THAMIRIS ADRIANO

PDS TECH

FULL STACK DEVELOPMENT

TYPESCRIPT

AULA 02

SUMÁRIO

O QUE VEM POR AÍ?

Nessa aula, as pessoas docentes apresentarão aspectos relacionados a alguns paradigmas de programação e em seguida discutirão o paradigma orientado a objetos.

Você vai compreender o motivo da importância desses conceitos, que formam o suporte para a criação de programas que se baseiam em princípios como classes, objetos, herança, polimorfismo, encapsulamento e outros tópicos essenciais.

HANDS ON

No primeiro vídeo, os(as) docentes apresentaram aos(às) estudantes os fundamentos da Programação Orientada a Objetos (POO) com foco em TypeScript. Explicaram também o que é um paradigma de programação, destacando os principais paradigmas existentes. Em seguida, se aprofundaram na Programação Orientada a Objetos, explicando como a POO se baseia na modelagem do mundo real usando objetos com propriedades e métodos.

Já na segunda videoaula, os(as) professores(as) continuaram explorando a Programação Orientada a Objetos, concentrando-se nos conceitos de herança e interfaces. Em seguida, foi abordado o uso de interfaces em TypeScript, enfatizando como elas definem contratos para classes e garantem a implementação correta de métodos e propriedades.

SAIBA MAIS

Paradigmas de Programação

Um paradigma de programação representa a estrutura organizacional e o conjunto de princípios que orientam o desenvolvimento de software em uma linguagem específica. Essa abordagem define não apenas a sintaxe da linguagem, mas também a maneira como você organiza e interage com o código. Em essência, o paradigma de programação é o modo como uma linguagem pensa e se comporta. Ele desempenha um papel fundamental na definição da lógica e na forma como os programas são construídos e compreendidos. Portanto, compreender o paradigma adotado por uma linguagem é fundamental para desenvolvedores(as), pois influencia diretamente a abordagem e a solução de problemas durante o desenvolvimento de software.

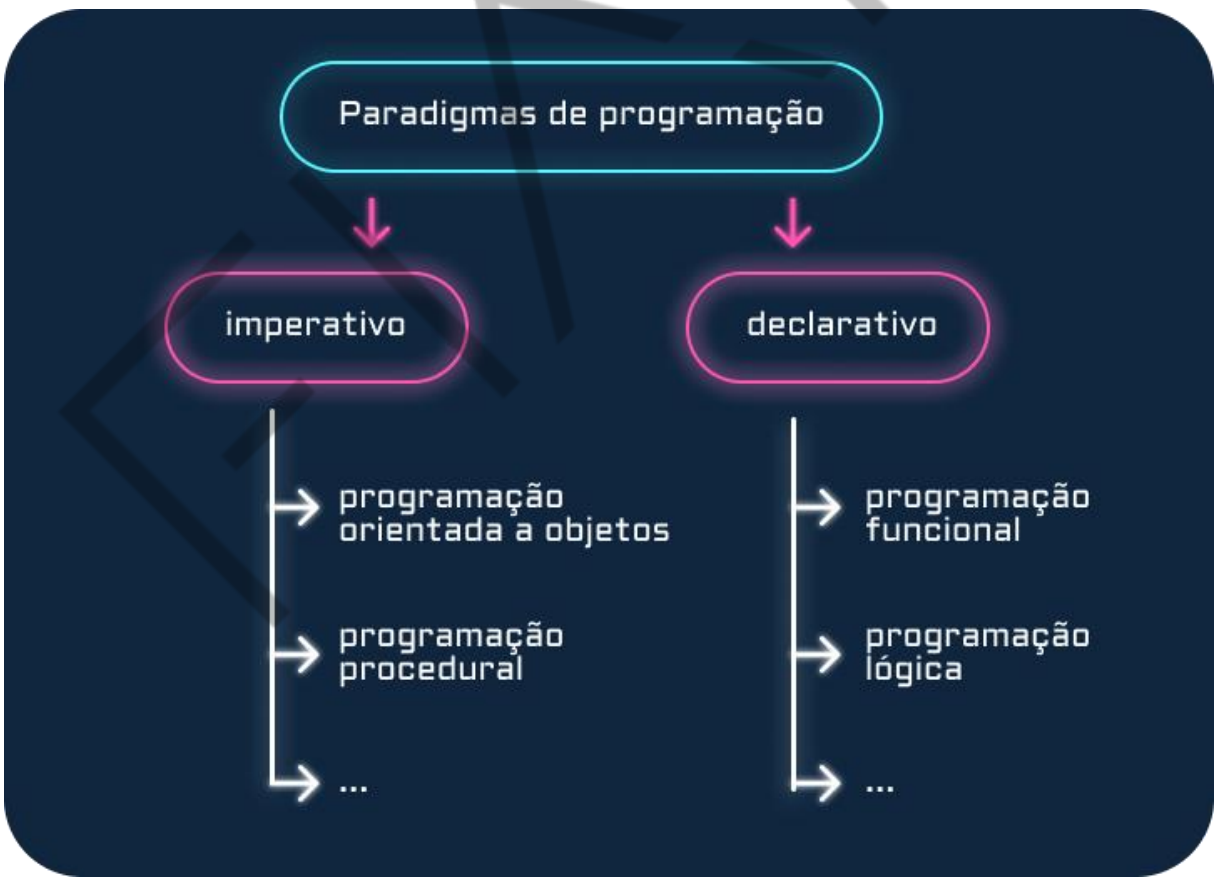

Figura 1 – Paradigmas de programação

Fonte: Sawa Studio (2022)

Os paradigmas de programação representam abordagens fundamentais para o desenvolvimento de software, cada um com sua própria filosofia e estilo. Existem diversos tipos de paradigmas, sendo os principais: imperativo, declarativo, funcional, lógico, orientado a objetos e orientado a eventos. Cada um deles oferece uma maneira distinta de encarar a construção de programas.

Paradigma Imperativo: como sugere o nome, a pessoa desenvolvedora especifica instruções para a máquina sobre como processar as operações de uma determinada forma. Dentro desse paradigma, encontramos duas subcategorias significativas:

Paradigma Procedural: este é um dos paradigmas mais comuns e é ideal para programação geral. Nele, o código é organizado como uma lista de instruções que a máquina executa sequencialmente, uma após a outra. Linguagens de programação como C e C++ seguem essa abordagem. É apropriado quando as operações são complexas, há dependências entre as execuções e há necessidade de visibilidade. Também é útil quando o programa é relativamente estático, com poucas mudanças planejadas no futuro.

Paradigma Orientado a Objeto: esse paradigma é amplamente aplicado e valorizado por vantagens como modularidade do código e capacidade de representar relações do mundo real no código. Linguagens como PHP, Java, Ruby, Python e C# são exemplos de linguagens que seguem esse paradigma. É particularmente útil quando muitos(as) desenvolvedores(as) trabalham em conjunto no projeto e não precisam entender profundamente cada componente, bem como quando o projeto prevê muitas mudanças no futuro. É nele que vamos focar nessa aula.

Agora, vamos explorar os paradigmas declarativos.

Paradigma Declarativo: ao contrário do paradigma imperativo, onde as instruções detalham o processo de execução passo a passo, o paradigma declarativo concentra-se em descrever o resultado desejado sem especificar o caminho para alcançá-lo. Nesse paradigma, as pessoas desenvolvedoras declaram as propriedades, relações e restrições do problema a ser resolvido, permitindo que a máquina determine a melhor abordagem para atender a essas especificações. Isso é especialmente útil em cenários onde a lógica é complexa e as regras de negócios

desempenham um papel fundamental. Dois exemplos de paradigmas declarativos são:

Paradigma Funcional: nesse paradigma, os programas são construídos com base em funções matemáticas puras que não possuem efeitos colaterais. Linguagens como Haskell, Lisp e Erlang adotam essa abordagem. É especialmente eficaz quando se lida com cálculos e transformações de dados, tornando o código mais conciso e legível.

Paradigma Lógico: no paradigma lógico, a lógica é usada para resolver problemas. A linguagem Prolog é um exemplo notável desse paradigma. Ele é especialmente útil em sistemas de suporte a decisões e aplicativos que envolvem regras complexas, como sistemas especialistas e consultas de banco de dados complexas.

Em resumo, os paradigmas declarativos se destacam por permitirem a descrição das características desejadas de um problema, deixando a máquina encarregada de encontrar a solução. Isso pode simplificar a escrita de código e tornar a manutenção e a compreensão mais acessíveis em situações que envolvem complexidade lógica.

POO - Programação orientada a objetos

A programação orientada a objetos, também conhecida pela sigla POO, facilita a relação do computador com o mundo real. Qualquer coisa que pertence ao mundo físico pode ser representada no mundo digital. A essas “coisas” damos o nome de “objeto” na POO.

No contexto do software, um objeto é tratado com atribuições específicas e funções designadas. Após passar por todas as etapas necessárias, o objeto gera uma saída que tem impacto no mundo real.

Fazendo uma simulação com algo do mundo real, vamos imaginar que precisamos organizar um guarda-roupa.

Figura 2 – Guarda-roupa
Fonte: Freepik (s.d.)

Devemos, então, considerar as roupas como item principal que teremos de lidar; sendo assim, elas poderiam ser os nossos objetos. Porém, ao falar de roupas, estamos falando de uma categoria específica de objetos, uma **classe**. Quando começamos a organizar o guarda-roupa, vemos que a classe roupas não é específica o suficiente, pois dentro desta classe temos muitas variáveis, como camisetas, sapatos, calças, entre outras, e queremos dar um mínimo de organização para que os itens semelhantes fiquem agrupados juntos. A solução para isso é separar as classes de acordo com as categorias das roupas, como meias, roupas de corrida, roupas de sair, etc.

Quando formos procurar uma blusa de corrida específica, sabemos que não devemos procurar na gaveta de meias, por exemplo, mas na parte de roupas de corrida, que é sua classe. Os **atributos** de determinadas peças de roupas as tornam únicas e conferem a elas o título de objeto.

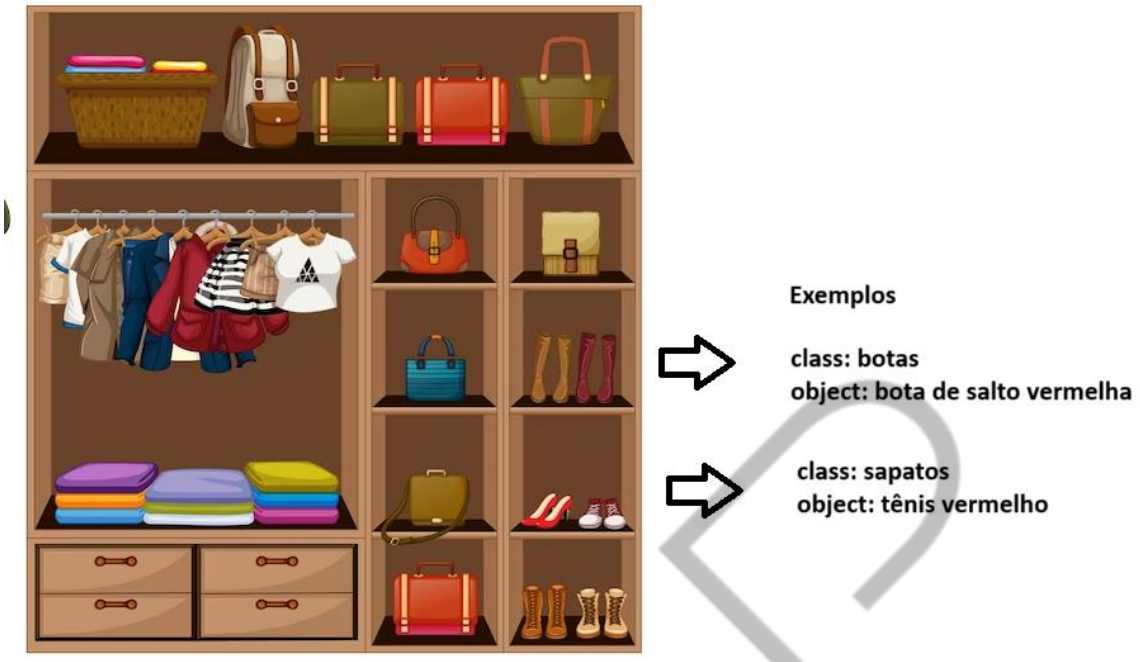

Figura 3 – Guarda-roupa 2

Fonte: Freepik (s.d.), adaptado por FIAP (2024)

Podemos aplicar a abordagem da Programação Orientada a Objetos não apenas à organização de roupas, mas também a uma ampla gama de atividades cotidianas. A essência da modelagem orientada a objetos é enxergar o mundo como um conjunto de objetos que se relacionam e possuem características distintas, bem como comportamentos específicos, expressos por meio de seus atributos e operações. Essa perspectiva permite representar e compreender melhor a interação e complexidade que permeiam diversos cenários da vida real.

Objetos e classes: conceitos

A POO é construída em torno dos conceitos de classes e objetos, e entender a função de ambos é fundamental na lógica de programação. Um ponto importante a ser esclarecido é que um objeto é criado a partir de uma classe, estabelecendo uma ponte entre o mundo real e a programação.

De forma simplificada, podemos pensar em objetos como elementos do cotidiano e nas classes como categorias que os agrupam. Por exemplo, considere a classe "Animal". Se pensarmos em animais específicos, como um cão, um gato e um pássaro, todos eles são objetos da classe "Animal". Cada um deles possui

características e comportamentos próprios, mas compartilham a categoria geral de "Animal". Portanto, a classe "Animal" é como uma gaveta que abriga diferentes tipos de animais, independentemente de suas espécies ou características individuais.

De um modo geral, uma classe e seus objetos podem ser representados da seguinte maneira:

Animal
Cachorro
Gato
Ornitorrinco

A primeira linha sempre corresponderá à classe e as demais aos objetos da classe.

Esse conceito de classes e objetos permite uma modelagem eficaz de sistemas complexos, onde categorias gerais podem ser definidas e objetos específicos podem ser criados a partir delas, com características únicas, mas ainda pertencendo à mesma classe.

Quando se trata de objetos, é importante destacar que a sua classificação não é estritamente definida, variando de acordo com o modelo em questão. Isso implica que a diversidade de objetos e classes pode ser vasta, dependendo da modelagem específica.

Em essência, um objeto pode ser imaginado como uma entidade que armazena dados ou informações relacionadas à sua estrutura, chamados atributos, e também possui um comportamento definido por meio de operações específicas.

Quanto às classes e seus atributos, é relevante observar que eles podem ser designados como privados ou públicos. Geralmente, uma classe marcada como "public" é de acesso aberto. Por outro lado, um atributo, quando definido como privado, só é visível e acessível dentro da própria classe à qual pertence, restringindo seu acesso a outros componentes do programa.

Instanciação e referência a objetos

Na POO, a interação entre a instanciação e a referência desempenha um papel crucial na busca por software mais ágil e eficiente. Nesse contexto, objetos colaboram, compartilhando seus atributos e métodos. O usuário do programa inicializa a execução, geralmente criando o primeiro objeto, a partir do qual todos os outros objetos são instanciados, direta ou indiretamente.

Segundo Barnes e Kölling (2004), é possível ilustrar a relação de classes, objetos e instâncias da seguinte maneira.

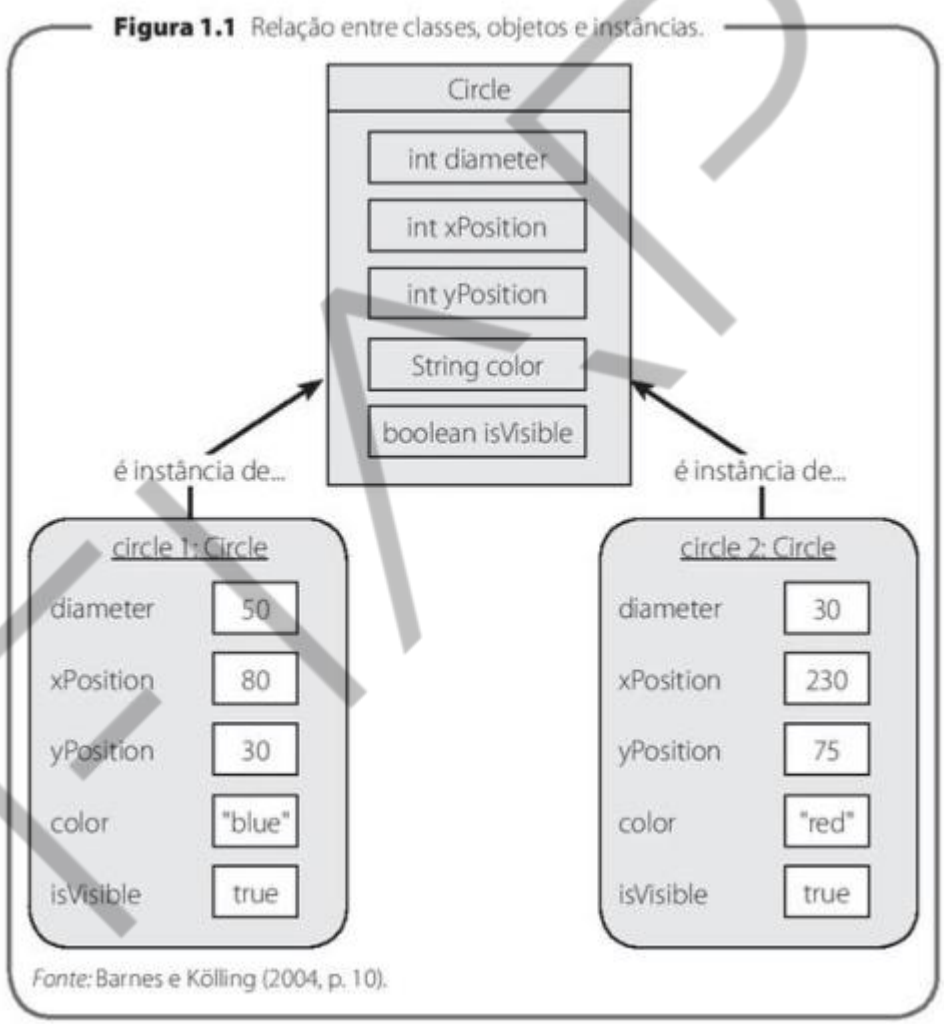

Figura 4 – Relação entre classes, objetos e instâncias
Fonte: Barnes e Kölling (2004)

Quando examinamos objetos distintos, é perceptível que objetos pertencentes à mesma classe compartilham exatamente os mesmos campos, incluindo o número, o tipo e os nomes dos campos. No entanto, o valor real de um campo específico pode

variar entre diferentes objetos da mesma classe. Em contraste, objetos de classes diferentes podem apresentar campos diferentes. Por exemplo, um objeto da classe "Círculo" possui um campo para o diâmetro, enquanto um objeto da classe "Triângulo" possui campos distintos para a largura e altura.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, exploramos os fundamentos dos paradigmas de programação, compreendendo como eles influenciam a forma como estruturamos e interagimos com o código em linguagens de programação específicas. Abordamos os principais paradigmas, como o imperativo, declarativo, funcional, lógico e orientado a eventos, destacando suas características distintas. Além disso, aprofundamos nossos conhecimentos no paradigma orientado a objetos, enfatizando a importância da modelagem por objetos, que permite representar o mundo real por meio de classes e objetos.

REFERÊNCIAS

Alura. **ORIENTAÇÃO A OBJETOS BÁSICA**. Disponível em: <https://www.alura.com.br/apostila-java-orientacao-objetos/orientacao-a-objetos-basica#motivacao-problemas-do-paradigma-procedural>. Acesso em: 29 jan. 2024.

BARNES, D; KÖLLING, M. **Programação Orientada a Objetos Com Java**. [s.l.]: Pearson, 2004.

Dev Media. **Orientação a Objetos: simples assim!** Disponível em: <https://www.devmedia.com.br/orientacao-a-objetos-simples-assim/3254>. Acesso em: 29 jan. 2024.

SAWA-Studio. **Paradigmas de programação**. Disponível em: <https://sawastudio.me/blog/index.php/2022/12/26/paradigmas-de-programacao/>. Acesso em: 29 jan. 2024.

PALAVRAS-CHAVE

Palavras-chave: Typescript. Desenvolvimento Básico. Superset.