

Capacidades da Unidade Curricular

- Reconhecer ferramentas para o desenvolvimento de atividades (repositório, controle de versão)
- Instalar ferramentas de acordo com requisitos de hardware, software e parâmetro de configuração.
- Reconhecer especificações técnicas e paradigmas de linguagem de programação
- Aplicar linguagem de programação por meio do ambiente integrado de desenvolvimento (IDE)
- Integrar banco de dados por meio da linguagem de programação

Capacidade Unidade Curricular

- Aplicar métodos e técnicas de programação
- Empregar comentários para documentação do código fonte
- Utilizar o ambiente de desenvolvimento (IDE) para rastreabilidade do código
- Identificar erros de acordo com o requisito do programa
- Utilizar o ambiente de desenvolvimento (IDE) para aplicação de teste unitário

Conhecimentos

- Linguagem de programação estruturada
- Linguagem de programação orientada a objetos
- Técnicas de programação
- Modelagem de Negócios
- Gestão da Qualidade
- Trabalho e profissionalismo
- Ética profissional
- Conexão com banco de dados
- Preparação do ambiente
- Programação de Aplicativos

Conceitos Básicos

Um algoritmo é uma sequência de instruções ou comandos realizados de maneira sistemática com o objetivo de resolver um problema ou executar uma tarefa.

Lógica de Programação é o modo como se escrevem programas de computador através de uma sequência de passos para executar uma ou várias funções, esta sequência é o algoritmo.

POO



Análise orientada à Objeto - POO

A Análise Orientada a Objetos (OOA) é um processo de desenvolvimento de sistemas que utiliza o conceito de objetos que interagem entre si e, através dessa interação, realizam tarefas computacionais.

O ponto de partida para a OOA é criar um modelo descritivo contendo informações do projeto.

Programação orientada à Objeto - POO

A programação orientada a objetos é um modelo de programação onde diversas classes possuem características que definem um objeto na vida real.

Cada classe determina o comportamento do objeto definido por métodos e seus estados possíveis definidos por atributos.

Programação orientada à Objeto - POO

Na análise e no projeto OO, estamos interessados no comportamento do objeto.

As operações são codificadas como métodos.

A representação de software OO do objeto é, dessa forma, uma coleção de tipos de dados e métodos.

No software OO: Um objeto é qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e os métodos que os manipulam.

Programação orientada à Objeto - POO

Os principais conceitos do paradigma orientado a objetos são:

Classes

Objetos

Associação

Encapsulamento

Herança

Polimorfismo.

Programação orientada à Objeto - POO

Classe

O termo classe refere-se à implementação de software de um tipo de objeto.

Um tipo de objeto especifica uma família de objetos sem estipular como o tipo e o objeto são implementados.

Programação orientada à Objeto - POO

Métodos

Os métodos especificam a maneira pela qual os dados de um objeto são manipulados.

Uma especificação dos passos pelos quais uma operação deve ser executada.

Ele é um script de implementação de uma operação.

Diferentes métodos podem ser usados para executar a mesma operação.

Os métodos de um tipo de objeto referenciam somente as estruturas de dados desse tipo de objeto.

A ação que um objeto ou uma classe podem desempenhar.

Os métodos são similares às funções e procedures do universo da programação estruturada.

Programação orientada à Objeto - POO

Cada conceito é uma ideia ou um entendimento pessoal que temos do nosso mundo. Os conceitos que adquirimos nos permitem dar sentido sobre as coisas do nosso mundo. Essas coisas às quais nossos conceitos se aplicam são denominados **objetos**.

Um objeto pode ser real ou abstrato.

Os **objetos** possuem informações (contém dados) e desempenham ações (possuem funcionalidade).

Qualquer coisa à qual um conceito ou tipo de objeto se aplica – uma instância de um conceito ou tipo de objeto.

Um objeto é uma instância de uma classe.

Programação orientada à Objeto - POO

Encapsulamento

O ato de empacotar ao mesmo tempo dados e objetos é denominado encapsulamento.

O objeto esconde seus dados de outros objetos e permite que os dados sejam acessados por intermédio de seus próprios métodos.

Isso é chamado de ocultação de informações.

Programação orientada à Objeto - POO

O **encapsulamento** protege os dados do objeto do uso arbitrário e não-intencional.

O encapsulamento é o resultado (ou ato) de ocultar do usuário os detalhes da implementação de um objeto.

O encapsulamento é importante porque separa a maneira como um objeto se comporta da maneira como ele é implementado.

A definição de como implementar os conhecimentos ou ações de uma classe, sem informar como isto é feito.

Programação orientada à Objeto - POO

Herança

É comum haver similaridades entre diferentes classes. Frequentemente, duas ou mais classes irão compartilhar os mesmos atributos e/ou métodos.

Como nenhum de nós deseja reescrever várias vezes o mesmo código, seria interessante se algum mecanismo pudesse tirar proveito dessas similaridades.

A herança é esse mecanismo.

Classe Forma:

- método abstrato:
- desenhar()

Classe Circulo herda Forma:

- propriedades:
 - raio
- método:
 - desenhar():

Exibir mensagem "Desenhando um círculo com raio " + raio

Classe Quadrado herda Forma:

- propriedades:

 - lado

- método:

 - desenhar():

Exibir mensagem "Desenhando um quadrado com lado " + lado

```
// Função para desenhar qualquer forma
```

```
Função desenharForma(forma: Forma):  
forma.desenhar()
```

```
// Criando objetos
```


```
circulo = Circulo(raio = 5)
```

```
quadrado = Quadrado(lado = 10)
```

```
// Usando polimorfismo
```

```
desenharForma(circulo) // "Desenhando um círculo com raio 5"
```

```
desenharForma(quadrado) // "Desenhando um quadrado com lado 10"
```

A large, irregular pink brushstroke shape serves as a background for the text. It has a soft, painterly texture with visible brush marks and a slightly darker pink outline.

*A maneira de começar
é parar de falar e
começar a fazer.*

Walt Disney

Atividade

Em uma folha colocar o nome dos colegas para entrega da atividade.

Atividade 1:

Discutir em grupo quais as etapas e procedimentos para preparar (fazer) um CAFÉ.



1. Classes e Objetos

– Classe `Cafeteira`:

- Representa a cafeteira.
- Possui atributos como capacidade, tipo de filtro e modelo.
- Métodos podem incluir `aquecerAgua()`, `prepararCafe()`, etc.

– Classe `Café`:

- Representa o café em pó.
- Atributos podem ser tipo, quantidade e intensidade.
- Métodos incluem `moerCafe()`, `medirQuantidade()`, etc.

– Classe `Água`:

- Representa a água.
- Atributos incluem temperatura e volume.
- Métodos poderiam ser `ferver ()`, `adicionarAgua()`.

– Classe `Xícara`:

- Representa a xícara que vai receber o café.
- Atributos incluem material e volume.

2. Instância de Objetos

- Criar Objetos:
- ``cafeteira = new Cafeteira ();``
- ``cafe = new Café ();``
- ``agua = new Água ();``
- ``xicara = new Xícara ();``

Esses objetos serão instanciados com os atributos e comportamentos definidos pelas classes.

3. Métodos e Interações

– Aquecer a Água:

- ``agua.aquecer();``

A água é aquecida utilizando o método ``aquecer()`` do objeto ``água``.

– Preparar o Café:

- ``cafeteira.prepararCafe(cafe, agua);``

O método ``prepararCafe()`` da classe ``Cafeteira`` utiliza os objetos ``Café`` e ``Água`` como parâmetros para preparar o café.

– Servir o Café:

- ``xicara.receberCafe(cafeteira);``

O método ``receberCafe()`` da classe ``Xícara`` recebe o café preparado pela ``Cafeteira``.

4. Herança

Suponha que você tenha diferentes tipos de cafeteiras, como `CafeteiraAutomatica`` e `CafeteiraManual``.

Ambas herdam da classe `Cafeteira``, mas podem ter métodos específicos ou sobrescritos:

- `CafeteiraAutomatica`` pode ter um método adicional `programarHorario()``.
- `CafeteiraManual`` pode sobrescrever o método `prepararCafe()`` para incluir a etapa de filtragem manual.

5. Encapsulamento

- Os detalhes de implementação de como a água é aquecida ou como o café é filtrado são encapsulados dentro das classes.

O usuário (programador) só precisa chamar os métodos públicos, como `prepararCafe()`, sem se preocupar com os detalhes internos.

6. Polimorfismo

Imagine que você tenha um método `fazerCafe(Cafeteira cafeteira)`.

Se você passar diferentes tipos de objetos `Cafeteira` (por exemplo, `CafeteiraAutomatica` ou `CafeteiraManual`), o método `prepararCafe()` será executado de acordo com o tipo específico de cafeteira, demonstrando o conceito de polimorfismo.

7. Abstração

A ideia geral de fazer café é abstraída em classes e métodos.

O usuário não precisa entender todos os detalhes físicos, como a física por trás do aquecimento da água, mas apenas os conceitos abstratos necessários para a preparação do café.

Concluindo

Essa analogia mostra como o conceito de POO pode ser aplicado para simplificar e organizar o processo de fazer café.

Tornando-o:

- mais modular,
- reutilizável,
- fácil de entender.

A large, irregular pink brushstroke shape that serves as a background for the title text. The stroke is thick and has a textured, hand-painted appearance with some darker and lighter shades of pink.

Ciclo de Vida de Software

Ciclo de Vida de Software

"O Ciclo de Vida de Desenvolvimento de Sistemas é um processo utilizado por um analista de sistemas para desenvolver um sistema de informação."

A principal função do ciclo de vida do desenvolvimento de software é indicar as fases, atividades, entregas e responsabilidades de cada envolvido no processo de desenvolvimento de software.

Ciclo de Vida de Software

- 1- **Estudo de Viabilidade ou Fase de requisitos** : levanta os requisitos mínimos, estuda a viabilidade e define o modelo a ser usado;
- 2- **Fase de projeto** : Envolve atividades de concepção, especificação, design da interface, prototipação, design da arquitetura;
- 3- **Fase de implementação** : tradução para uma linguagem de programação das funcionalidades definidas durante as fases anteriores;
- 4- **Fase de testes** : realização de testes no que foi desenvolvido de acordo com os requisitos;
- 5- **Fase de produção** : implantação em produção do produto final;

Diferenças do Ciclo de Vida

Paradigma Tradicional:

Análise (Pouca),
Projeto (Pouco),
Codificação (Muita),
Teste (Muito,
geralmente
confundido com
codificação),
Manutenção
(Muita);

Paradigma OO:

Análise (Muita)
Projeto (Muito)
Re-Análise, Re-Projeto, Codificação
(Simplificada), Re-Análise, Re-
Projeto, Re-Codificação, Teste
(Reduzido), Re- Análise, etc...

Projeto Orientado à objeto

No Projeto OO você usa conceitos de Engenharia de Software para produzir o produto, e não somente como documentação.

Engenharia de software é uma área da engenharia e da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade.



Como o cliente explicou



Como o lider de projeto entendeu



Como o analista planejou



Como o programador codificou



O que os beta testers receberam



Como o consultor de negocios descreveu



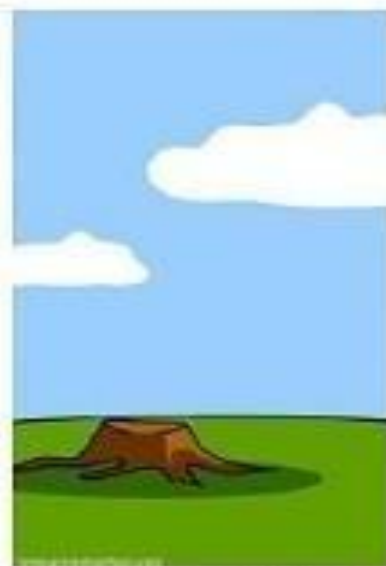
Valor que o cliente pagou



Como o projeto foi documentado



O que a assistencia tecnica instalou



Como foi suportado



Quando foi entregue



O que o cliente realmente necessitava

Atividade Avaliativa

Realizar pesquisa qualitativa sobre os temas descritos abaixo, posteriormente enviar o resultado da pesquisa para e-mail marcio.senhorinha@edu.sc.senai.br com título **Atividade Avaliativa 1 POO - Turma**

Por que a programação orientada a objetos é importante?

Qual a diferença entre programação orientada a objetos e programação estruturada?

O que são classes e objetos dentro da programação orientada a objetos?

Quais são os pilares da programação orientada a objetos?

Quais são os benefícios de se utilizar programação orientada a objetos?