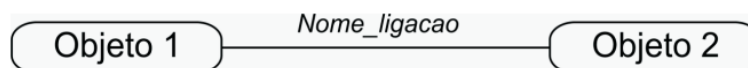
Ligações e associações

Ligações e associações são mecanismos que estabelecem relacionamentos entre objetos e classes. A seguir, cada um desses mecanismos será descrito:

Ligações

Uma ligação conecta duas instâncias de objetos, de forma física ou conceitual, por exemplo, Paulo é aluno da Faculdade_Impacta. Uma ligação é uma instância de uma associação.

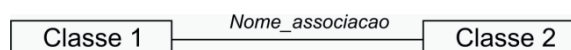
Na notação de diagramas OMT, uma ligação é representada como uma linha que conecta dois objetos. A imagem seguinte exemplifica um diagrama OMT com ligação:



Associações

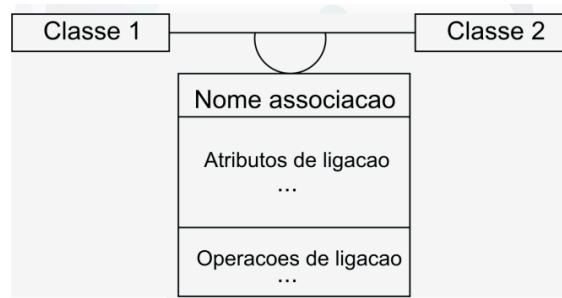
As associações definem um conjunto de ligações que compartilham a mesma semântica e estrutura, por exemplo, uma pessoa é aluna de uma faculdade. Podemos entender que, assim como as classes descrevem grupos de objetos potenciais, as associações descrevem grupos de ligações potenciais.

Uma associação é representada por uma linha que conecta duas classes. A próxima imagem exemplifica um diagrama OMT com associação:



Na notação de diagramas OMT, normalmente, os nomes das associações e das ligações são exibidos em estilo itálico e podem ser omitidos quando só existir uma associação de sentido óbvio entre um par de classes, para as associações, ou entre um par de objetos, para as ligações.

A OMT introduz o conceito de atributo de ligação para os casos em que os atributos dizem respeito a associações e não a classes. A associação pode ser modelada como uma classe conectada à associação quando possuir operações associadas:



O que é um software

Software é um conjunto de instruções a serem executadas por um computador com o objetivo de resolver algum problema. Desenvolver um software é elaborar tais instruções de maneira lógica e ordenada, em um formato que o computador consiga interpretar.

A elaboração de um software passa por cinco fases: Conceito, Design, Desenvolvimento, Testes e Implantação. A seguir veremos o que acontece em cada uma dessas fases.

Fase 1 – Conceito

Nesta fase, o problema a ser resolvido é entendido pelos analistas para que seja possível construir a solução da melhor maneira possível. Esta é a fase mais importante do projeto, porque servirá de guia para todo o restante do processamento.

Fase 2 – Design

Nesta fase serão escolhidas as tecnologias que melhor atenderão às necessidades dos clientes, assim como a equipe necessária, o prazo e o custo do projeto. Normalmente, esta etapa é feita com os analistas trabalhando em conjunto com os programadores e técnicos. O resultado final desta fase são diversos diagramas em UML ou outra linguagem de especificação.

Fase 3 – Desenvolvimento

Nesta fase entra o trabalho dos programadores, técnicos, especialistas em bancos de dados (dbas) e designers. É a construção do software, feita de acordo com as especificações definidas na fase 2.

Fase 4 – Testes

A fase de testes consiste em verificar se as regras definidas na fase 2 atendem às necessidades definidas na fase 1. Muitas vezes, neste ponto o desenvolvimento retorna à fase 2 para redesenhar parte do projeto ou, às vezes, para a fase 1, para um melhor entendimento das regras do negócio.

Fase 5 – Implantação

Uma vez testado, o software pode ser implantado no ambiente de produção. Muitas vezes é necessário outro projeto apenas para esta fase, principalmente quando já existe um sistema e os dados necessitam de migração. Às vezes é necessário voltar às fases anteriores para ajustar algo que funcionou bem no ambiente de testes, mas que se mostrou ineficiente no ambiente de produção.

Tipos de software

Um software é composto por diversos tipos de projetos. Às vezes, um único projeto é suficiente e, às vezes, são necessários mais de 100 projetos para criar uma solução que atenda às necessidades de uma empresa.

Interface de usuário (User Interface ou UI)

A UI é a parte do software com a qual o usuário final interage. Quando uma pessoa utiliza um caixa eletrônico de um banco, por exemplo, os botões na tela e as mensagens são interfaces de usuário. Quando o usuário confirma uma transferência de saldo, não é a interface que faz a transação, mas outro componente com o qual a interface se comunica.

Existem diversas interfaces:

- Programas Windows, como Excel ou Word;
- Programas Web, como lojas on-line;
- Programas para celular;
- Programas para dispositivos portáteis;
- Programa para o surface da Microsoft.

Componentes

Os componentes são a parte do software que faz algum processamento, mas que não interage com o usuário final diretamente. Por exemplo, ao confirmar uma transação em um caixa eletrônico, um componente deve guardar essa informação em um banco de dados. Mas esse componente pode ser, também, acionado pela Internet ou por um celular. Esse componente não está ligado a uma interface em particular, ele apenas recebe uma mensagem e realiza a tarefa de gravação.

Existem diversos tipos de componentes:

- Componentes para gravar em bancos de dados;
- Componentes distribuídos, para serem chamados de outro computador;
- Componentes contendo grupos de controles para criar interfaces de usuários.

Serviços

O Windows utiliza o conceito de serviços, que é um software ativado automaticamente quando o computador é ligado. Esse tipo de software é chamado de servidor. Por exemplo, os softwares antivírus são geralmente instalados como serviços no computador. Assim, o usuário não precisa se lembrar de chamar o programa toda vez que ligar o computador e, além disso, muitos vírus entram em ação antes de o usuário fazer o login na máquina.

Web services

Os Web services são componentes instalados em um servidor Web. A vantagem de um Web service é fazer com que um computador que não está na rede local possa chamar serviços de outro computador pela Internet.

Linguagens de programação

Uma das decisões que um analista deve tomar é a tecnologia que será utilizada para construir um software. Isso envolve escolher o sistema operacional, o ambiente de execução do software, a linguagem de programação, as ferramentas de desenvolvimento, o banco de dados, a metodologia de desenvolvimento e os profissionais necessários.

Vejamos algumas linguagens existentes:

Clipper	T-SQL
COBOL	Ruby
Visual Basic	ActionScript
Java	Assembly
C++	Pascal
C#	Python
Visual Basic .NET	PHP
JavaScript	Self
VBScript	OCaml
J#	Fortran

Bancos de dados

Uma parte fundamental de uma aplicação é como as informações são gravadas e como são recuperadas.

Existem dois tipos de bancos de dados: de servidor e de arquivo.

Banco de dados de arquivo

Neste tipo de banco de dados, as informações são gravadas diretamente em um ou mais arquivos, e estes são abertos diretamente pelo sistema operacional. Este tipo de banco é ideal para aplicações isoladas, que não utilizam a rede. Ao compartilhar o arquivo em rede, corre-se o risco de corromper o arquivo caso haja uma falta de energia no momento de uma gravação ou mesmo se houver um problema com a placa de rede.

Podemos citar como exemplos de bancos de dados de arquivo: Access, DBF, Excel, Paradox.

Banco de dados de servidor

Neste tipo de banco de dados, um serviço instalado em um computador (chamado servidor) acessa os dados e cuida da integridade dos mesmos. Os aplicativos e usuários acessam o servidor e não os arquivos diretamente. Este tipo de banco de dados é ideal para aplicações multiusuários e para grande volume de dados.

Podemos citar como exemplos de banco de dados de servidor: SQL Server, Oracle, MySQL, PostgreSQL, DB2, Firebird.

Tecnologias e ferramentas

Existem muitas ferramentas disponíveis para ajudar um profissional a desenvolver qualquer parte de um software. As ferramentas que reúnem em um único lugar todos os recursos necessários para criar um software são chamadas IDE ou Ambiente de Desenvolvimento Integrado.

Java

O ambiente Java conta com aplicativos que facilitam o design de aplicações. As ferramentas mais conhecidas são:

- NetBeans;
- Eclipse;
- jEdit;
- JBuilder;
- JDeveloper.

Plataforma .NET

A plataforma .NET (Linguagens VB.NET e C#) conta com muitas ferramentas para criar softwares desde a concepção até a implantação final. Algumas dessas ferramentas são:

- Visual Studio;
- Web Developer;
- C# Express;
- VB Express.

Frameworks

Frameworks são conjuntos de códigos compilados prontos para serem usados em um aplicativo. O uso de um framework pode acelerar o processo de desenvolvimento de um software. A decisão de usar ou não um framework é geralmente polêmica dentro de um ambiente corporativo. De um lado, o framework pode tornar mais ágil o desenvolvimento, mas por outro pode limitar seriamente a construção de algo novo.

Alguns frameworks são: Hibernate / AJAX.NET / Struts

Metodologias de desenvolvimento

Construção de software é uma ciência recente e em fase de estruturação. Várias metodologias de desenvolvimento têm aparecido e desaparecido com o passar dos anos. Na verdade, cada empresa acaba criando uma metodologia própria de desenvolvimento com o passar do tempo, baseada em suas necessidades, disponibilidade de mão de obra, prazos, custos e casos de sucesso e fracasso.

Algumas metodologias famosas são:

Programação estruturada

É uma forma de programa em sequência, priorizando as ações a serem tomadas. Muito popular nos anos 1960, foi sucedida pela Programação orientada a objetos.

RUP

O Rational Unified Process é uma metodologia orientada a objetos cujo ponto principal é a notação UML para definir os procedimentos. É geralmente usado para grandes projetos, quando existe um analista ou uma equipe de analistas que determina minuciosamente o que deve ser feito.

XP

O Extreme Programming é uma metodologia geralmente usada por pequenas equipes de programadores. O ponto principal é a interação da equipe com o cliente e a entrega semanal de unidades de softwares prontas para uso. O XP incorpora nas suas premissas que o objetivo do software é mutável e o cliente pode mudar de ideia no meio do processo. As mudanças são bem-vindas e fazem parte do processo de criação.

Scrum

O método Scrum de desenvolvimento é utilizado em algumas empresas de construção de veículos no Japão e consiste na entrega de pequenos programas funcionais em intervalos regulares e reuniões diárias sobre o andamento.