Conhecimetos basicos

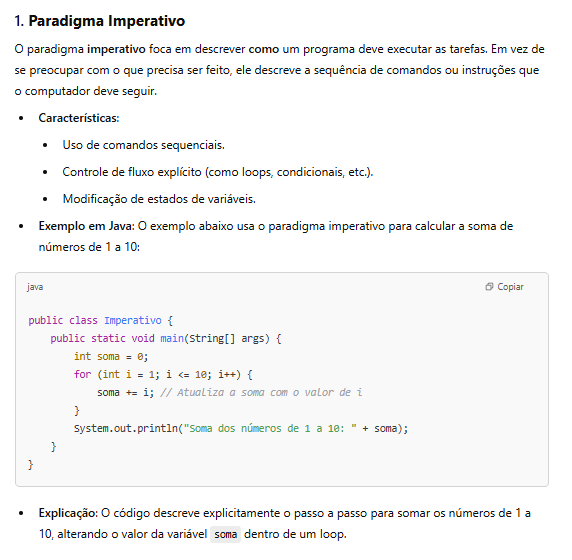
**1. Algoritmos e Estruturas de Dados**

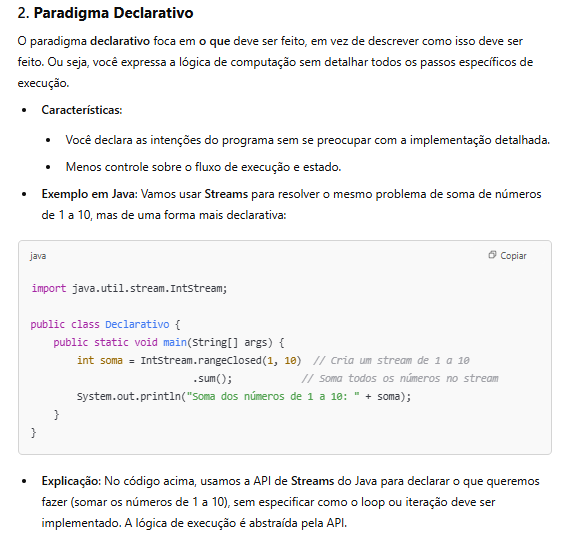
* **Algoritmos**: Como ordenar, buscar, dividir e conquistar, e entender a complexidade dos algoritmos (Big O).
* **Estruturas de dados**: Listas, pilhas, filas, árvores, grafos, tabelas hash, etc. Entender como utilizá-las para otimizar o desempenho e a memória.

**2. Complexidade Computacional**

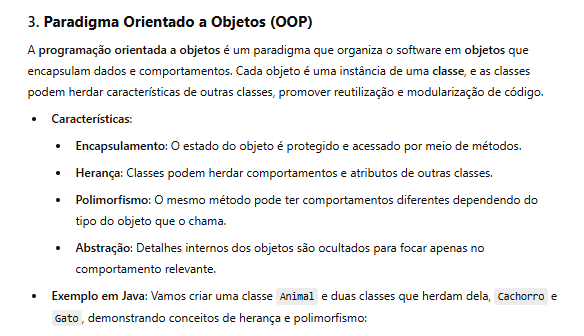
* **Análise de Complexidade**: Como calcular a eficiência de algoritmos em termos de tempo e espaço, compreendendo a notação Big O, Omega e Theta.

**3. Paradigmas de Programação**

* **Programação Imperativa vs. Declarativa**: Diferença entre ambos e quando usar cada um.
* 



* **Programação Orientada a Objetos (POO)**: Conceitos como classes, objetos, herança, polimorfismo, encapsulamento e abstração.



Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

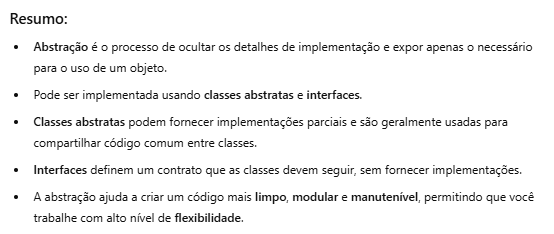
Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

Texto, Carta

Descrição gerada automaticamente



* **Programação Funcional**: Funções puras, imutabilidade, recursão, funções de ordem superior e uso de lambdas.

**4. Sistemas Operacionais**

* **Conceitos básicos de OS**: Gerenciamento de memória, processos, threads, escalonamento de CPU, sistemas de arquivos e I/O.
* **Concorrência**: Como lidar com threads, sincronização, mutexes, deadlocks e questões de paralelismo.

**5. Redes de Computadores**

* **Camadas do Modelo OSI**: Entender como as diferentes camadas de redes funcionam e se comunicam.
* **Protocolos de Comunicação**: HTTP, TCP/IP, DNS, etc.
* **Segurança**: Conceitos de criptografia, autenticação, SSL/TLS e práticas de segurança em redes.

**6. Banco de Dados**

* **Modelagem de Dados**: Normalização, tabelas, chaves primárias e estrangeiras.
* **SQL e NoSQL**: Diferenças, quando usar cada tipo, e como escrever queries eficientes.
* **Transações**: ACID, isolamento de transações e controle de concorrência.

**7. Desenvolvimento de Software e Metodologias Ágeis**

* **Metodologias Ágeis**: Scrum, Kanban, Sprints, iteração contínua.
* **Controle de Versão**: Git, GitHub/GitLab/Bitbucket, conceitos de branches, merges, pull requests.

**8. Testes de Software**

* **Testes Unitários**: Criação e importância de testes para garantir a qualidade do código.
* **Testes de Integração e Funcionais**: Garantir que o sistema como um todo funcione corretamente.
* **TDD (Test Driven Development)**: Escrever testes antes de implementar o código.

**9. Design de Software**

* **Princípios de Design**: SOLID, DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid), YAGNI (You Ain't Gonna Need It).
* **Padrões de Projeto**: Singleton, Factory, Observer, Strategy, etc.

**10. Arquitetura de Software**

* **Padrões de Arquitetura**: MVC, MVVM, microserviços, monolitos, etc.
* **Escalabilidade e Manutenibilidade**: Como criar sistemas que possam crescer e evoluir facilmente.