**apítulo 3 - Variáveis Primitivas e Controle de Fluxo**

* **Declaração e uso de variáveis**: As variáveis são declaradas com um tipo e nome, e uma vez declaradas, não podem mudar de tipo.
* **Tipos primitivos**: Existem oito tipos primitivos em Java (int, boolean, char, etc.).
* **Casting e promoção**: Conversão explícita ou implícita de tipos de dados, como transformar int em double.
* **Estruturas de controle**: if, else, switch são usados para decidir o fluxo do programa com base em condições booleanas. while, for e do-while controlam repetições.
* **Escopo de variáveis**: As variáveis são acessíveis apenas dentro do bloco em que são declaradas​(apostila-java-orientaca…)​(apostila-java-orientaca…).

**Capítulo 4 - Orientação a Objetos Básica**

* **Orientação a objetos**: Paradigma que organiza o código em classes e objetos, promovendo encapsulamento, herança e polimorfismo.
* **Classes**: Modelos que descrevem objetos. Contêm atributos (variáveis) e métodos (funções que realizam ações nos atributos).
* **Objetos**: Instâncias de uma classe que guardam dados e comportamentos.
* **Encapsulamento**: Protege dados e fornece acesso controlado a eles através de métodos (getters e setters).
* **Métodos**: Funções que definem os comportamentos de um objeto. Podem ou não ter retorno, e podem receber parâmetros​(apostila-java-orientaca…)​(apostila-java-orientaca…)​(apostila-java-orientaca…).

**Capítulo 5 - Modificadores de Acesso e Atributos de Classe**

* **Modificadores de acesso**: public, private, protected controlam o nível de visibilidade de classes, atributos e métodos:
  + public: Acesso por qualquer classe.
  + private: Acesso apenas dentro da própria classe.
  + protected: Acesso por classes no mesmo pacote ou subclasses.
* **Atributos de classe (static)**: Atributos compartilhados entre todas as instâncias de uma classe. Métodos estáticos podem ser chamados sem criar instâncias da classe.
* **Construtores**: Funções especiais usadas para inicializar objetos. Um construtor pode ser sobrecarregado para permitir diferentes formas de criação de objetos​(apostila-java-orientaca…)​(apostila-java-orientaca…).

**Capítulo 9 - Herança, Reescrita e Polimorfismo**

* **Herança**: Um mecanismo que permite que uma classe herde atributos e métodos de outra. A classe filha pode adicionar novos atributos e métodos ou sobrescrever os existentes.
* **Reescrita (Override)**: A capacidade de uma classe filha fornecer sua própria implementação de um método herdado.
* **Polimorfismo**: A habilidade de um objeto ser tratado como instância de várias classes relacionadas, geralmente via herança. Isso permite que métodos de uma superclasse sejam adaptados para o comportamento específico da subclasse​(apostila-java-orientaca…)​(apostila-java-orientaca…).

**Capítulo 10 - Classes Abstratas**

* **Classe abstrata**: Não pode ser instanciada diretamente. Serve como modelo para outras classes que a estendem.
* **Métodos abstratos**: Devem ser implementados pelas subclasses que herdam a classe abstrata.
* **Reuso de código**: Classes abstratas permitem o reuso de código comum entre várias subclasses, mas forçam a implementação de métodos específicos em cada subclasse.
* **Exemplo**: Funcionario como classe abstrata, que define um método abstrato getBonificacao(), implementado por subclasses como Gerente e Presidente​