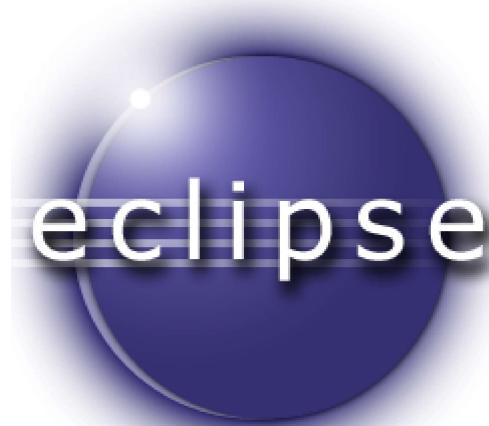


Ambiente de Desenvolvimento e Sintaxe

Apresentação



Fonte: <https://goo.gl/kpt6k7>

Nesta aula, você conhecerá o ambiente de desenvolvimento Eclipse, que é uma Interface de Desenvolvimento que potencializa o desenvolvimento e ainda organiza seus projetos.

Além disso, você conhecerá o restante das estruturas de programação para que você consiga resolver problemas computacionais. Mais especificamente, serão vistos: os tipos de dados, as estruturas de controle de fluxo, desde condicionais até as repetições. Com isso, você ganha musculatura para resolver problemas cada vez mais complexos na linguagem.

É importante lembrar que esta aula não tem como propósito abordar a lógica de programação, mas apenas lhe apresentar a sintaxe de recursos que você já conhece. Foi papel das disciplinas anteriores fazer isso por você.

Para finalizar, estará disponível uma lista de exercícios, para que você coloque em prática tudo que foi visto nesta aula. Esta aula é muito importante para lhe dar base para os desafios que estão por vir. Seja aplicado, siga as

orientações apresentadas ao longo da aula, faça todos os exercícios e assista aos vídeos.

Espere! Você fez os exercícios da aula passada? Caso não, corra lá e faça, antes de começar esta aula. Pois é importante para o seu processo de aprendizagem.

Conteúdo

Ambiente de Desenvolvimento

Você já pode perceber que é desestimulante criar programas em editores de texto simples. Primeiro, porque é fácil você inserir erros em seus códigos, as palavras reservadas não se destacam, trocar uma letra ou outra é muito fácil, esquecer uma chave ou colchete, etc. Segundo, porque a morosidade de ficar criando arquivos e executando por linhas de comando retarda a execução dos aplicativos, concorda?

O processo de construção deve ser acelerado e com qualidade. Para isso, um dos recursos é uma boa IDE de desenvolvimento. IDE, do inglês *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, é um [WIKIPÉDIA](#) (2017a). É programa para o programador programar programas, entendeu?

Um bom IDE deve trazer facilidades na edição, compilação, linkedição, na busca de erros, no deployment, ou seja, apoiar nas diversas fases do desenvolvimento de software.

Atualmente, no mercado, existem diversos IDEs de desenvolvimento: Eclipse, Netbeans, Jbuilder, etc. As empresas criam seus ambientes de desenvolvimento de acordo com a necessidade e estrutura organizacional. Para este curso, será utilizado a **IDE Eclipse**, que é um software livre e amplamente utilizado pelo mundo. Foi inicialmente construído pela IBM e é baseado em Java. Falando em IDE, você sabia que o **Calango** foi desenvolvido em Java? Sim, o Calango foi feito em Java por dois estudantes da Universidade Católica de Brasília, motivo de orgulho.

Chegou a hora de baixar o Eclipse, e você pode (deve) fazer o *download* do *Eclipse for JEE Developers*. Observe que, na primeira aula, foi informado que seu curso está situado na parte do JSE da tecnologia Java, e agora você está sendo solicitado a baixar um IDE destinado a desenvolvedores JEE. Então... Isso só confirma a teoria de que o JSE é base para todas as outras partes (JEE e JME), afinal, quem pode mais, pode menos. Com esse IDE, você não precisará reinstalar quando estiver trabalhando com aplicações na Internet, e com certeza seus estudos não pararão por aqui, né? Ele já atenderá a sua necessidade, tanto agora, quanto no futuro.

O Eclipse tem vários empacotamentos, cada um deles tem nomes peculiares, como: Mars, Juno, Neon, Kepler são exemplos, dentre vários. O ideal é que você utilize o **Neon**, para que tenha o mesmo IDE no qual os vídeos de apoio foram gravados.

Para fazer o *download* do **Eclipse Neon**, você deverá apontar qual é o sistema operacional e a arquitetura (32 bits ou 64 bits) da sua aplicação.

Importante

- [Baixe o Eclipse Neon](#) ↗

Para instalar o Eclipse, o procedimento é muito fácil, basta você descompactar que está pronto para uso, mas se lembre-se de que o JDK deve estar instalado corretamente.

Organização por Projetos no Eclipse

Ao abrir o Eclipse, ele irá questioná-lo onde você deseja salvar seus projetos. Você definirá a área de trabalho (*workspace*). Para cada programa que você fizer, é indicado que crie um projeto novo. Padronizar os nomes dos projetos ajudará na organização.

Para os exercícios, adote o padrão de nome de projetos: AulaXExercicioY.

Acompanhe o **vídeo 1** para visualizar um exemplo de organização e visão inicial do eclipse.

Vídeo 1

Jeysel POO unid 1 aula 2 pt 1



Executando e Importando Projetos

A IDE de desenvolvimento, no papel de facilitadora de algumas atividades rotineiras no processo de desenvolvimento, não poderia deixar de lado a execução dos programas.

Outros projetos podem ter sido feitos por outro desenvolvedor e devem ser importados para o Eclipse. A ferramenta possibilita esse tipo de interação.

Acompanhe o vídeo 2 para você conferir como é esse processo.

Java é uma linguagem fortemente tipada. Você só aloca um espaço de memória se determinar o tipo antes. O mesmo acontece com o C. Na tabela 1, são apresentados os tipos de dados do Java importantes para este curso.

Tabela 1 – Listagem dos tipos de dados.

Tipo	Grupo	Origem	Tamanho	Exemplo de uso
<i>byte</i>	Inteiro	primitivo	8 bits	<code>byte b = 127;</code>
<i>short</i>	Inteiro	primitivo	16 bits	<code>short s = 10;</code>
<i>int</i>	Inteiro	primitivo	32 bits	<code>int i = 8907;</code>
<i>long</i>	Inteiro	primitivo	64 bits	<code>long l = 34567L;</code>
<i>float</i>	Flutuante	primitivo	32 bits	<code>float f = 34.0F;</code>
<i>double</i>	Flutuante	primitivo	64 bits	<code>double d = 34.0;</code>
<i>char</i>	Caractere	primitivo	16 bits	<code>char c = 'a';</code>
<i>String</i>	Texto	objeto	Não se aplica	<code>String texto = "abc";</code>
<i>boolean</i>	Lógico	primitivo	depende da JVM	<code>boolean b = false;</code>
<i>[]</i>	Array	objeto	Não se aplica	<code>int []nums = new int [5];</code>

Vários dos tipos listados na tabela anterior você já conhece. A última coluna é o exemplo de uso, que serve para você observar, na prática, a aplicação do tipo de dado para facilitar a compreensão, com a declaração da variável e atribuição do seu valor literal.

Os tipos **byte** e **short** podem ser novidades para você; eles são numéricos e utilizados para armazenar valores de pequeno porte.

O tipo **int** dispensa apresentações. É o tipo inteiro padrão da linguagem Java. *Toda vez que for declarar uma variável do tipo inteiro, faça isso com int.* "E se for a idade de uma pessoa? A pessoa mais velha do mundo viveu 122 anos, então posso usar um byte?" A resposta é NÃO. O tipo padrão numérico inteiro é o int, você não está em um ambiente crítico de desenvolvimento com restrição de memória.

O tipo de dado **long** é inteiro, mas tem capacidade dobrada frente ao **int**.

O tipo **float** vai deixar saudade. O tipo flutuante padrão do Java é o **double**.

Programadores C insistem ao declarar variáveis flutuantes como *float* em Java.

Perceba que, na última coluna, para atribuir um valor do tipo float literal, é necessário realizar um *casting*  para realizar a atribuição. Por isso, é justificada a presença do F/final da atribuição. Float não é o padrão.

O tipo **double**, como já foi dito, é o tipo flutuante padrão do Java. Sempre que for declarar uma variável que armazenará valores flutuantes, faça isso com double.

O tipo **char** é diferente do C por conta do tamanho. Na linguagem C, este tipo de dado possui 8 bits, mas no Java possui 16 bits. Isso se justifica porque o Java é capaz de representar caracteres no formato UTF-16. Lembra dos problemas de acentuação no C? No Java, você não terá isso.

O tipo **String** você já conhece. Lembra que você tinha o tipo "Texto" no Calango na disciplina de algoritmos? Pois então, para armazenar uma cadeia de caracteres faça uso deste OBJETO. Mais detalhes sobre as Strings serão vistos na Aula 4.

Outra novidade para você pode ser o tipo **boolean**. Esse tipo de dado serve para armazenar valores lógicos, resultantes de comparações, por exemplo. Sempre que quiser representar um valor booleano, ou seja, se algo é falso ou verdadeiro, faça isso com boolean.

O tipo **[]** (array) tem a mesma finalidade da que você já conhece do C, manipular um conjunto de variáveis do mesmo tipo por meio de índices. As ações de percorrer e de atribuir acontecem da mesma forma que você fazia em C.

Os tipos de dados que estão destacados na tabela são os tipos que você fará uso neste curso.

Para facilitar para você, a Tabela 2 traz exemplos de leituras de dados para os tipos destacados, exceto o *array*, utilizando o objeto **Scanner**, conforme apresentado na aula anterior.

Tabela 2 – Exemplos de entrada de dados para os tipos principais.

Tipo	Exemplo

Vídeo 2

Jeysel POO unid 1 aula 2 pt 2



A ferramenta Eclipse fornece vários tipos de atalhos de comandos para facilitar a vida do desenvolvedor nas diversas etapas de produção. Neste momento do aprendizado, é interessante que você não faça uso de alguns recursos, pois poderiam retardar o processo de aprendizagem pelo qual você está passando.

Mais adiante, na disciplina, será disponibilizado um novo material, para que melhore ainda mais sua destreza na ferramenta. Se descobrir sozinho, ótimo. Mas lembre-se de que não é porque você sabe tudo de Eclipse que você é um bom programador orientado a objetos em uma determinada linguagem. Uma pessoa especializada em *Word* pode não escrever bons textos. Não perca o foco!

Sintaxe

Continuando a evoluir suas habilidades, ainda mais agora que você conheceu o Eclipse, serão apresentadas a você as estruturas que você já conhece, mas sobre as quais ainda pode ter algumas dúvidas. Você já sabe, só não sabe disso.

O conteúdo passará pelo que você já conhece e focar no que você ainda não conhece. Muitas das estruturas a seguir são exatamente iguais ao C; para as que forem diferentes, será dada maior ênfase. E se prepare, pois vários exercícios serão explorados com a finalidade de te deixar bem ambientado com a tecnologia.

Tipos de Dados

Tipo	Exemplo
int	int numInt = new Scanner(System.in).nextInt();
double	double numFl = new Scanner(System.in).nextDouble();
char	char varChar = new Scanner(System.in).next().charAt(0);
String	String varStr = new Scanner(System.in).nextLine();
boolean	boolean varBoo = new Scanner(System.in).nextBoolean();

Pronto, agora você foi apresentado aos tipos de dados do Java e à forma de leitura de cada um deles.

Agora, dê uma pausa na leitura, vá até o Caderno de Exercícios e **faça o exercício 1 desta aula.**

Já que você fez o exercício, deve ter percebido que o Eclipse colocou alertas em cada linha que você realizou uma chamada de leitura de dados. Veja o quanto é importante você seguir as orientações que estão nas aulas. Não fez o exercício? Então, dê uma pausa e vá até lá. Espero você!

A utilização do objeto Scanner é uma forma de aproximar você do que já fazia no C. Ler dados do console é abrir um canal de comunicação, e o Eclipse te avisa o não fechamento do Scanner. Isso não é problema para nós. É a forma mais simples de aproximar você do passado. Se você tem TOC (Transtorno Obsessivo Compulsivo), assim como alguns programadores, ao ver esses avisos na lateral do seu IDE, coloque a instrução acima da declaração da classe para retirar os avisos do eclipse, conforme o exemplo apresentado na Tabela 3.

Tabela 3 – Trecho de código para retirar os alertas do eclipse.

```
@SuppressWarnings("all") //coloque somente na classe que fizer leitura de
dados, ok?
public class Aula2Exercicio1 {}
```

Para Refletir

Como seria fazer a entrada de dados para popular um *array* de inteiro de 5 posições em java?

Na última linha da Tabela 1, há um exemplo de criação do *array* de cinco posições. Mas, para popular, você deve acessar uma determinada posição. Lembra como fazia no C (na disciplina anterior)? É exatamente da mesma forma. Na Tabela 2 tem um exemplo de leitura de um dado inteiro. Com essas informações, consegue encontrar a resposta para a questão do "Para Refletir" anterior? Se não conseguir, conclua o estudo da aula e volte nessa questão.

Operadores

Os operadores, em qualquer linguagem de programação, servem para a resolução de problemas computacionais. Durante sua jornada no curso, você percebeu que seus programas devem tomar algumas decisões e manipular dados. A seguir, a Tabela 4 apresenta os operadores que provavelmente serão utilizados por você.

Tabela 4 – Lista de operadores com exemplo de uso.

Tipo do Operador	Lista de Operadores	Exemplo de Uso
Sufixais	var++ , var--	<pre>int var = 0; var++; System.out.print(var++);</pre>
Prefixais	-num , ++num	<pre>double num = 10; System.out.print(--num);</pre>
Aritméticos	*, + , - , / , %	<pre>double num = 10.0; System.out.print(num % 2);</pre>

Tipo do Operador	Lista de Operadores	Exemplo de Uso
Concatenação	+	<pre>double num = 10.5; System.out.print("Valor é: "+num); String a = "A", b = "B"; String texto = a + b; System.out.print(texto);</pre>
Igualdade	== , !=	<pre>int num1 = 10, num2 = 20; if(num1 == num2) System.out.println("Iguais"); if(num1 != num2) System.out.println("Diferentes");</pre>
Comparativos	>=, <=, > , <	<pre>int num1 = 10, num2 = 10; if(num1 >= num2) System.out.println("Verdadeiro"); else System.out.println("Falso");</pre>
Lógicos	&& ,	<pre>int num1 = 10, num2 = 11; if(num1 == 10 && num2 != 10) System.out.println("Verdadeiro"); else System.out.println("Falso");</pre>
Atribuição	= , += , -= , *=, /= %=	<pre>double num = 10.0; System.out.print(num += 20); System.out.print(num /= 2);</pre>

Ao ler os exemplos de uso na última coluna da tabela anterior, sua curiosidade deve ter levado você a testá-los mais de perto, sim? Por exemplo: qual a saída do código localizado na primeira linha do operador sufixal da tabela acima? Você acha que é 2? Pois não é. Faça você mesmo e descubra. A sua aprendizagem depende de seu empenho e de sua curiosidade.

Controle de Fluxo

Quando um programa é executado, ele segue uma sequência de execução; entretanto, muitas vezes, se faz necessário mudar esse fluxo devido a algum valor diferente, testar um valor para efetuar operações diferentes, ou mesmo repetir várias vezes um trecho de código. Você já deve ter percebido isso, pois isso é programar.

Logo, as linguagens de programação oferecem as estruturas de controle. A seguir, na Tabela 5, serão apresentadas as estruturas de controle do Java. Você observará que há muita similaridade com a linguagem C. Ao final da tabela, serão apresentadas as peculiaridades, que são poucas.

Tabela 5 – Lista de controles de fluxo com exemplos.

Controle	Tipo	Exemplo
if / else	seleção	<pre>int num1 = 10, num2 = 20; if(num1 == num2) System.out.println("Iguais"); if(num1 != num2) System.out.println("Diferentes");</pre>
switch	escolha	<pre>char situacao = 'e'; switch(situacao){ case 'c': System.out.print("Casado");break; case 'd': System.out.print("Divorciado");break; case 's': System.out.print("Solteiro");break; case 'e': System.out.print("Enrolado");break; case 'f': System.out.print("Ficando");break; }</pre>
for	repetição	<pre>for (int i = 0; i < 10; i++) { System.out.println(i); }</pre>
while	repetição	<pre>int num = 0; while (num < 10){ System.out.print(++num); } // mude o valor de num para 10 e teste</pre>

Controle	Tipo	Exemplo
do/while	repetição	<pre>int num = 0; do { System.out.print(++num); } while(num < 10); // mude o valor de num para 10 e teste</pre>

Conforme você pode observar nos exemplos de código, tudo é muito semelhante ao C. Na verdade, não há diferença. Mas duas coisas importantes devem ser destacadas:

O **switch**, nas versões mais recentes do Java, aceita uma String como valor de decisão. Faça o teste você mesmo! Mude o exemplo do **switch** para String. Lembre-se de mudar os valores literais dos casos para String também.

Na estrutura de repetição **for**, foi declarada uma variável 'i', que controla a estrutura de repetição. Esta variável terá o escopo apenas dentro deste **for**. Faça você mesmo! Tente imprimir o valor de 'i' fora da estrutura do **for** no Eclipse.

Estrutura de Dados Homogênea

Imagine um programa que deve receber a idade de 5.000 pessoas, e na sequência apresentar todos os valores informados, a média das idades e quantas são maiores que 18 anos. Perceba que, se você não armazenar os valores, não será possível fazer a apresentação depois. É claro que temos que definir variáveis inteiras para armazenamento. Partiu... declarar 5000 mil variáveis? Não faça isso! Declarar um *array* (vetor) é a melhor opção. A Tabela 6 apresenta a inicialização, manipulação e iteração (como percorrer) um *array*. A última coluna apresenta comentários importantes sobre cada ação em um *array*.

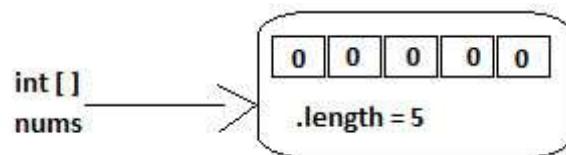
Tabela 6 – Lista de ações sobre os arrays com a sintaxe.

Ações	Sintaxe	Comentário
Inicialização	<pre>int [] nums = new int [5]; //ou int [] numsDois = {0,0,0,0,0};</pre>	Esta instrução cria um <i>array</i> com 5 posições, que varia de 0 a 4. Todos os valores são inicializados com 0.

Ações	Sintaxe	Comentário
Manipulação	nums[2] = 0; if(nums[0] == 0){...} System.out.print(nums[0]);	Para acessar uma determinada posição de um <i>array</i> , deve ser utilizado um valor inteiro como índice.
Iteração	for(int i=0;i<nums.length; i++) System.out.print(values[i]);	Percorrer um <i>array</i> em Java é facilitado com o atributo <i>length</i> do objeto <i>array</i> , ele retorna a quantidade de elementos que existe no <i>array</i> declarado.

As três ações descritas na tabela anterior, você já realizou utilizando C. A novidade para você foi o atributo **Length**. Tipos de dados **[]** (*array*) no Java são objetos, e você já deve começar a saber que objetos têm atributos, um atributo presente em todos os objetos do tipo **[]** (*array*) é o **Length**; que retorna o mesmo valor da declaração da inicialização do *array*. Observe a Figura 2.1, que representa a instanciação de um *array* em memória para o código de exemplo citado na Tabela 6.

Figura 1 – Representação de instanciação de um array em memória.



Para Refletir ⚡

Para passar um *array* como parâmetro para uma função em C, você precisa passar o tamanho do *array*. No Java, com o atributo *length* dos objetos *array* do Java, essa necessidade se mantém?

Para passar um *array* como argumento para uma função escrita em C, é necessário passar um argumento a mais, que é o tamanho do *array*. Esse parâmetro serve para a função saber quando parar durante a iteração (ato de percorrer) do *array*. Em Java, com o atributo *.length*, não se torna mais necessária a passagem de argumento extra para controle do *array*. O próprio objeto *array* tem essa informação.

Convenção Java

Ao codificar suas aplicações, é comum que tenha sua própria forma de definir suas estruturas, declarar variáveis etc. Uns declaram variáveis com letra maiúscula e usam *underline* para separação de todas as variáveis. Cada linguagem traz uma identidade na codificação que toda a comunidade de programadores daquela linguagem utiliza. Com o Java não é diferente. Todavia, existe uma convenção  que foi definida para a linguagem e publicada em 1997.

Java adota o padrão *CamelCase* para identificação de suas estruturas. Ao definir algo com mais de uma palavra, na mudança de palavra, deve mudar a caixa para alta para diferenciar, por exemplo, um método que calcule a média: `calcularMedia()`.

Perceba que, na mudança de palavra, a caixa ficou alta para facilitar a leitura.

A declaração de variáveis e métodos segue a mesma regra: **inicia-se com caixa baixa e na mudança de palavra fica caixa alta. Na definição de classes, deve começar SEMPRE com letra maiúscula.**

Na declaração de constantes, você deve fazer tudo em caixa alta e separar por *underline*, por exemplo: **MEDIA_APROVACAO**.

Cabe destaque que não utilizar os padrões definidos pela convenção não gera erros em termos de compilação, porém, outros programadores Java perceberão que você não programa dentro dos padrões da linguagem e isso é um ponto muito negativo. Imagine: "Se ele não usa a nomenclatura para desenvolver, imagine o paradigma orientado a objetos?". Saiba que você pode até mesmo ser eliminado de uma seleção por conta disso. Então, utilizar a nomenclatura Java acaba se tornando "optatório" (optativo, mas obrigatório). Você pode até não usar, mas não é uma boa ideia. Procure seguir os padrões dos vídeos, que sua adaptação virá naturalmente.

Na Prática

"Prezado(a) estudante,

Esta seção é composta por atividades que objetivam consolidar a sua aprendizagem quanto aos conteúdos estudados e discutidos. **Caso alguma dessas atividades seja avaliativa, seu (sua) professor (a) indicará no Plano de Ensino e lhe orientará quanto aos critérios e formas de apresentação e de envio.**"

Bom Trabalho!

Atividade 01

^

Crie um projeto Java no ambiente eclipse (sugestão de nome de projeto - Aula2Exercicio1).

Na sequência, crie uma classe com o nome **Programa**. Nesta classe, você deve fazer a entrada de dados das seguintes informações: o salário de um professor, a idade de uma criança, se um servidor público tem plano de saúde ou não, se uma pessoa é casada (c) ou solteira (s) e, por fim, o nome de um aluno.

Ao terminar as entradas de dados, você deve fazer a apresentação dos valores utilizando `System.out.println()`. Além disso, coloque textos com valor significativo para o usuário, por exemplo: "O nome informado foi: João Silveira Neto".

Atividade 02

^

Implemente um programa em Java que calcule o juro de uma dívida que você contraiu no mês passado no crediário de uma loja. A taxa de juros mensal e o valor da dívida serão fornecidos pelo usuário.

Atividade 03



Implemente um programa em Java que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês. Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, o programa deve mostrar como resultado o seu nome e o seu salário no final do mês.

Atividade 04



Escreva um programa em Java que leia a idade e a altura de 10 pessoas. Calcule e informe a média das alturas das pessoas com mais de 50 anos. Para isso, use **for**.

Atividade 05



Escreva um programa Java que leia 500 valores inteiros e positivos e:

- Encontre o maior valor.
- Encontre o menor valor.
- Calcule a média dos números lidos.

Faça as apresentações para o usuário de forma organizada.

Atividade 06



Faça um programa Java que receba duas notas de um estudante em valores double. Na sequência, apresente todos os valores informados, a média, e se ele está aprovado ou reprovado. Detalhe: você não deve permitir que os valores informados como notas sejam inferiores a 0 ou superiores a 10. Para validação, use **do/while**.

Atividade 07



Faça um programa Java que receba o nome de um estudante e duas notas. Na sequência, apresente todos os valores informados, a média, e se ele está aprovado ou reprovado. Detalhe: você não deve permitir que os valores informados como notas sejam inferiores a 0 ou superiores a 10. O usuário deve informar se quer continuar a usar o programa informando S-Sim ou N-Não.

Atividade 08



Faça um programa que receba 5 mil dados do usuário do tipo inteiro. **Sabe-se que valores negativos não são aceitos.** Após receber esses valores e popular o array, imprima na saída padrão a média dos valores, quantos valores são ímpares e todos os valores que foram informados.

Saiba Mais

Para ampliar seu conhecimento a respeito desse assunto, veja abaixo a(s) sugestão(ões) do professor:

- Para saber um pouco mais sobre a plataforma IDE Eclipse, acesse o [site da IBM](#) ↗.
- Para saber um pouco mais sobre como retirar os alertas do Eclipse, acesse o [site da IBM](#) ↗.
- Leia o documento [Java Code Conventions](#) ↗ para conhecer mais detalhes a respeito da convenção Java.

Referências

- BOOCH, G. **Object-Oriented Design with Applications**, Benjamin-Cummings, 1991.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML – guia do usuário**. 2. ed. Rio de Janeiro: Campus. 2006.
- CARDELLI, L.; WEGNER, P. **On Understanding Types, Data Abstraction, and Polymorphism**. ACM Computing Surveys (CSUR). vol. 17, pp. 471-523. 1985.
- DEITEL H. M.; DEITEL, P. J. **Java, como programar**. 6. ed. Porto Alegre: Bookman. 2006.
- DICIONÁRIO AURELIO. 2017. Disponível em:
<https://contas.tcu.gov.br/dicionario/home.asp> Acesso em: 19 mar. 2017.
- ERICH, G. et al. **Padrões de projeto**: soluções reutilizáveis de software rientado a objetos. Porto Alegre: Bookman. 2000.
- HORSTMANN, C. S.; CORNELL, G. **Corejava 2 – Volume I – Fundamentals**. São Paulo: Makron Books. 2010.
- NEWRELIC. **The Most Popular Programming Languages of 2016**. 2016. Disponível em:
<https://blog.newrelic.com/2016/08/18/popular-programming-languages-2016-go>. Acesso em: 9 Mar 2017.
- NIEMEYER, P.; KNUDSEN, J. **Aprendendo Java**. Rio de Janeiro: Campus. 2000.
- ORACLE. **Java Licensing Logo**. 2017. Disponível em:
<http://www.oracle.com/us/technologies/java/java-licensing-logo-guidelines-1908204.pdf>. Acesso em: 13 mar. 2017.
- SIERRA, K.; BATES, B. **Certificação Sun para Programador JAVA 5 Guia de Estudo**. Rio de Janeiro: Alta Books, 2006.
- SILBERSCHATZ, A; GALVIN, P. B.; GAGNE, G. **Sistemas operacionais com Java**. 6. ed. Rio de Janeiro. Editora Campus, 2004.

- STUCKEY, P. J.; SULZMANN, M. **A theory of overloading**. International Conference on Functional Programming. Proceedings of the seventh ACM SIGPLAN international conference on Functional programming. Pittsburgh, PA, USA, 2002. pp. 167-178.
- WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Conteúdo sobre o **Ambiente de Desenvolvimento Integrado**. 2017. Disponível em: <https://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integrado>. Acesso em: 5 mar. 2017.

Finalizando... C

Nesta aula, você conheceu a IDE Eclipse e já pôde reconhecer alguns benefícios da sua utilização. Você foi apresentado a todas as estruturas que já conhecia de outros momentos de estudos em disciplinas anteriores que já cursou.

*Esta aula serviu para lhe dar maior confiança com o ambiente e com a sintaxe do Java. Veja se você já seguiu todas as orientações apresentadas ao longo da aula e, se realizar os exercícios desta aula, propostos na Prática, você terá grandes chances de compreender na prática os conceitos importantes de orientação a objetos que serão apresentados. **Muitos estudantes sofrem com a orientação a objetos por ignorarem algumas etapas do aprendizado.***

*Agora você precisa se movimentar. Faça **todos** os exercícios desta aula. Somente depois de realizá-los, assista ao **vídeo 3**.*

Vídeo 3

Jeysel POO unid 1 aula 2 pt 3



*Para acompanhar um exemplo prático de array, assista ao **vídeo 4**.*

Vídeo 4

Jeysel POO unid 1 aula 2 pt4



Mas, atenção! Não assista ao vídeo sem realizar os exercícios, pois ele será esclarecedor somente se você fizer os exercícios antes. Bons estudos! Agora... se ainda não realizou os exercícios, feche a janela com o vídeo e realize-os. Em seguida, sim, retorno para assistir ao vídeo com o exemplo.