Linguagem Orientada a Objetos

Encontro 01 - Introdução

Prof. Luiz Augusto Rodrigues
luiz.a.rodrigues@cogna.com.br

Avisos Importantes

Nenhum material será enviado via e-mail. Os materiais serão disponibilizados no **AVA** e no google drive:
https://bit.ly/47qzfvl

Dúvidas, questionamentos, entre outros deverão ser realizados pelo **e-mail** e pelo **Whatsapp** da disciplina**.**

Para ingressar no grupo do **Whatsapp** da disciplina acesse o link a seguir e selecione sua disciplina.
linklist.bio/profluizao_2023-2

A Disciplina de Programação Orientada a Objetos, a partir deste slide, será referenciada pela sigla **LOO**.

Assuntos Abordados

Neste encontro, debateremos os seguintes tópicos:

- Introdução
- Paradigma POO versus Paradigma Estruturado
- Princípios de POO
- Pilares de POO - Abstração
- Pilares de POO - Encapsulamento
- Pilares de POO - Herança
- Pilares de POO - Polimorfismo
- Componentes Essenciais

LOO

Encontro 01

Instruções Importantíssimas!!!

Instruções Importantíssimas!!!

Para este semestre, na disciplina de POO, utilizaremos OBRIGATORIAMENTE as seguintes ferramentas:

- Sistema Operacional Windows 10, Linux ou MacOS.
- Linguagem de Programação Java.
- JDK 17 ou superior
  - https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.exe
- VS Code.

Instruções Importantíssimas!!!

- Git 2.39.2 ou superior
  - https://github.com/git-for-windows/git/releases/download/v2.39.2.windows.1/Git-2.39.2-64-bit.exe
- Github
  - https://github.com/
  - https://desktop.github.com/
- Tutorial em Git e Github
  - Tutorial Git e Github 2022 – Introdução prática para iniciantes
  - Canal DevSuperior
  - https://youtu.be/_hZf1teRFNg

Quem for utilizar Linux ou MacOSx, fale comigo após a aula, ou pelo grupo do WhatsApp, para instruções de instalação.

Instruções Importantíssimas!!!

- Os relatórios não serão utilizados nessa disciplina, por se tratar de uma disciplina 90% prática.
- No lugar, teremos atividades de pós-aula, que deverão ser respondidas e justificadas, para fixação do conteúdo discutido na aula.
- As atividades de aula estarão disponíveis apenas no Github da turma.
- O tutorial do Git e Github é **obrigatório**.

LOO

Encontro 01

Introdução

Introdução

Quando desenvolvemos um sistema, temos sempre um objetivo em mente, criar a solução para um problema.

Partimos com um propósito de chegar a um resultado final que atenda com sucesso à demanda desejada. Por isso, é fundamental entender cada etapa, com muita clareza.

Os passos seguintes para desenvolver uma aplicação são desenhar a arquitetura, o design e os padrões de projeto que serão mais adequados para o desenvolvimento do produto final.

A partir daí, é possível ter mais assertividade na escolha das tecnologias que serão aplicadas. E é aqui que entra a programação orientada a objetos (POO).

Introdução

Paradigmas de programação existem para responder questões que surgem justamente do processo de pensar na abordagem que será utilizada para a solução de problemas.

Enquanto uma linguagem de programação é a implementação de fato de uma ferramenta para desenvolver um software, o paradigma de programação é o modelo conceitual e o conjunto de padrões e metodologias que serão aplicadas no uso de uma linguagem para o desenvolvimento de um software.

A POO é um paradigma de programação que se propõe a abordar o design de um sistema em termos de entidades, os objetos, e relacionamentos entre essas entidades.

Introdução

Não é à toa que a programação orientada a objetos se tornou muito utilizada no mercado de trabalho e famosa na comunidade das pessoas desenvolvedoras. Sua utilização traz diversos benefícios para os códigos, que abordaremos melhor mais para frente nesse mesmo artigo. Alguns desses benefícios são o encapsulamento e a reutilização de códigos, os tornando mais legíveis e trazendo uma maior produtividade para quem está desenvolvendo.

Exatamente por se tornar algo mundialmente utilizado e que vem ganhando cada vez mais força, a sua implementação e estudo se tornou fundamental para qualquer desenvolvedor.

LOO

Encontro 01

Paradigma POO versus Paradigma Estruturado

Paradigma POO versus Paradigma Estruturado

Na programação estruturada, a abordagem utilizada para o desenvolvimento de sistemas é procedural, ou seja, pensamos em um programa de computador como uma rotina ou sequência lógica, com início e fim determinados.

Os programas são usados para processar uma entrada, alterando seus dados até que a saída desejada seja produzida.

A programação estruturada é composta por três tipos de estruturas básicas: sequências (os comandos a serem executados), condições (blocos de decisão) e repetições (sequências que devem ser repetidas até chegar a determinada condição).

Esta abordagem pode se tornar problemática à medida que os sistemas começam a ficar complexos. O acesso às variáveis se torna um problema, pois sub-rotinas devem compartilhar essas variáveis para processá-las.

Paradigma POO versus Paradigma Estruturado

O reuso de rotinas fica muito difícil, na programação estruturada, obrigando códigos que desempenham essencialmente a mesma função tenham de ser reescritos.

A legibilidade e a organização do código também é prejudicada, pois os programas passam a ficar muito longos e praticamente ilegíveis.

Na orientação a objetos, por outro lado, uma variável será vista como um atributo de um objeto, algo que ele "tem". Um funcionário tem um nome. As funções passarão a ser as responsabilidades do objeto.

Se quisermos uma lista dos funcionários de um departamento, a função que retorna esses dados será responsabilidade da classe Departamento.

Paradigma POO versus Paradigma Estruturado

A linguagem C é a principal representante da programação estruturada. Se trata de uma linguagem considerada de baixo nível, que atualmente não é utilizada para projetos muito grandes. A sua principal utilização, devido ao baixo nível, é em programação para sistemas embarcados ou outros em que o conhecimento do hardware se faz necessário para um bom programa.

Essa colocação nos traz a um detalhe importante: a programação estruturada, quando bem feita, possui um desempenho superior ao que vemos na programação orientada a objetos. Isso ocorre pelo fato de ser um paradigma sequencial, em que cada linha de código é executada após a outra, sem muitos desvios, como vemos na POO. Além disso, o paradigma estruturado costuma permitir mais liberdades com o hardware, o que acaba auxiliando na questão desempenho.

Paradigma POO versus Paradigma Estruturado

Entretanto, a programação orientada a objetos traz outros pontos que acabam sendo mais interessantes no contexto de aplicações modernas. Como o desempenho das aplicações não é uma das grandes preocupações na maioria das aplicações (devido ao poder de processamento dos computadores atuais), a programação orientada a objetos se tornou muito difundida. Essa difusão se dá muito pela questão da reutilização de código e pela capacidade de representação do sistema muito mais perto do que veríamos no mundo real.

Mais à frente, vamos entender estas e mais características da programação orientada a objetos e como ela facilita o desenvolvimento de sistemas conceitualmente mais próximos da situação do mundo real que pretendemos modelar.

Paradigma POO versus Paradigma Estruturado

LOO

Encontro 01

Princípios de POO

Princípios de POO

O que são princípios?

Princípio significa o início, fundamento ou essência de algum fenômeno.

Também pode ser definido como a causa primária, o momento, o local ou trecho em que algo, uma ação ou um conhecimento tem origem.

Sendo que o princípio de algo, seja como origem ou proposição fundamental, pode ser questionado.

Princípios de POO

Para uma linguagem ser considerada no paradigma do POO, existem quatro características principais que precisam fazer parte de sua aplicação:

- Abstração,
- Encapsulamento,
- Herança
- Polimorfismo.

Princípios de POO

LOO

Encontro 01

Pilares de POO - Abstração

Pilares de POO - Abstração

A abstração consiste em um dos pontos mais importantes dentro de qualquer linguagem Orientada a Objetos. Como estamos lidando com uma representação de um objeto real (o que dá nome ao paradigma), temos que imaginar o que esse objeto irá realizar dentro de nosso sistema. São três pontos que devem ser levados em consideração nessa abstração.

Pilares de POO - Abstração

O primeiro ponto é darmos uma identidade ao objeto que iremos criar.

Essa identidade deve ser única dentro do sistema para que não haja conflito. Na maior parte das linguagens, há o conceito de pacotes (ou namespaces).

Nessas linguagens, a identidade do objeto não pode ser repetida dentro do pacote, e não necessariamente no sistema inteiro.

Nesses casos, a identidade real de cada objeto se dá por:

```
<nome_do_pacote>.<nome_do_objeto>.</nome_do_objeto></nome_do_pacote>
```

Pilares de POO - Abstração

A segunda parte diz respeito a características do objeto. Como sabemos, no mundo real qualquer objeto possui elementos que o definem. Dentro da programação orientada a objetos, essas características são nomeadas propriedades.

Por exemplo, as propriedades de um objeto "Cachorro" poderiam ser "Tamanho", "Raça" e "Idade".

Cachorro
Tamanho Raça Idade

Pilares de POO - Abstração

Por fim, a terceira parte é definirmos as ações que o objeto irá executar. Essas ações, ou eventos, são chamados métodos.

Esses métodos podem ser extremamente variáveis, desde "Acender()" em um objeto lâmpada até "Latir()" em um objeto cachorro.

Lâmpada
Potência Cor
Acender()

Cachorro
Tamanho Raça Idade
Latir()

Pilares de POO - Abstração

LOO

Encontro 01

Pilares de POO - Encapsulamento

Pilares de POO - Encapsulamento

O encapsulamento é uma das principais técnicas que define a programação orientada a objetos.

Se trata de um dos elementos que adicionam segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta.

Pilares de POO - Encapsulamento

A maior parte das linguagens orientadas a objetos implementam o encapsulamento baseado em propriedades privadas, ligadas a métodos especiais chamados **getters** e **setters**, que irão retornar e setar o valor da propriedade, respectivamente.

Essa atitude evita o acesso direto a propriedade do objeto, adicionando uma outra camada de segurança à aplicação.

Para fazermos um paralelo com o que vemos no mundo real, temos o encapsulamento em outros elementos.

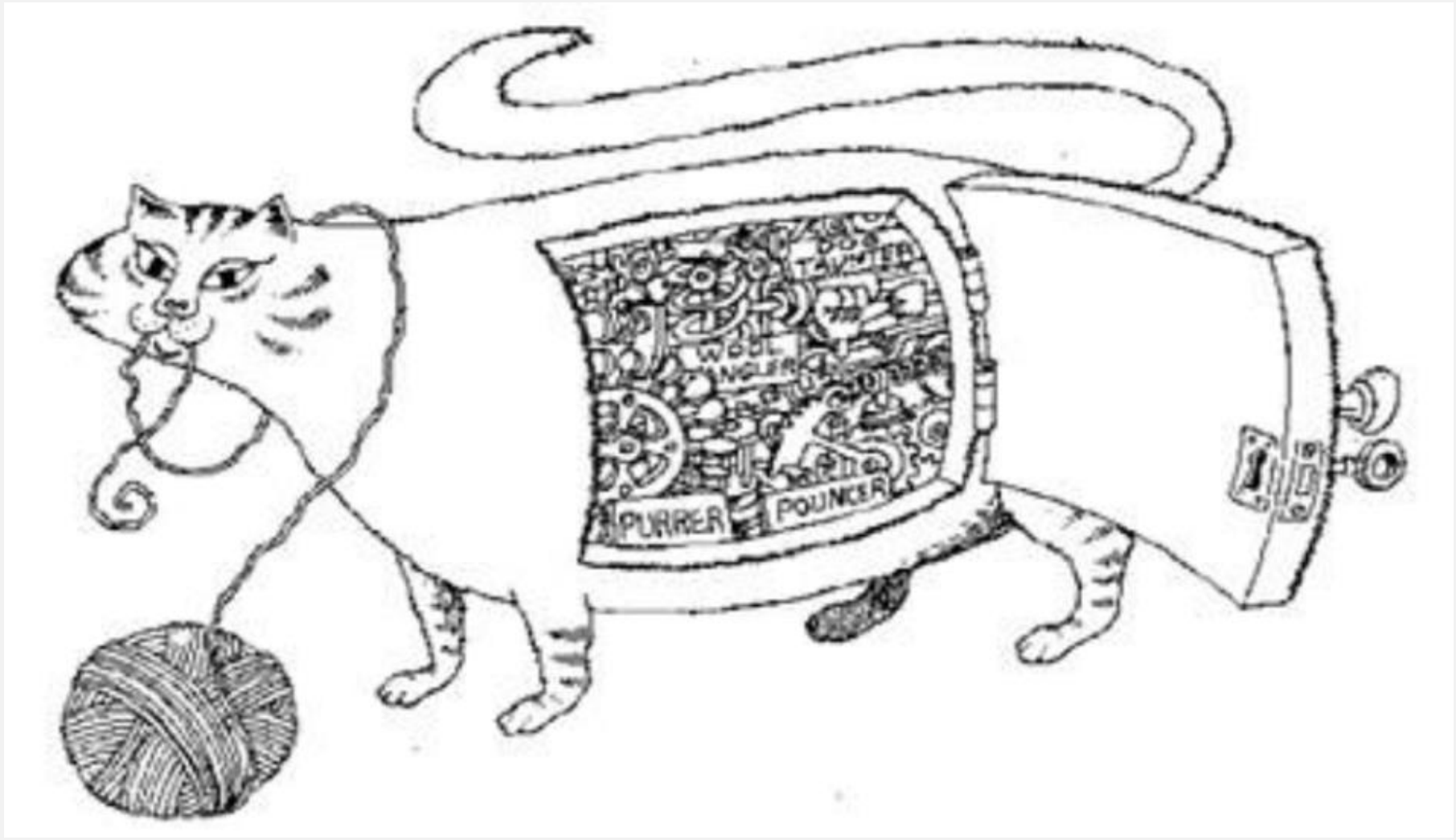
Pilares de POO - Encapsulamento

Por exemplo, quando clicamos no botão ligar da televisão, não sabemos o que está acontecendo internamente. Podemos então dizer que os métodos que ligam a televisão estão encapsulados.

Aqui usamos algumas palavras reservadas para definir o encapsulamento. Estas palavras são:

- **Private** – privado, que deve ser visualizado e manipulado apenas pelo próprio objeto.
- **Public** – público, que pode ser visualizado e manipulado pelo próprio objeto e por outros que o acessem.
- **Protected** – protegido, que pode ser visualizado e manipulado pelo próprio objeto e por outros que sejam derivados (classes filhas). Relacionado com o pilar de herança.

Pilares de POO - Encapsulamento



LOO

Encontro 01

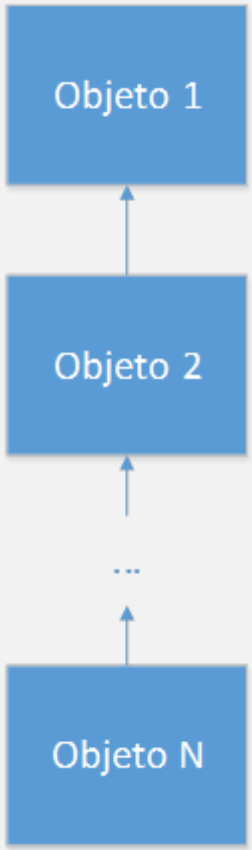
Pilares de POO - Herança

Pilares de POO - Herança

O reuso de código é uma das grandes vantagens da programação orientada a objetos. Muito disso se dá por uma questão que é conhecida como herança. Essa característica otimiza a produção da aplicação em tempo e linhas de código.

Para entendermos essa característica, vamos imaginar uma família: a criança, por exemplo, está herdando características de seus pais. Os pais, por sua vez, herdam algo dos avós, o que faz com que a criança também o faça, e assim sucessivamente. Na orientação a objetos, a questão é exatamente assim.

Pilares de POO - Herança



O objeto mais acima, também chamado de objeto pai, é considerado mais generalizado, isto é, seus métodos e propriedades podem ser melhorados.

O objeto abaixo na hierarquia irá herdar características de todos os objetos acima dele, seus "ancestrais", assim podendo especializar seus métodos e propriedades.

A herança a partir das características do objeto imediatamente acima é considerada herança direta, enquanto as demais são consideradas heranças indiretas.

Por exemplo, na família, a criança herda diretamente do pai e indiretamente do avô e do bisavô.

Pilares de POO - Herança

Importante

Aqui temos o uso da palavra reservada **Protected** (protegido).

Como dito anteriormente, os atributos, propriedades e métodos assinaladas com esse nível de encapsulamento podem ser acessados na classe pai, sem necessariamente serem especializados.

Falaremos mais a respeito quando debatermos classes abstratas.

LOO

Encontro 01

Pilares de POO - Polimorfismo

Pilares de POO - Polimorfismo

Outro ponto essencial na programação orientada a objetos é o chamado polimorfismo.

Na natureza, vemos animais que são capazes de alterar sua forma conforme a necessidade, e é dessa ideia que vem o polimorfismo na orientação a objetos. Como sabemos, os objetos filhos herdam as características e ações de seus "ancestrais".

Entretanto, em alguns casos, é necessário que as ações para um mesmo método seja diferente. Em outras palavras, o polimorfismo consiste na alteração do funcionamento interno de um método herdado de um objeto pai.

Pilares de POO - Polimorfismo

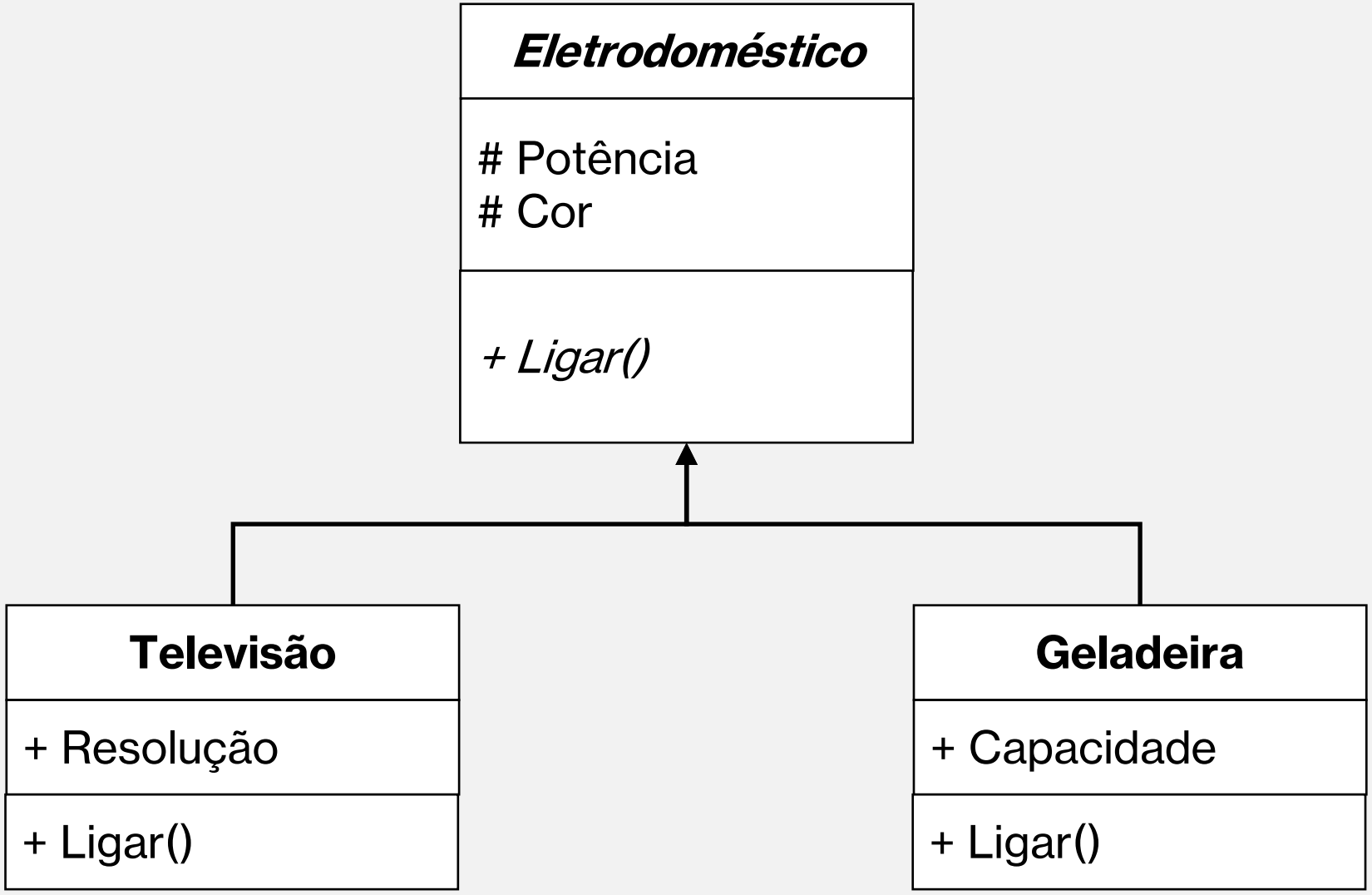
Como um exemplo, temos um objeto genérico “Eletrodoméstico”.

Esse objeto possui um método, ou ação, “Ligar()”.

Temos dois objetos, “Televisão” e “Geladeira”, que não irão ser ligados da mesma forma.

Assim, precisamos, para cada uma das classes filhas, reescrever o método “Ligar()”.

Pilares de POO - Polimorfismo



Pilares de POO - Polimorfismo

Com relação ao polimorfismo, valem algumas observações:

- Como se trata de um assunto que está intimamente conectado à herança, entender os dois juntamente é uma boa ideia.
- Outro ponto é o fato de que as linguagens de programação implementam o polimorfismo de maneiras diferentes.
- O C#, por exemplo, faz uso de método virtuais (com a palavra-chave **virtual**) que podem ser reimplementados (com a palavra-chave **override**) nas classes filhas.
- Já em Java, apenas o atributo "@Override" é necessário.

Importante

Alguns símbolos e nomenclaturas da UML (abordaremos mais profundamente no decorrer da disciplina):

+ Público

- Privado

Protegido

Classes genéricas, ou classes base, são escritas em itálico.

Classes concretas, ou classes reais (filhas ou não), são escritas normalmente.

LOO

Encontro 01

Componentes Essenciais de POO

Componentes Essenciais de POO

Classes

Classes são estruturas das linguagens de POO criadas para conter os dados que devem ser representados e as operações que devem ser efetuadas com esses dados para determinado modelo.

De modo mais direto, uma classe é uma estrutura que abstrai um conjunto de objetos com características similares, define o comportamento de seus objetos (através de métodos), e os estados possíveis destes objetos (através de atributos).

Analogia com a forma de bolo.

Componentes Essenciais de POO

Objetos

Objetos são instâncias de classes, que determinam qual informação um objeto contém e como ele pode manipulá-la.

Trata-se de uma entidade capaz de reter um estado (informação, dados) e que oferece uma série de informações (comportamento), ou para examinar ou para afetar este estado.

É através deles que praticamente todo o processamento ocorre em sistemas implementados com linguagens de programação orientada a objetos.

Componentes Essenciais de POO

Atributos

Atributos são características de um objeto, nos quais são armazenados os dados e informações relevantes ao objeto em si, seus tipos de dados.

Como vimos anteriormente, os atributos podem ser categorizados em:

- Privados (somente o próprio objeto conhece e manipula);
- Públicos (acessíveis por qualquer objeto terceiro);
- Protegidos (apenas as classes derivadas, ou filhas, pode acessar e manipular).

Componentes Essenciais de POO

Propriedades

Propriedades são estruturas em uma classe que definem o acesso a informações dentro dela.

Uma propriedade é composta por dois métodos especiais:

GET - método para retornar a informação;

SET - método para definir o valor da propriedade.

Geralmente, para cada atributo existe um par de métodos que permitem acesso e manipulação, direta ou indiretamente (tratamentos especiais).

Existem situações nas quais temos apenas um método GET, que permite apenas obter o valor do atributo (chamado de Somente leitura).

Componentes Essenciais de POO

Métodos

Os métodos são procedimentos ou funções que realizam as ações próprias do objeto.

Assim, os métodos são as ações que o objeto pode realizar.

Tudo o que o objeto pode fazer só é possível através de seus métodos, pois é por meio deles que um objeto se manifesta, ou seja, interage com os outros objetos.

Componentes Essenciais de POO

Métodos Construtores

Partindo da mesma ideia de método, todo o objeto precisa de algo que possa "dar um boot", isto é, que possa ser usado para inicializar o objeto, processo esse denominado criação de instância de objetos.

A esse método especial damos o nome de Construtor. Ele é o primeiro método a ser invocado, e no qual podemos inicializar todos os atributos necessários para criar uma nova instância do objeto.

Aqui temos uma característica importante: toda classe tem um construtor vazio, e outros que podem manipular os dados durante a criação da instância, graças ao pilar de Polimorfismo.

Componentes Essenciais de POO

Instância de objeto

Também chamado por alguns autores de instância de uma classe, trata-se de um objeto recém criado, cujo comportamento e estado são definidos por uma classe.

As novas instâncias compartilham o mesmo conjunto de atributos, embora sejam diferentes quanto ao conteúdo desses atributos. Por exemplo, uma geladeira pode ser amarela, outra pode verde, outra pode ser branca, e assim por diante.

Uma nova instância de objeto é a concretização de uma classe. Em termos intuitivos, uma classe é como um "molde" que gera instâncias de um certo tipo; um objeto é algo que existe fisicamente e que foi "moldado" pela classe.

Exemplo

Um bom exemplo de classes e objetos, aplicando todas as características descritas aqui, pode ser o caso da forma de bolo.

Uma forma de bolo redonda pode criar bolos redondos, mas cada bolo criado pode ser de um sabor diferente.

Uma forma de bolo quadrada também pode fazer a mesma coisa, isto é, criar diversos bolos, cada um de um sabor diferente.

Porém, apesar de formatos diferentes, sabores diferentes, recheios diferentes, todos são BOLOS.

LOO

Encontro 01

Atividade de Aula

Atividade de Aula

1. Assistir ao Tutorial do Git e Github:

Tutorial Git e Github 2022 – Introdução prática para iniciantes

Canal DevSuperior

https://youtu.be/_hZf1teRFNg

2. Acessar a pasta do Github da disciplina, utilizando o Git ou o Github Desktop:

Repositório da turma no Github

https://github.com/profluizao/loo-2023-2

Atividade de Aula

3. Responder a atividade disponível apenas no Github ou no Google Drive da disciplina (apenas no Encontro 01).

4. Enviar as respostas justificadas no e-mail do professor, em formato PDF, com o seguinte assunto:

[LOO] ATIVIDADE 01

5. Data de entrega: 14/08/2023, até às 23h00.

Encerramento

Dúvidas e sugestões, entre em contato pelo whatsapp da disciplina, ou mande um e-mail para luiz.a.rodrigues@cogna.com.br