# Unidade 2

Nesta unidade o aluno iniciará o seu aprendizado com a prática na programação, sendo assim a parte conceitual será menos aprofundada. Na disposição da sala, após identificado os alunos com maior e menor conhecimento, mescle-os, motivando-os a trabalhar em equipe aumentando a integração e agregando conhecimento.

## Aula 1- Conceitos básicos de Java

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para:

* Realizar a instalação do ambiente de desenvolvimento;
* Realizar a instalação da IDE Netbeans;
* Desenvolver o programa Hello World.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 30 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 50 minutos para desenvolver as atividades propostas para a turma.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Introdução;
* Começando a trabalhar com o Java;
* Primeiros passos: como escrever um programa em Java;
* Primeiro programa: Hello World;
* Classes, objetos e instâncias;
* Programa Hello World (Explicação).

### PONTOS IMPORTANTES

Para garantir ao aluno um aprendizado significativo, resgate os principais conceitos de cada tópico abordado na aula. Neste momento, é importante que você utilize dicas e sugestões para reforçar os temas trabalhados, oferecendo também exemplos que possibilitem a relação entre teoria e prática.

### – Introdução

#### Por que Java?

Educador, neste tópico fale um pouco sobre o surgimento da linguagem de programação orientada a objetos, Java. Comente sobre as vantagens a reusabilidade, o foco na representação dos dados e a criação de classes, objetos e instâncias que se comunicam por meio de funções. Neste primeiro contato você identificará os alunos que possuem conhecimento sobre Java, possibilitando uma melhor disposição na sala de aula.

### – Começando a trabalhar com o Java

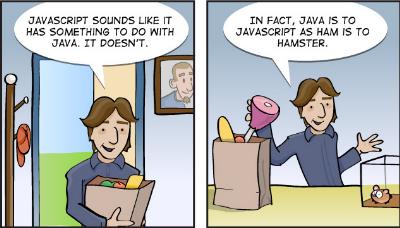
Neste tópico inicia-se o processo de instalação do ambiente de desenvolvimento em Java, fale sobre suas quatro principais plataformas.

#### Nunca programou?

Educador, neste item você deve convencer que o curso dará suporte até para quem nunca programou. Explique que o curso começará com tópicos sobre programação básica em Java chegando a níveis mais avançados.

Neste começo, o importante é estimular o aluno e ressaltar às vantagens em programar em Java, apresentar as IDEs e suas vantagens e, principalmente, mostrar as diferenças entre as plataformas Java, esclarecendo a diferença de Java e JavaScript, que é uma das principais dúvidas dos alunos neste contexto. Essas diferenças são facilmente encontradas na internet, veja a seguir:

* Uma discussão sobre Java vs Javascript (inglês): <https://www.quora.com/What-is-the-difference-between-Java-and-JavaScript>
* Tirinha sobre a diferença entre Java e Javascript (inglês):



#### - IDE – Integrated Development Environment

Educador, um IDE é um ambiente de desenvolvimento integrado, em outras palavras é o *software* utilizado para programar. Fale sobre as várias opções de IDEs gratuitos como: NetBeans, Eclipse, entre outras.

* + 1. **- JDK – Java Development Kit**

O JDK é o kit básico para desenvolvimento em Java, sem ele não é possível ter acesso as classes básicas do Java como a System, a String, entre outras que serão ensinadas no decorrer do curso.

**1.2.3 - Java não é JavaScript!**

Educador, alunos iniciantes em programação geralmente confundem JavaScript com Java, sendo assim, é importante ressaltar as diferenças entre elas. Ressalte que Java é uma linguagem de programação orientada a objetos tradicional, que precisa de um compilador, um interpretador e uma máquina virtual para ser executada. JavaScript é uma linguagem de Script, não é essencialmente orientada a objetos e não necessita de um compilador ou uma máquina virtual para ser executado, necessitando apenas de um interpretador para ser executada.

### 1.3 – Primeiros Passos: Como escrever um programa em Java

Antes de começar a programar, é necessário preparar as ferramentas básicas para o Java funcionar.

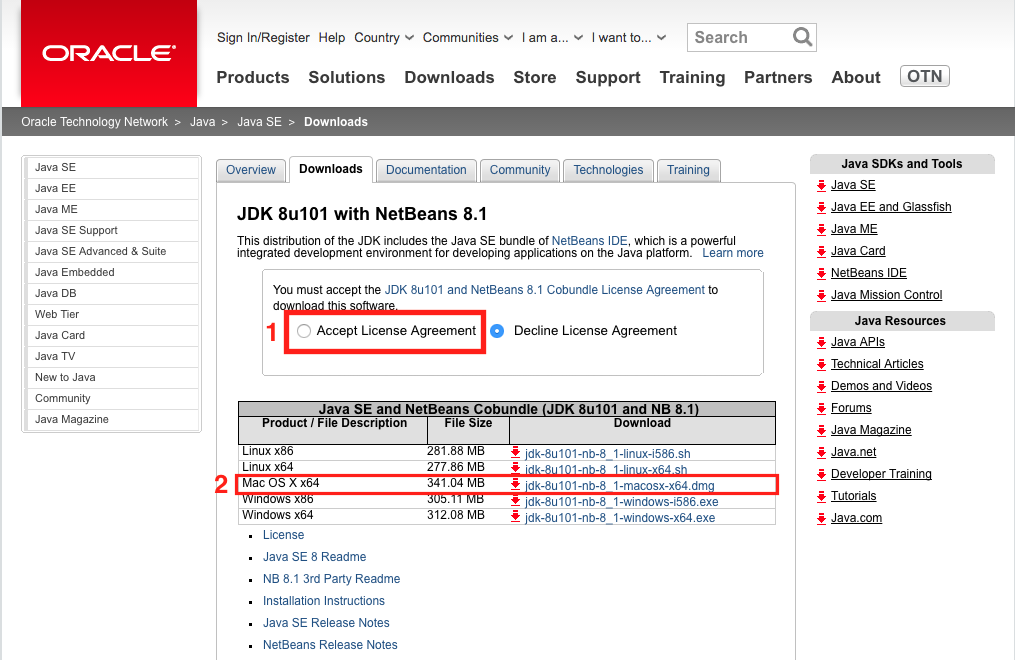
**Instalação JDK + Netbeans**

Instale o pacote completo, que contém o JDK e a IDE Netbeans.

1. Acesse o site http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html e escolha a opção “Netbens with JDK 8”. Acompanhe a imagem a seguir:



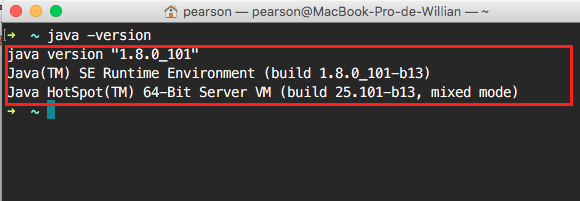
1. Na nova página, aceite os termos de licença (1) e faça o download do pacote conforme o sistema operacional que você usa (2). No caso do exemplo, utilizaremos Mac OSX. Veja a imagem a seguir:



1. Após realizar o download, abra o pacote baixado e prossiga nos passos da instalação. Esse passo depende de cada Sistema Operacional, faça uma instalação convencional.
2. Para testar o Netbeans abra-o e realize a configuração inicial. Para testar o JDK abra o terminal (CMD no Windows) e realize o comando:

java –version

Se a instalação foi feita corretamente, uma mensagem com a versão do Java instalado aparecerá como na imagem a seguir:



O ambiente está pronto para programar.

Após preparar o ambiente crie o projeto realizando o passo a passo disponível no livro do aluno.

### 1.4 – Primeiro programa: Hello World

Educador, lembre-se que a estrutura public static void main (String[] args){…} é criada automaticamente pela IDE quando marcada a opção “Criar Classe Principal”, o que torna o projeto executável. Caso essa opção não seja marcada à linha de comando System.out.println(“Hello Word”) ela não poderá ser executada.

Educador, nos exemplos anteriores, assim como nos seguintes, é importante atentar-se as janelas mostradas, pois podem ocorrer diferenças nas interfaces da IDE instalada nos computadores. A IDE adotada é a NetBeans 8.1 no OSX El Capitan, verifique se há diferença entre a versão da escola e o OS adotado. O código deverá ficar assim:

1. **public** **static** **void** main(String[] args) {
2. // TODO code application logic here
3. System.out.println("Hello World");
4. }

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 1 > Exemplos > 1.4** no seu Netbeans.

### 1.5 – Classes, Objetos e Instâncias

É comum o aluno sentir dificuldade na distinção entre classe, objeto e instância. No início, será um assunto bastante abstrato e complexo de se absorver, mas é importante que o aluno saiba que até o final do curso ele dominará esse assunto. Para isso, deve-se criar paralelos entre o mundo real e a computação e, explicitar exemplos práticos e palpáveis para que o aluno obtenha mais familiaridade com o assunto e passe a ter o domínio O intuito é expor ao aluno à programação para que ele não se sinta desmotivado.

### 1.6 – Programa Hello World (Explicação)

Pegue um algoritmo do tipo descrição narrativa e faça uma comparação com a classe, isso ajudará a conexão entre os assuntos e a aproximará a linguagem do aluno com a de programação.

Educador apresente os principais elementos em um código Java, ou seja, uma simples apresentação da estrutura de um código é o suficiente para que o aluno compreenda esse primeiro programa e sua linha lógica. Faça colocações mostrando que a classe está em maior nível que os demais elementos e o método está dentro de uma classe, isso pode ser facilmente observado pelas chaves nas quais percebe-se o nível de cada atributo mostrado. Ressalte a questão da identação para organização do código. Cada um desses atributos será melhor explanado mais adiante. O código deverá ficar como a seguir:

1. **public** **class** Main {
3. /\*\*
4. \* @param args the command line arguments
5. \*/
6. **public** **static** **void** main(String[] args) {
7. // TODO code application logic here
8. }
10. }

Listagem 2.2

### Exercícios

As perguntas têm por objetivo fixar os principais conceitos abordados durante a aula. Para isso, viabilize o tempo necessário para o aluno responder às perguntas, acompanhando-os nas dúvidas. Para correção das perguntas, considere as respostas a seguir:

1. **Quais são as principais plataformas do Java e para que servem?**

Resposta: O Java possui quatro plataformas principais: Java SE (Standard Edition), Java EE (Enterprise Edition), Java ME (Micro Edition) e JavaFX. Java SE serve para aplicações desktop em geral e Java EE é uma extensão para aplicações web e servidores. Para multimídia usa-se a plataforma JavaFX, também conhecida como Flash Players, com integração ao Java. Finalmente, na criação de sistemas embarcados como celulares mais antigos e sistemas de controles industriais, utiliza-se Java ME.

1. **O que são IDEs? Cite exemplos.**

Resposta: IDE, do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um software que reúne características e ferramentas de apoio ao desenvolvimento de softwares com o objetivo de agilizar o processo. Alguns exemplos são: Netbeans, Eclipse, DrJava, etc.

1. **Qual é a importância do JDK para uma aplicação Java?**

Resposta: É um kit de desenvolvimento que atua tanto como um compilador, quanto um interpretador

1. **O que é um compilador e um interpretador?**

Resposta: Compilador é uma ferramenta que torna a linguagem de inteligível pelo computador, tornando-a uma linguagem binária. Um interpretador verifica e aponta erros de semântica de programação em seu programa.

1. **Qual a diferença entre Java e JavaScript?**

Resposta: O Java é usado para criações stand-alone, que são softwares maiores que se executam por si só ou por applets, parte integrada de algum software. O JavaScript serve para implementar funcionalidades web que um simples HTML e CCS não poderiam alcançar.

1. **Qual a diferença entre Classe, Objeto e Instância?)**

Resposta: O nível mais alto neste grupo é a classe, ela define as especificações e o comportamento de um objeto, por meio de atributos (propriedades) e métodos (ações). O Objeto aplica as especificações definidas por uma classe, ou seja, existe a partir de um “molde” (classe), eles são instâncias de uma classe. Sendo assim, as instâncias de uma classe compartilham o mesmo conjunto de propriedades, embora o conteúdo dessas propriedades seja diferente entre si.

1. **Qual é a função da classe principal do Java?**

Resposta: A classe principal do Java é o ponto de partida e de término da execução do programa.

1. **Qual instrução utilizada para exibir um texto na tela?**

Resposta: Use o exemplo Hello World.

System.out.println("Hello World");

1. **Qual é o único método obrigatório do Java?**

Resposta: O único método obrigatório do Java é o Método main().

1. **Cite com exemplos a diferença entre Classes, Objetos e Instâncias.**

Resposta: Uma Planta é uma classe que representa os vegetais, a Árvore é um objeto da classe Planta e a Goiabeira é uma instância do objeto Árvore. Ao mesmo tempo, Goiabeira também é uma instância da classe Planta, que está no topo dessa hierarquia.

### TDP

Ainda considerando a teoria relacionada à prática, é necessário que você seja um orientador do processo de desenvolvimento desta atividade, garantindo que o aluno tenha o conhecimento necessário para tornar-se um profissional qualificado para o mercado de trabalho.

Nesse capítulo foi visto que o método main possui alguns argumentos que não foram explicados:

1. **public** **static** **void** main(String[] args) {
2. System.out.println("Hello World");
3. }

Listagem 2.4

#### Linha de comando

O parâmetro args é utilizado para quando uma aplicação Java é executada via linha de comando. Esse parâmetro recebe uma lista de argumentos do tipo String. Estes argumentos são passados pela linha de comando ou internamente pelo IDE utilizado.

#### Tarefa

Construa um programa no bloco de notas de seu computador (não utilize IDE) que possua os seguintes requisitos:

1. Primeiramente crie o programa Hello World e execute-o via linha de comando.
2. Atualize o código para exibir “Hello **seu** **nome**“, sendo que “seu nome” será um argumento enviado via linha de comando.

Agora que você conseguiu executar um programa Java usando linhas de comando, viu que, sem sombra de dúvidas, o uso de um IDE facilita e aumenta a produtividade da programação.

Você se lembra do Nelson? Ele gosta de Java e provavelmente vai pedir que o projeto dos filmes seja codificado nessa linguagem.

#### Tarefa

1. Faça uma pesquisa sobre os melhores IDEs para Java. Escolha um deles, faça download e instale-o no seu computador.

#### SUGESTÃO:

#### Tarefa 1

A seguir uma sugestão de código utilizado para receber um texto via linha de comando e exibi-lo na tela.

1 - Abra o programa Bloco de Notas ou qualquer editor de texto do seu computador e insira o código a seguir:

1. **public** **final** **class** Main {
2. **public** **static** **void** main(String[] args) {
3. String userInput = "";
4. **for** (String text: args) {
5. userInput += text;
6. }
7. System.out.println("Hello " + userInput);
8. }
9. }

2 - Salve o arquivo e dê o nome de Main.java.

3 - Abra o terminal e navegue até a pasta em que a classe está guardada utilizando o comando:

cd caminho/até\_a\_pasta\_de/Main.java

4 - Compile o programa utilizando o comando:

javac Main.java

5 - Por fim, execute o programa com o comando:

java Main *Seu Nome*

O seu nome será exibido seguido da saudação Hello:

> Hello Jorge

**Material de Apoio:** Para conferir o TDP deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 1 > TDP** no seu Netbeans.

#### SUGESTÃO:

#### Tarefa 2

Existem várias opções de IDEs gratuitos disponibilizados para download, conforme a seguir:

* **Netbeans** (<http://netbeans.org>) - é o IDE mais utilizada mundialmente e o mais recomendado para iniciantes em Java. É potente, rápido e pode suportar todas as plataformas Java, desde a SE até a FX;
* **Eclipse:** (<http://www.eclipse.org>) - um dos IDEs mais conhecidos e também, como o NetBeans, indicado para iniciantes;
* **DrJava:** DrJava é um IDE Java leve, designado para iniciantes e estudantes da linguagem Java e desenvolvido pelo grupo JavaPLT. Sua interface foi desenvolvida utilizando o framework Swing, que será alvo de estudo no último capítulo, por isso sua aparência irá variar de acordo com o sistema operacional que ele está rodando;
* **BlueJ:** foi desenvolvido principalmente com propósito educacional para dar suporte ao ensino e aprendizado de programação orientada a objetos;
* **IntelliJ:** um dos IDEs mais completo. Bastante parecido com o Netbeans, além de suportar programação em Java, suporta diversas outras linguagens. Possui uma Interface Builder com integração na criação de apps Android, um IDE novo e bem promissor.

### COTEÚDOS COMPLEMENTARES

Educador, para agregar novos conhecimentos sobre os assuntos desta aula, conheça algumas sugestões de conteúdo:

* DEITEL.  **Java – Como Programar**. Prentice Hall.  8a Edição, 2010

### REFERÊNCIAS

* DEITEL.  **Java – Como Programar**. Prentice Hall.  8a Edição, 2010
* JOHANSEN, ANDREW.  **Java – The Ultimate Beginner’s Guide!**. Publicação independente.  Kindle Edition, 2015

### Conheça todas as referências utilizadas pelo autor para elaboração dessa aula no Livro do Aluno

* QUORA. **What is the Diference between Java and JavaScript**. Disponível em: <https://www.quora.com/What-is-the-difference-between-Java-and-JavaScript>. Acesso em: 13 de. 2016.
* QUORA. **What are the best IDEs for Java Programmers**. Disponível em: <https://www.quora.com/What-are-the-best-IDEs-for-Java-programmers>. Acesso em: 13 de. 2016.

## Aula 2 - Programação em Java

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para:

* Entender o básico da sintaxe e elementos da linguagem Java.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 60 minutos de aula expositiva;
* 20 minutos para tirar as dúvidas dos alunos;
* 10 minutos para desenvolver as atividades propostas para a turma.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Palavras reservadas;
* Comentários no código;
* Nomeando classes;
* Definição de pacote;
* Instruções de importação;
* Variáveis;
* Métodos;
* Tipo de dados primitivos;
* Strings;
* Operadores ;
* Operadores condicionais e instruções de controle;
* Escopo da variável;
* Loops.

### PONTOS IMPORTANTES

Para garantir ao aluno um aprendizado significativo, resgate os principais conceitos de cada tópico abordado na aula. Neste momento, é importante que você utilize dicas e sugestões para reforçar os temas trabalhados, oferecendo também exemplos que possibilitem a relação entre teoria e prática.

### 2. – Programação em Java

Educador,este capítulo introduz o aluno na programação em Java com os conceitos básicos.

### 2.1 – Palavras reservadas

Explique que as palavras reservadas são palavras de uso exclusivo da sintaxe da linguagem, elas possuem propriedades próprias e não podem ser utilizadas como identificadores, ou seja, nomes de variáveis.

Educador, no link a seguir é possível visualizar uma pequena lista de palavras reservadas.

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html>

### 2.2 – Comentários no código

Educador, informe que para obter uma melhor organização e manter a lógica do código fácil de ser entendida basta inserir comentários, eles servem para documentar ou explicar o funcionamento de trechos de código.

Educador, acesse o artigo sobre comentários no link a seguir:

<http://www.oracle.com/technetwork/articles/java/index-137868.html>

### 2.3 – Nomeando Classes

Educador, explique que é possível definir os nomes das classes e como melhor indicá-las. Ensine como o aluno deve evitar a colisão de nomes ao utilizar o mesmo nome de classe para expressar dois conceitos diferentes.

Acesse o artigo sobre nomeação de classes e pacotes Java no link a seguir: <http://www.oracle.com/technetwork/java/codeconventions-135099.html>

### 2.4 – Definição de pacote

Educador, um pacote pode ser criado para organização dos arquivos do projeto, sendo assim, explique ao aluno como definir um pacote e fale sobre o mecanismo de *namespace*.

Acesse o link a seguir sobre nomeação de classes e pacotes Java: <http://www.oracle.com/technetwork/java/codeconventions-135099.html>

### 2.5 – Instruções de importação

Quando é necessário fazer uso de uma classe que está em um pacote diferente, é preciso importá-la, explique ao aluno como realizar essa importação.

Acesse o link a seguir sobre documentação Java:

<https://docs.oracle.com/javase/tutorial/java/package/usepkgs.html>

### 2.6 – Variáveis

Educador, variáveis são espaços na memória que guardam valores, usadas para manipulá-los, explane sobre esse tópico com o aluno.

Para agregar mais informações sobre variáveis acesse o link a seguir:

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>

### 2.7 – Métodos

Os métodos são ações ou funções que um objeto pode executar, exponha esse tema em sala de aula.

#### 2.7.1 - Métodos construtores

Educador, fale que os métodos construtores são invocados no momento que o seu respectivo objeto é instanciado e demonstre como eles são declarados.

#### 2.7.2 - Definição de classe com um construtor

Educador, explique que os construtores de uma classe podem ser definidos com ou sem parâmetros, tendo como intuito pré-inicializar alguma variável do objeto.

#### 2.7.3 - Outros métodos

Existem outros métodos que rodam funções especificas e podem possuir algum tipo de retorno, fale sobre a diferença entre os métodos construtores

#### 2.7.4 - Métodos estáticos e de instância

Métodos de instância dependem do estado de uma instância de objeto específica para seus comportamentos. Métodos estáticossão aqueles cujos comportamentos não dependem de qualquer estado do objeto. Esclareça essas diferenças para o seu aluno.

Acesse o link a seguir sobre métodos Java:

<https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>

### 2.8 – Tipo de dados primitivos

Educador, explique que os tipos de dados primitivos são aqueles que não possuem estrutura de objetos e seus possíveis valores são literais, não sendo uma representação orientada a objetos.

Saiba mais acessando o link a seguir:

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

### 2.9 – Strings

Educador, comente que as Strings são cadeias de caracteres e com elas pode-se manipular textos. Strings serão amplamente utilizadas neste curso e em programas, sendo assim é importante que você tenha domínio sobre esta classe (String).

Agregue conhecimento acessando o link a seguir:

http://docs.oracle.com/javase/6/docs/api/java/lang/String.html

#### 2.9.1 – Strings

Educador, explique que Stringssão objetos de primeira classe do tipo String, que comparado à linguagem C, é semelhante ao tipo de dado primitivo char, mas este pode manter somente um caractere Unicode, e no String pode-se armazenar palavras ou frases como valor.

#### 2.9.2 – Concatenando strings

Ressalte que para se concatenar Strings usa-se o sinal de soma (+)

#### 2.9.3 - Encadeando chamadas de métodos

Esclareça ao aluno que encadeia-se objetos para que uma modificação sempre retorne à modificação e não a original.

### 2.10 – Operadores

Os operadores são utilizados para operações aritméticas.

Saiba mais acessando o link a seguir:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

### 2.10.1 – Operadores aritméticos da linguagem Java

Educador, ressalte qual a utilidades dos operadores aritméticos e demonstre como eles são listados.

### 2.10.2 – Operadores adicionais

Educador, além dos operadores aritméticos fale também sobre os operadores adicionais.

### 2.11 – Operadores condicionais e instruções de controle

Educador, fale que os operadores condicionais são utilizados para realizar decisões condicionais. Essas decisões são ramificações que o fluxo do programa sofre quando uma asserção imposta pelo programador é válida.

#### 2.11.1 - Operadores relacionais e condicionais

Explique para que servem os operadores relacionais e condicionais, utilize a tabela disponibilizada no Livro do aluno.

#### 2.11.2 - A instrução if

Educador, informe que a instrução if avalia as condições booleanas entre parênteses e as executa se estas forem verdadeiras.

### 2.12 – Escopo da variável

Educador, fale sobre o escopo da variável e como ocorre um erro de compilação.

Acesse o link a seguir:

https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

#### 2.12.1 - A instrução else

Explique em quais situações é utilizada a instrução else.

Acesse os links a seguir:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

### 2.13 – Loops

Por último, fale sobre os loops, que eles são estruturas de repetição condicionais.

#### 2.13.1 - O que é um loop?

Educador, explique que para executar um bloco de instruções tantas vezes quanto uma condição retornar verdadeira, utilizamos um loop.

#### 2.13.2 - Loops while

Relate que loops while é uma construção de programação que executa repetidamente certas ações enquanto uma condição não é atendida.

#### 2.13.3 - Ramificação de loop

Explique como resgatar um loop antes que a expressão condicional seja avaliada com false.

Documentação Java:

* <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html>
* <https://docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html>
* <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html>

### Exercícios

As perguntas têm por objetivo fixar os principais conceitos abordados durante a aula. Para isso, viabilize o tempo necessário para o aluno responder às perguntas, acompanhando-os nas dúvidas. Para correção das perguntas, considere as respostas a seguir:

1. **O que são palavras reservadas? Cite exemplos.**

Resposta: São palavras especialmente designadas e pré-estabelecidas para uso de certos métodos e estruturas, por isso estas palavras não devem ser utilizadas para nomear construções próprias em Java. Alguns exemplos de palavras reservadas são: class, private, final, enum, int, while, if, etc.

1. **Para que servem os comentários? Quais são os tipos de comentários? Cite exemplos.**

Resposta: Servem para especificar o nome do autor, tipo de programa e facilitar a compreensão e organização de certos trechos do programa. Os comentários podem ser de uma única linha ou múltiplas linhas. Exemplo:

// Este é um comentário de uma linha

/\* Este

\* é

\* um

\* comentário

\* de

\* múltiplas

\* linhas

\*/"

1. **Como as classes devem ser declaradas/nomeadas?**

Resposta: Pela convenção *camel-case*, começando com letra maiúscula e, em caso de nome composto, não inserir espaço e iniciar o segundo nome também com letra maiúscula. Não iniciar o nome da classe com números ou caractere especial.

1. **O que são pacotes?**

Resposta: É um mecanismo de *namespace* e atua como um organizador, que torna os nomes neste pacote exclusivos dentro dele, podendo ser utilizados novamente em outros pacotes, mas não no mesmo pacote. Também serve para você organizar seus aplicativos mais complexos em unidades de funcionalidade discretas.

1. **Para que serve uma instrução de importação? Mostre um exemplo de importação de uma classe específica e outro que importa de um pacote inteiro.**

Resposta: As importações (import) se assemelham à definição de classe. Muitas das classes não estão localizadas por padrão no compilador e devem ser importadas para serem utilizadas. Uma instrução de importação informa ao compilador Java onde estão localizadas certas classes as quais você fará uso. import classes.Car; e import java.awt.\*;.

1. **O que são as variáveis e constantes? Quando utilizar uma constante no lugar de uma variável?**

Resposta: Variáveis referenciam um valor mutável. Ela pode ser simples ou de escopo. A variável de escopo distingue e diferencia cada instância em que ela está contida, além de definir o estado desta. As constantes referenciam um valor não mutável. Variáveis de escopo também podem ser constantes, ou seja, definem o estado da instância, porém não podem ser alteradas. Quando é necessário que um valor não seja alterado, use constantes. Caso contrário, use variáveis.

1. **Quando utilizar a palavra reservada "this"?**

Resposta: Quando há duas variáveis com o mesmo nome, usa-se a palavra this para se referir a uma delas.

1. **O que são Strings? Como concatenar Strings?**

Resposta: Variáveis do tipo String são objetos de primeira classe e podem armazenar palavras ou frases como valor. Para concatenar Strings, escreve-se a palavra ou frase entre aspas e usa-se o sinal "+" para concatena-la com outra String ou com outras variáveis e objetos.

1. **Cite os tipos de operadores.**

Resposta: Os tipos de operadores são aritméticos, condicionais e instrutores de controle.

1. **Quais são os modificadores de acesso? Cite as diferenças.**

Resposta: Os modificadores de acesso são: public, private e protected. O modificador public significa que as informações contidas na classe podem ser acessadas livremente por qualquer outra classe. O modificador private significa que as informações da classe podem ser acessadas apenas por ela mesma. O modificador protected significa que as informações da classe só podem ser acessadas por ela mesma e pelas subclasses.

1. **Explique o funcionamento da instrução "if". Quando utilizar a instrução "else"?**

Resposta: if é um operador condicional que executa os comandos dentro deles somente se seu método for satisfeito, ou seja, quando a instrução if for verdadeira. O else é utilizado apenas quando a condição na instrução if é falsa.

1. **O que são Loops? Quais tipos de loops existentes? O que é ramificação de loop?**

Resposta: É uma construção de programação que executa repetidamente enquanto uma condição não é atendida. Os tipos de loops são: while e for. Ramificação loop serve para resgatar um loop antes que a expressão condicional seja avaliada como false, na linguagem Java temos a instrução break.

### TDP

Ainda considerando a teoria relacionada à prática, é necessário que você seja um orientador do processo de desenvolvimento desta atividade, garantindo que o aluno tenha o conhecimento necessário para tornar-se um profissional qualificado para o mercado de trabalho.

#### 2.16.1 - A reunião

O dia da reunião chegou e adivinha? Você se destacou! Você havia sido o único a preparar o ambiente e a realizar uma pesquisa de mercado (sim a pesquisa sobre filmes que você fez, é chamada de pesquisa de mercado). Nelson gostou muito da pesquisa de mercado, pois é uma etapa do processo de desenvolvimento de software designada aos gerentes.

O resultado desta reunião foi que você agora é o substituto de Nelson. Ele vai tirar um período de licença médica e você deverá tocar o barco daqui para frente.

#### 2.16.2 - Estudo de caso

Nelson falou com o cliente e coletou uma série de informações para você. Leia o caso:

“O cliente é chamado **Júlio**. Júlio é um cinéfilo, crítico e colunista de um jornal famoso. Ele é simplesmente obcecado por filmes e qualquer outra arte audiovisual.

Júlio possui uma agenda, onde anota a data de lançamento de praticamente todos os filmes. Ele sempre monitora esta agenda, pois quando tais datas chegam, ele vai ao cinema, assiste o filme e publica uma crítica sobre ele em uma coluna do jornal. Faz isso religiosamente e praticamente todas as semanas ele assiste alguns filmes.

No início do ano Júlio saiu de férias e foi fazer um passeio em seu barco à vela, e como era de se esperar, não desgrudou de sua agenda.

Por um pequeno descuido, ele a deixou cair no mar. Ela afundou como uma âncora.

Júlio no desespero, abortou suas férias e tentou reescrever sua agenda, que tinha registros lançamentos de filmes até o ano 2020. Perdeu dois dias árduos nesta empreitada.

Agora, para não ter férias perdidas, resolveu contratar nossa empresa para criar um aplicativo desktop que cadastre estes filmes e os mantenham seguros”

A próxima etapa é a que chamamos de Storyboard.

#### 2.16 - Storyboard

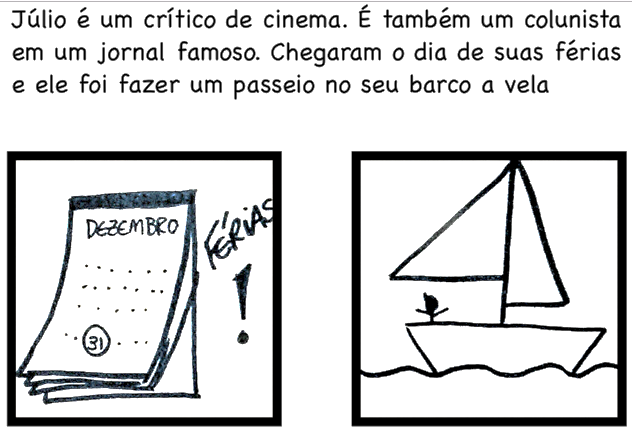
O storyboard é uma sequência de imagens que narra uma história de forma simples e objetiva. Em Análise de Projetos, ela é uma forma de simular o uso de um produto antes mesmo dele começar a ser desenvolvido.

Esta simulação revela se o produto realmente é viável ou não.

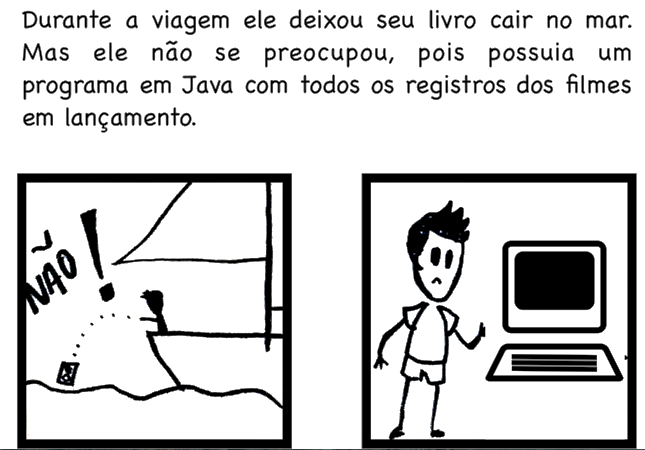
Dado um caso, ou problema, o storyboard deve narrar o mesmo caso, introduzindo o produto como uma solução.

#### Tarefa

Crie um storyboard que narre o acidente de Júlio, mas que com a introdução do aplicativo, prive-o da necessidade de reescrever sua agenda. A seguir a sugestão de storyboard para este exemplo:







O ideal é que o projeto siga usando este storyboad, que foi desenhado à mão, sendo possível fazê-lo no quadro negro, criando um esboço.

### COTEÚDOS COMPLEMENTARES

Educador, para agregar novos conhecimentos sobre os assuntos desta aula, conheça algumas sugestões de conteúdo:

* DEITEL.  **Java – Como Programar**. Prentice Hall.  8a Edição, 2010

### REFERÊNCIAS

* ORACLE. **Java™ Platform, Standard Edition 8 API Specification**. Disponível em: <https://docs.oracle.com/javase/8/docs/api/overview-summary.html>. Acesso em: 13 dec. 2016.
* IBM. **Introdução a programação Java.** Disponível em: **<**https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1>. Acesso em: 13 dec. 2016.
* DESARROLLOWEB. **Human Computer Interaction – Storyboarding.** Disponível em: <http://desarrolloweb.dlsi.ua.es/cursos/2015/hci/storyboarding>>. Acesso em: 13 dec. 2016.

### Conheça todas as referências utilizadas pelo autor para elaboração dessa aula no Livro do Aluno:

* ORACLE. **Keywords**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/\_keywords.html>. Acesso em: 13 de. 2016.
* ORACLE. **How to Write Doc Comments for the Javadoc Tool**. Disponível em: <http://www.oracle.com/technetwork/articles/java/index-137868.html>. Acesso em: 13 de. 2016.
* ORACLE. **Code Conventions – Naming Classes Conventions**. Disponível em: <http://www.oracle.com/technetwork/java/codeconventions-135099.html>. Acesso em: 13 de. 2016.
* ORACLE. **Code Conventions – Naming Packages Conventions**. Disponível em: <http://www.oracle.com/technetwork/java/codeconventions-135099.html>. Acesso em: 13 de. 2016.
* ORACLE. **Using** **Package Members**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/package/usepkgs.html>. Acesso em: 13 de. 2016.
* ORACLE. **Variables**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>. Acesso em: 13 de. 2016.
* ORACLE. **Methods**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>. Acesso em: 13 de. 2016.
* ORACLE. **Data Types**. Disponível em: <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>. Acesso em: 13 de. 2016.
* ORACLE. **String**. Disponível em: <http://docs.oracle.com/javase/6/docs/api/java/lang/String.html>. Acesso em: 13 de. 2016.
* ORACLE. **Access Control**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html>. Acesso em: 13 de. 2016.
* ORACLE. **Operators**. Disponível em <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>. Acesso em: 13 de. 2016.
* ORACLE. **IF**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html>. Acesso em: 13 de. 2016.
* ORACLE. **While**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html>. Acesso em: 13 de. 2016.
* ORACLE. **For**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html>. Acesso em: 13 de. 2016.
* ORACLE. **For Each**. Disponível em: <https://docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html>. Acesso em: 13 de. 2016.

## Aula 3 –Java em Prática

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para:

* Interpretar um problema com a visão de desenvolvedor de software;
* Implementar uma solução de software baseado no problema interpretado.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Análise de requisitos do projeto;
* Mão na massa.

### PONTOS IMPORTANTES

Para garantir ao aluno um aprendizado significativo, resgate os principais conceitos de cada tópico abordado na aula. Neste momento, é importante que você utilize dicas e sugestões para reforçar os temas trabalhados, oferecendo também exemplos que possibilitem a relação entre teoria e prática.

### 3.1 – Análise de Requisitos do Projeto

#### 3.1.1 Definindo os requisitos

O documento de requisitos deve conter todas as informações necessárias para o desenvolvimento de software. Ele deverá esclarecer todas as dúvidas e mostrar as funcionalidades e regras que o software implementará.

Educador, dê exemplos de requisitos não funcionais, conforme a seguir:

* **Performance**: define o quão rápido e eficiente é a execução do app. Um app com lentidão, travamento e que exija um alto consumo de energia, certamente possui baixa performance.

* **Usabilidade**: define o quão fácil e intuitivo é o uso do app. Botões e textos pequenos ou exagerados, cores que cansam a vista, difícil acesso as funcionalidades são características de uma usabilidade ruim.
* **Manutenibilidade**: define o quão fácil e rápido a manutenção do app poderá ser.
* **Segurança**: define quão seguro serão os dados mantidos pelo software, sendo assim protegido.

**Educador,** neste momento foi mostrado outras responsabilidades de um desenvolvedor de software. O programador não se limita em apenas codificar, ele deve coletar, estudar, processar e encontrar soluções para os problemas. No decorrer do curso as fases de análise e design de software serão abordadas. Atente-se aos próximos passos, pois serão uma sequência mecânica de etapas a serem seguidas.

### 3.2 – Mão na massa

Educador, por ser um passo a passo longo realize-o junto aos alunos, projete no telão a execução do passo a passo e peça que os alunos o acompanhe.

#### 3.2.1 - Criando uma nova classe

Procure utilizar a nomenclatura do texto, agilizando a detecção de possíveis erros dos alunos.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 3 > Exemplos > 3.2.1** no seu Netbeans.

#### 3.2.2 - Criando os atributos

Educador, utilize essa prática para obter exemplos de atributos e escreva-os em seu quadro. Para melhor entendimento adote o exemplo do texto.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 3 > Exemplos > 3.2.2** no seu Netbeans.

#### 3.2.3 - Testando os atributos da classe Car

Educador, erros comuns como a falta de fechamento de bloco ou de ortografia podem ocorrer, sendo assim, atente-se a eles e reforce tais pontos para que sejam minimizados.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 3 > Exemplos > 3.2.3** no seu Netbeans.

### Exercícios

As perguntas têm por objetivo fixar os principais conceitos abordados durante a aula. Para isso, viabilize o tempo necessário para o aluno responder às perguntas, acompanhando-os nas dúvidas. Para correção das perguntas, considere as respostas a seguir:

1. **O que você entende por requisitos funcionais e não funcionais? Cite exemplos.**

Resposta: Os requisitos funcionais caracterizam as funcionalidades que o software terá e geralmente são definidas pelo cliente. Já os não funcionais não possuem finalidade clara de uso, pois estão implícitos, mas são partes-chaves do programa.

1. **O que são atributos? Cite mais dois sinônimos de atributos.**

Resposta: Os atributos são propriedades ou características que um objeto deve ter. Os sinônimos são: variável de escopo e propriedade.

1. **O que é necessário para adicionar um atributo a uma classe?**

Resposta: Especificar o modificador de acesso, um tipo, um identificador e, opcionalmente um valor.

1. **Para que serve o método toString(), de onde ele vem? Como é a sua implementação?**

Resposta: Serve para converter um objeto para notação de String. Geralmente utilizado para obter uma representação mais amigável do objeto ao tentar exibi-lo com String. Ele vem da classe Object.

Implementação:

@Override

public String toString() {

return "Alguma string";

}

1. **Como acessamos e atribuímos valor a um atributo de um objeto.**

Resposta: Primeiro devemos instanciar o objeto. Depois acessamos o atributo utilizando o caractere "." e atribuímos o valor.

Exemplo:

meuObjeto.atributo = "algum valor";

### TDP

Ainda considerando a teoria relacionada à prática, é necessário que você seja um orientador do processo de desenvolvimento desta atividade, garantindo que o aluno tenha o conhecimento necessário para tornar-se um profissional qualificado para o mercado de trabalho.

#### Fase de análise de requisitos

Depois de escutarmos o caso do cliente, é preciso formalizar os requisitos. Os requisitos são funcionalidades que o software deverá ter.

Após diversas reuniões com o cliente, o gerente comercial formalizou os seguintes requisitos que entrarão no contrato:

#### Projeto Filmes

O projeto filmes se trata de uma plataforma de cadastro e consulta de filmes e seriados. Detalhes como sinopse, data de lançamento e elenco estarão contidos no sistema. Os requisitos são os seguintes:

##### Requisitos não funcionais

1. O sistema deverá possuir uma interface gráfica.
2. O sistema deverá armazenar os dados em arquivo.
3. O sistema deverá funcionar totalmente offline.
4. O sistema deverá fazer tratamento de erros, mostrando mensagens de alerta caso o usuário insira algum dado inválido.

##### Requisitos funcionais

1. O sistema deverá cadastrar filmes
   1. Um filme terá como conteúdo:
      1. Nome
      2. Nome original
      3. Data de lançamento
      4. Gênero
      5. Duração
      6. Sinopse
      7. Produtoras
      8. Países de origem
      9. Orçamento
      10. Receita
2. O sistema deverá cadastrar Gêneros
   1. O gênero deverá ser utilizado no cadastro e edição de filmes
   2. Um gênero terá como conteúdo
      1. Nome
      2. Identificador
3. O sistema deverá cadastrar os Créditos do Filme
   1. O crédito de um filme terá como conteúdo:
      1. Equipe
         1. Uma equipe é formada pelas pessoas que estiveram nos bastidores do filme.
         2. Um integrante de equipe terá como conteúdo:
            1. Um nome
            2. Um cargo ou posição
      2. Elenco
         1. Um elenco é formado por atores do filme.
         2. Um ator terá como conteúdo:
            1. Um nome
            2. Nome do personagem
4. Os filmes deverão ser editáveis.
5. Os filmes deverão ser deletáveis.

Este é o documento inicial de requisitos. Você deve sentir-se a vontade em adicionar mais requisitos ao projeto. A próxima fase é localizar e definir as entidades que possivelmente virarão classes.

#### Tarefa

Localize e liste todas as possíveis classes do projeto.

Educador, liste todas as possíveis classes do projeto (sem levar em consideração Herança, Interface e Abstração):

* Filme;
* Gênero do filme;
* País da produtora;
* Companhia/Produtora do filme;
* Ator do filme;
* Membro do Elenco;
* Membro da Equipe;
* Cargo do membro da equipe.

Para que o material de apoio não fique muito extenso, não crie um cadastro de Cargo, País e Gênero. Estes três casos podem ser facilmente cadastrados automaticamente via código, podendo vir a ser considerado decisão de projeto, por exemplo. Para que o usuário não tenha que cadastrar país por país, pode-se disponibilizar todos os países. Para Gêneros e Cargo, podemos selecionar os mais utilizados, apenas para nível didático e deixá-los pré-cadastrados. Caso você queira implementar o cadastro destas entidades, sinta-se livre.

### COTEÚDOS COMPLEMENTARES

Educador, para agregar novos conhecimentos sobre os assuntos desta aula, conheça algumas sugestões de conteúdo:

* DEITEL.  **Java – Como Programar**. Prentice Hall.  8a Edição, 2010.
* PRESSMAN, ROGER S. **Engenharia de Software – Uma Abordagem Profissional.** ARTMED. 7a Edição, 2011.

### REFERÊNCIAS

* PRESSMAN, ROGER S. **Engenharia de Software – Uma Abordagem Profissional.** ARTMED. 7a Edição, 2011.

### Conheça todas as referências utilizadas pelo autor para elaboração dessa aula no Livro do Aluno:

* PRESSMAN, ROGER S. **Engenharia de Software – Uma Abordagem Profissional.** ARTMED. 7a Edição, 2011.

## Aula 4 - Métodos e Encapsulamento

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Mostrar os primeiros conceitos e mais básicos da programação orientada a objetos
* Aplicar conhecimentos sobre Métodos e Encapsulamento em uma aplicação simples elaborada na aula 3.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Métodos;
* Encapsulamento;
* Obtendo a entrada do usuário.

### PONTOS IMPORTANTES

Para garantir ao aluno um aprendizado significativo, resgate os principais conceitos de cada tópico abordado na aula. Neste momento, é importante que você utilize dicas e sugestões para reforçar os temas trabalhados, oferecendo também exemplos que possibilitem a relação entre teoria e prática. Não é necessário que a parte prática da aula realizada pelos alunos estejam idênticas a mostrada no livro do aluno, desde que estes trabalhos estejam minimamente condizentes com o trabalho proposto. Você, educador deve ser flexível se os alunos apresentarem soluções diferentes às apresentadas em aula. Em programação existem diferentes maneiras de se atingir o mesmo objetivo. Isto vale para todas as aulas.

Educador, esta aula é 100% prática, seguir o passo a passo é fundamental. Caso algum problema aconteça, você poderá utilizar o exemplo pronto que está zipado no material digital do professor.

### 4.1 – Métodos

Mais uma explicação, desta vez prática, sobre métodos. Métodos são as ações que um objeto pode exercer.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 4 > Exemplos > 4.1** no seu Netbeans.

### 4.2 – Encapsulamento

Tem a utilidade de isolar partes do programa para que o acesso a estas partes fique, de forma controlada, limitada. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 4 > Exemplos > 4.2** no seu Netbeans.

### 4.3 – Obtendo entrada do usuário

Neste tópico mostramos o uso da classe Scan para obter o que o usuário digita no teclado.

**Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 4 > Exemplos > 4.3** no seu Netbeans.

### 4.4 – Resumo

Retome os tópicos abordados na aula questionando aos seus alunos.

O que aprendemos hoje pessoal? Você pode fazer isso em forma de Quizz ou para engajar sua turma pode finalizar a aula, perguntando se alguém tem dúvidas sobre aula de hoje e quem se habilitaria a resumir o que foi visto, faça isso toda aula com um aluno diferente para treiná-los quanto ao desenvolvimento de comunicação e ao final da fala do aluno, resuma os tópicos.

Nessa aula adicionamos ações à classe, baseados nos conceitos de Métodos. Aprendemos a encapsular atributos e a criar um construtor para a Classe. Também vimos como instanciar um novo objeto e como o programa recebe a entrada do usuário.

### Exercícios

As perguntas têm por objetivo fixar os principais conceitos abordados durante a aula. Para isso, viabilize o tempo necessário para o aluno responder às perguntas, acompanhando-os nas dúvidas. Para correção das perguntas, considere as respostas a seguir:

1. **O que são métodos?**

Resposta: Métodos são ações designadas aos objetos.

1. **Quais são as categorias de métodos?**

Resposta: Métodos construtores, estáticos e de instância

1. **O que é e quais são os elementos de uma assinatura de método?**

Resposta: A combinação de elementos estruturais que definem um método é chamada de assinatura de métodos, que inclui elementos como: accessSpecifier, returnType, methodName, argumentList.

1. **O que é encapsulamento de atributos? Para que serve?**

Resposta: É isolar partes do programa para que o acesso a estas partes fique, de forma controlada, limitada. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações.

1. **Qual o intuito da criação de construtores para uma classe?**

Resposta: Criar construtores é uma maneira mais fácil de dar as características ao objeto. Um construtor permite que seja passado via parâmetro todas as informações básicas que o objeto deve conter, no momento em que ele é instanciado.

1. **Como instanciar um novo objeto? Utilize o exemplo do Carro.**

Resposta:

Car myCar = new Car("Chevrolet");

Scanner scanner = new Scanner(System.in);

System.out.print("Digite o tempo percorrido em minutos: ");

1. **Qual é a instrução utilizada para receber a entrada do usuário? Mostre um exemplo.**

Resposta:

timeInMinutes = scanner.nextInt();

1. **Para finalizar o projeto faça um programa que utilize a classe Car. O programa deve receber via entrada de usuário todas as características de um carro e armazenar em um objeto carro. Armazene quantos carros forem necessários, deixando no poder do usuário a hora de parar de cadastrar carros. Ao terminar o cadastro faça com que todos os carros andem (o usuário deverá fornecer o tempo de cada carro) e logo em seguida faça com que todos os carros parem (o usuário deverá fornecer o consumo em litros de cada carro). Após este processamento exiba o carro considerado mais econômico dentre os cadastrados pelo usuário.**

Resposta: **Material de Apoio:** Para conferir o exemplo deste tópico abra o projeto **Material de Apoio POO > Unidade 2 > Aula 4 > Exemplos > 4.5** no seu Netbeans.

~~Código no arquivo zip~~

### 4.6 – TDP

Ainda considerando a teoria relacionada à prática, é necessário que você seja um orientador do processo de desenvolvimento desta atividade, garantindo que o aluno tenha o conhecimento necessário para tornar-se um profissional qualificado para o mercado de trabalho.

#### 4.6.1 - Identificando métodos e atributos

Você está se saindo muito bem. O seu time de desenvolvimento possui um arquiteto de softwares. O arquiteto é responsável por modelar e organizar tecnicamente o projeto, além de tentar converter a linguagem dos requisitos do sistema para uma linguagem mais técnica.

João, seu arquiteto está com dificuldades em identificar os atributos e métodos das classes do projeto.

#### Tarefa

Ajude João. Identifique todos os atributos e métodos das classes. Não é necessário implementar, apenas liste.

### SUGESTÃO:

#### Identificando atributos e métodos

Como o aluno ainda não tem muita familiaridade com a programação, por enquanto faremos apenas uma listagem destes métodos e atributos. A implementação ficará por conta da aula seguinte.

* Filme
  + Atributos
    - Nome (String)
    - Nome original (String)
    - Data de lançamento (long)
    - Gênero (Gênero)
    - Duração (long)
    - Sinopse (String)
    - Produtoras (ArrayList<Companhia>)
    - Orçamento (Double)
    - Receita (Double)
    - Elenco (ArrayList<Elenco>)
    - Equipe (ArrayList<Equipe>)
  + Métodos
    - Mostrar receita com formato monetário
    - Mostrar orçamento com formato monetário
    - Mostrar duração com formato de minutos
    - Mostrar data de lançamento com formato de data “dd/mm/aaaa”
* Gênero
  + Atributos
    - Nome (String)
* País
  + Atributos
    - Nome (String)
    - ISO 3166-1 (String)
* Companhia
  + Atributos
    - Nome (String)
    - País (País)
* Ator
  + Atributos
    - Nome (String)
* Membro do Elenco
  + Atributos
    - Ator (Ator)
    - Personagem (String)
* Membro da Equipe
  + Atributos
    - Nome (String)
    - Cargo (Cargo)
* Cargo
  + Nome (String)

### COTEÚDOS COMPLEMENTARES

Educador, para agregar novos conhecimentos sobre os assuntos desta aula, conheça algumas sugestões de conteúdo:

* DEITEL.  **Java – Como Programar**. Prentice Hall.  8a Edição, 2010

### REFERÊNCIAS

* JOHANSEN, ANDREW.  **Java – The Ultimate Beginner’s Guide!**. Publicação independente.  Kindle Edition, 2015

### Conheça todas as referências utilizadas pelo autor para elaboração dessa aula no Livro do Aluno

* ORACLE. **Methods**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>. Acesso em: 13 de. 2016.
* ORACLE. **Access Control**. Disponível em: <https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html>. Acesso em: 13 de. 2016.