

Apostila do curso FJ-11

Java e Orientação a Objetos

Aprenda **Java**, a linguagem de programação mais usada no mundo, e os princípios da **programação orientada a objetos**. Este material gratuito é o que usamos no [curso de Java](#) da Caelum e esperamos que seja útil no seu aprendizado. Não deixe também de [compartilhar](#) essa apostila com seus amigos.

A **Caelum** oferece [cursos de TI](#) desde 2004 em todo o Brasil. É conhecida por seus cursos nas áreas de [Java](#), [Ruby](#), [mobile](#), [front-end](#), [.NET](#) e [agile](#), além de [cursos online](#). Temos diversas [apostilas abertas](#) para download e consulta gratuita. E, se estiver interessado em nossos cursos, não deixe de [entrar em contato](#).

SUMÁRIO

1. [Como Aprender Java](#)

1. [O que é realmente importante?](#)
2. [Sobre os exercícios](#)
3. [Tirando dúvidas e indo além](#)

2. [O que é Java](#)

1. [Java](#)
2. [Uma breve história do Java](#)
3. [Máquina Virtual](#)
4. [Java lento? Hotspot e JIT](#)
5. [Versões do Java e a confusão do Java2](#)
6. [JVM? JRE? JDK? O que devo baixar?](#)
7. [Onde usar e os objetivos do Java](#)
8. [Especificação versus implementação](#)
9. [Como o FJ-11 está organizado](#)
10. [Compilando o primeiro programa](#)
11. [Executando seu primeiro programa](#)
12. [O que aconteceu?](#)

13. [Para saber mais: como é o bytecode?](#)
14. [Exercícios: Modificando o Hello World](#)
15. [O que pode dar errado?](#)
16. [Um pouco mais...](#)
17. [Exercícios adicionais](#)

3. [Variáveis primitivas e Controle de fluxo](#)

1. [Declarando e usando variáveis](#)
2. [Tipos primitivos e valores](#)
3. [Exercícios: Variáveis e tipos primitivos](#)
4. [Discussão em aula: convenções de código e código legível](#)
5. [Casting e promoção](#)
6. [O if e o else](#)
7. [O While](#)
8. [O For](#)
9. [Controlando loops](#)
10. [Escopo das variáveis](#)
11. [Um bloco dentro do outro](#)
12. [Para saber mais](#)
13. [Exercícios: Fixação de sintaxe](#)
14. [Desafios: Fibonacci](#)

4. [Orientação a objetos básica](#)

1. [Motivação: problemas do paradigma procedural](#)
2. [Criando um tipo](#)
3. [Uma classe em Java](#)
4. [Criando e usando um objeto](#)
5. [Métodos](#)
6. [Métodos com retorno](#)
7. [Objetos são acessados por referências](#)
8. [O método transfere\(\)](#)
9. [Continuando com atributos](#)
10. [Para saber mais: Uma Fábrica de Carros](#)
11. [Um pouco mais...](#)
12. [Exercícios: Orientação a Objetos](#)
13. [Desafios](#)
14. [Fixando o conhecimento](#)

5. [Um pouco de arrays](#)

1. [O problema](#)
2. [Arrays de referências](#)
3. [Percorrendo uma array](#)
4. [Percorrendo uma array no Java 5.0](#)
5. [Exercícios: Arrays](#)
6. [Um pouco mais...](#)

7. [Desafios](#)
8. [Testando o conhecimento](#)
6. [Modificadores de acesso e atributos de classe](#)
 1. [Controlando o acesso](#)
 2. [Encapsulamento](#)
 3. [Getters e Setters](#)
 4. [Construtores](#)
 5. [A necessidade de um construtor](#)
 6. [Atributos de classe](#)
 7. [Um pouco mais...](#)
 8. [Exercícios: Encapsulamento, construtores e static](#)
 9. [Desafios](#)
7. [Herança, reescrita e polimorfismo](#)
 1. [Repetindo código?](#)
 2. [Reescrita de método](#)
 3. [Invocando o método reescrito](#)
 4. [Polimorfismo](#)
 5. [Um outro exemplo](#)
 6. [Um pouco mais...](#)
 7. [Exercícios: Herança e Polimorfismo](#)
 8. [Discussões em aula: Alternativas ao atributo protected](#)
8. [Eclipse IDE](#)
 1. [O Eclipse](#)
 2. [Apresentando o Eclipse](#)
 3. [Views e Perspective](#)
 4. [Criando um projeto novo](#)
 5. [Criando o main](#)
 6. [Executando o main](#)
 7. [Pequenos truques](#)
 8. [Exercícios: Eclipse](#)
 9. [Discussão em aula: Refactoring](#)
9. [Classes Abstratas](#)
 1. [Repetindo mais código?](#)
 2. [Classe abstrata](#)
 3. [Métodos abstratos](#)
 4. [Aumentando o exemplo](#)
 5. [Para saber mais...](#)
 6. [Exercícios: Classes Abstratas](#)
 7. [Desafios](#)
10. [Interfaces](#)
 1. [Aumentando nosso exemplo](#)

2. [Interfaces](#)
 3. [Dificuldade no aprendizado de interfaces](#)
 4. [Exemplo interessante: conexões com o banco de dados](#)
 5. [Exercícios: Interfaces](#)
 6. [Exercícios avançados opcionais](#)
 7. [Discussão: favoreça composição em relação à herança](#)
-
11. [Exceções e controle de erros](#)
 1. [Motivação](#)
 2. [Exercício para começar com os conceitos](#)
 3. [Exceções de Runtime mais comuns](#)
 4. [Outro tipo de exceção: Checked Exceptions](#)
 5. [Um pouco da grande família Throwable](#)
 6. [Mais de um erro](#)
 7. [Lançando exceções](#)
 8. [O que colocar dentro do try?](#)
 9. [Criando seu próprio tipo de exceção](#)
 10. [Para saber mais: finally](#)
 11. [Exercícios: Exceções](#)
 12. [Desafios](#)
 13. [Discussão em aula: catch e throws em Exception](#)
-
12. [Pacotes - Organizando suas classes e bibliotecas](#)
 1. [Organização](#)
 2. [Diretórios](#)
 3. [Import](#)
 4. [Acesso aos atributos, construtores e métodos](#)
 5. [Usando o Eclipse com pacotes](#)
 6. [Exercícios: Pacotes](#)
-
13. [Ferramentas: jar e javadoc](#)
 1. [Arquivos, bibliotecas e versões](#)
 2. [Gerando o JAR pelo Eclipse](#)
 3. [Javadoc](#)
 4. [Gerando o Javadoc](#)
 5. [Exercícios: Jar e Javadoc](#)
-
14. [O pacote java.lang](#)
 1. [Pacote java.lang](#)
 2. [Um pouco sobre a classe System](#)
 3. [java.lang.Object](#)
 4. [Casting de referências](#)
 5. [Métodos do java.lang.Object: equals e toString](#)
 6. [Integer e classes wrappers \(box\)](#)
 7. [Autoboxing no Java 5.0](#)

8. [java.lang.String](#)
9. [java.lang.Math](#)
10. [Exercícios: java.lang](#)
11. [Desafio](#)
12. [Discussão em aula: O que você precisa fazer em Java?](#)
15. [Pacote java.io](#)
 1. [Conhecendo uma API](#)
 2. [Orientação a objetos no java.io](#)
 3. [InputStream, InputStreamReader e BufferedReader](#)
 4. [Lendo Strings do teclado](#)
 5. [A analogia para a escrita: OutputStream](#)
 6. [Uma maneira mais fácil: Scanner e PrintStream](#)
 7. [Um pouco mais...](#)
 8. [Exercícios: Java I/O](#)
 9. [Discussão em aula: Design Patterns e o Template Method](#)
16. [Collections framework](#)
 1. [Arrays são trabalhosos, utilizar estrutura de dados](#)
 2. [Listas: java.util.List](#)
 3. [Listas no Java 5 e Java 7 com Generics](#)
 4. [A importância das interfaces nas coleções](#)
 5. [Ordenação: Collections.sort](#)
 6. [Exercícios: Ordenação](#)
 7. [Conjunto: java.util.Set](#)
 8. [Principais interfaces: java.util.Collection](#)
 9. [Percorrendo coleções no Java 5](#)
 10. [Para saber mais: Iterando sobre coleções com java.util.Iterator](#)
 11. [Mapas - java.util.Map](#)
 12. [Para saber mais: Properties](#)
 13. [Para saber mais: Equals e hashCode](#)
 14. [Para saber mais: Boas práticas](#)
 15. [Exercícios: Collections](#)
 16. [Desafios](#)
 17. [Para saber mais: Comparators, classes anônimas, Java 8 e o lambda](#)
17. [Programação Concorrente e Threads](#)
 1. [Threads](#)
 2. [Escalonador e trocas de contexto](#)
 3. [Garbage Collector](#)
 4. [Exercícios](#)
 5. [E as classes anônimas?](#)
18. [E agora?](#)
 1. [Web](#)

2. [Praticando Java e usando bibliotecas](#)
3. [Grupos de Usuários](#)
4. [Próximos cursos](#)

19. [Apêndice – Sockets](#)

1. [Motivação: uma API que usa os conceitos aprendidos](#)
2. [Protocolo](#)
3. [Porta](#)
4. [Socket](#)
5. [Servidor](#)
6. [Cliente](#)
7. [Imagem geral](#)
8. [Exercícios: Sockets](#)
9. [Desafio: Múltiplos Clientes](#)
10. [Desafio: broadcast das mensagens](#)
11. [Solução do sistema de chat](#)

20. [Apêndice – Problemas com concorrência](#)

1. [Threads acessando dados compartilhados](#)
2. [Controlando o acesso concorrente](#)
3. [Vector e Hashtable](#)
4. [Um pouco mais...](#)
5. [Exercícios avançados de programação concorrente e locks](#)

21. [Apêndice – Instalação do Java](#)

1. [Instalando no Ubuntu e em outros Linux](#)
2. [No Mac OS X](#)
3. [Instalação do JDK em ambiente Windows](#)

22. [Apêndice – Debugging](#)

1. [O que é debugar](#)
2. [Debugando no Eclipse](#)
3. [Perspectiva de debug](#)
4. [Debug avançado](#)
5. [Profiling](#)
6. [Profiling no Eclipse TPTP](#)

Você encontra a Caelum também em: