# Unidade 2

O objetivo principal da unidade 2 é fazer o primeiro contato do com a programação na prática. Nesta unidade não deveremos nos aprofundar muito em conceitos, mas deveremos encaminhar para uma dinâmica totalmente prática e objetiva, para que o aluno veja como é de fato programar. É importante dizer que devido a esse início os alunos poderão ter dificuldades principalmente na aula 2 desta unidade.

Seria ideal que os alunos fossem dispostos de modo que os que possuem maior facilidade de aprendizado estejam ao lado dos que possuem mais dificuldades, para que possam ir se auxiliando, pois problemas de lógica de programação são extremamente difíceis para o iniciante em programação.

Obs: Ao longo deste livro você encontrará links e informações de suporte em inglês, fique atento e procure se preparar pois muito conteúdo de programação, ainda está em inglês.

## 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. Nesse primeiro contato com o aluno você identificará os alunos que possuem um conhecimento sobre Java, possibilitando que você realize uma melhor disposição na sala de aula.

### – Começando a trabalhar com o Java

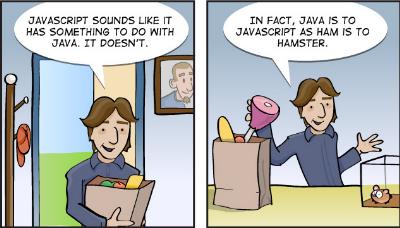
Neste tópico inicia-se o processo de instalação do ambiente de desenvolvimento em Java. É importante que você, educador, confira se os sites mostrados nesta aula ainda estão no ar ou se estão com um visual diferente. Eles estão em constante mudança.

#### Nunca programou?

Educador, nesta seção você deve convencer que o curso dará suporte até para quem nunca programou na vida. Explique que o curso começará com tópicos sobre programação básica em Java e chegará a níveis mais avançados que nem todos os cursos abrangem. O objetivo aqui é motivar o aluno.

Neste começo, o importante é 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. Estas diferenças são facilmente encontradas na internet:

* 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, uma IDE é um ambiente de desenvolvimento integrado. Em outras palavras é o software que utilizamos para programar.

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

O JDK é o kit básico para desenvolvimento em Java. Sem ele não temos acesso as classes básicas do Java como a System, a String e outras que você verá no progresso deste curso.

**1.2.3 - Java não é JavaScript!**

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

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

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

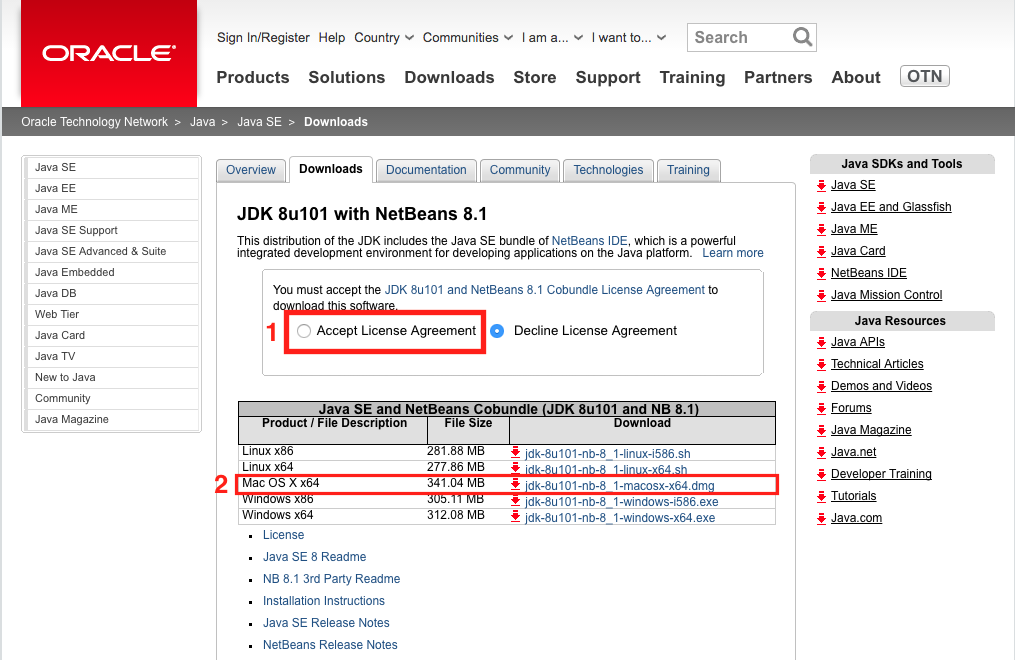
**Instalação JDK + Netbeans**

Iremos instalar 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”, vide imagem abaixo:



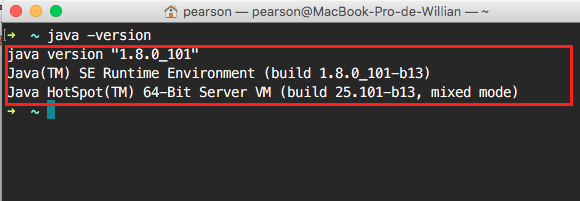
1. Na nova página, aceite os termos de licença (1) e faça o download do pacote conforme o sistema operacional que você usa (2). No caso do exemplo, utilizaremos Mac OSX, vide imagem abaixo:



1. Após realizar o Download, abra o pacote baixado e prossiga nos passos da instalação. Este passo depende de cada Sistema Operacional, mas não é necessário fazer nada de especial, apenas uma instalação convencional.
2. Pronto. 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 irá aparecer, vide imagem abaixo:



O ambiente está pronto para programar.

Após preparar o ambiente crie o projeto, disponível no livro do aluno, com sua sala, passo a passo.

### 1.4 – Primeiro programa: Hello World

**Dica: A estrutura public static void main (String[] args){…} é criada automaticamente pela IDE quando marcamos a opção “Criar Classe Principal”, o que torna o projeto executável. Caso está opção não seja marcada a linha de comando System.out.println(“Hello Word”); não poderá ser executada.**

Educador, nos exemplos anteriores, assim como nos seguintes, é importante se atentar que as janelas mostradas podem ter diferenças quanto a interfaces da IDE instalada nos computadores da escola. A IDE adotada aqui é a NetBeans 8.1 no OSX El Capitan, verifique se há diferenças com a versão da escola e também 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 áspero de se absorver, mas é importante que o aluno saiba que até o final do curso ele dominará este 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. Por enquanto não nos preocupamos com teoria. O intuito é expor o aluno à programação para ele não se sentir entediado.

### 1.6 – Programa Hello World (Explicação)

Acho que seria interessante pegar um algoritmo do tipo descrição narrativa e fazer uma comparação com a classe. Isso poderia ajudar a conexão entre os assuntos e aproximar a linguagem do aluno com a de programação.

**Educador**: Aqui, o que devemos fazer é apresentar os principais elementos em um código em Java. Uma simples apresentação da estrutura de um código é o suficiente para que o aluno tenha compreendimento deste primeiro programa e de sua linha lógica. Faça colocações mostrando que a classe está em maior nível que os demais elementos e o método estão dentro de uma classe, isso pode ser facilmente observado pelas chaves onde percebemos o nível de cada atributo mostrado. Frise bem a questão da identação para organização do código. Cada um destes atributos serão melhores explanados mais adiante. O código deverá ficar assim:

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

### 1.6 – 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 a 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 da comunicação e ao final da fala do aluno, resuma os tópicos.

### 1.7 – Exercícios

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

Resposta: Há quatro plataformas principais do Java: Java SE (Standard Edition), Java EE (Enterprise Edition), Java ME (Micro Edition) e JavaFX. Java SE é usada para aplicações desktop em geral e Java EE é uma extensão para aplicações web e servidores. JavaFX é uma plataforma para multimídia (também conhecido como Flash Players) com integração ao Java. Finalmente, Java ME é usada na criação de sistemas embarcados como celulares (mais antigos) e sistemas de controles industriais.

1. **O que são IDEs? Cite exemplos.**

Resposta: IDE, do ingles 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? (igual a questão 5?)**

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 programação (que é familiar ao Homem) inteligível pelo computador, tornando-a uma linguagem binária. Um interpretador verifica e aponta erros de semântica de programação em seu programa.

1. **Qual é a importância do JDK para uma aplicação Java? (não seria 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 (um pouco confuso). O JavaScript serve para implementar funcionalidades Web que um simples HTML e CCS não poderiam alcançar.

1. **Qual é diferença entre o Java e o JavaScript? (não seria Qual a diferença entre Classe, Objeto e Instância?)**

Resposta: Classe: 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). Objeto: Objetos aplicam as especificações definidas por uma classe, ou seja, existem a partir de um “molde” (classe). Eles são instâncias de uma classe. Instância: 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: Ela é o ponto de partida e de término da execução do programa.

1. **Qual instrução utilizada para exibir um texto na tela? Use o exemplo Hello World.**

Resposta:

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

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

Resposta:

Método main().

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

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

### 1.8 – TDP

Vimos neste capítulo 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

#### 1.8 - Linha de comando

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

#### 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 conseguimos executar um programa Java usando linhas de comando, vimos que, sem sombra de dúvidas, o uso de uma IDE facilita e aumenta a produtividade da programação.

Lembra de Nelson? Ele gosta de Java e provavelmente vai pedir que o projeto dos filmes seja codificado nesta linguagem.

#### Tarefa

Faça uma pesquisa sobre as melhores IDEs para Java. Escolha uma delas, faça download e instale-a no seu computador.

#### SUGESTÃO:

#### Tarefa 1

Uma sugestão de código utilizado para receber um texto via linha de comando e exibi-lo na tela é o seguinte:

1 - Abra o programa Bloco de Notas (ou qualquer editor de texto do seu computador) e insira o código abaixo:

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 que esta classe está guardada utilizando o comando:

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

Se você está utilizando Windows, utilize dir ao invés de cd. O comando dir no Windows só apresenta a estrutura de diretórios. O comando é o mesmo cd. Acho que esta anotação deve ser retirada.

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 gratuitas disponibilizadas para download. Separamos algumas delas:

* **Netbeans** (<http://netbeans.org>) - É a IDE mais utilizada mundialmente e a mais recomendada para iniciantes em Java. É potente, rápida e pode suportar todas as plataformas Java, desde a SE até a FX.
* **Eclipse:** (<http://www.eclipse.org>) - Uma das IDEs mais conhecidas é também, como o NetBeans, indicada para iniciantes.
* **DrJava:** DrJava é uma IDE Java leve, designada 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:** Uma das IDEs mais completas. Bastante parecida com o Netbeans, além de suportar programação em Java, suporta diversas outras linguagens. Possui um Interface Builder com integração na criação de apps Android. Uma IDE nova e bem promissora.

## 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. A aula 2 mostra rapidamente os conceitos básicos de Java e programação, mas por se tratar de uma aula extensa, economizamos em exercícios práticos, então esta aula será como uma referência, onde o aluno poderá consultar conceitos no decorrer do curso e trabalhos práticos.

Educador,o aluno poderá chegar sem nenhum conhecimento sobre programação, preparamos este capítulo para referência básica sobre a linguagem Java. Não é uma documentação, esta aula apenas pincela alguns conceitos de programação básicos e para melhor didática omitimos certas informações. Para cada seção você poderá consultar uma documentação oficial do Java.

### 2.1 – Palavras reservadas

As palavras reservadas são palavras de uso exclusivo da sintaxe da linguagem. São palavras que possuem propriedades próprias e não podem ser utilizadas como identificadores (nomes de variáveis).

Documentação Java: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/\_keywords.html

### 2.2 – Comentários no código

Uma forma de se organizar e manter a lógica do código fácil de ser entendida é inserindo comentários. Estes comentários servem para documentar ou explicar o funcionamento de trechos de código.

Artigo sobre comentários Java:

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

### 2.3 – Nomeando Classes

Existe uma sintaxe própria para criação de classes. A nomeação de classes e pacotes Java possui alguns padrões e convenções a serem seguidos.

Artigo sobre nomeação de classes e pacotes Java: <http://www.oracle.com/technetwork/java/codeconventions-135099.html>

### 2.4 – Definição de pacote

Um pacote pode ser criado para organização dos arquivos do projeto. É como uma estrutura de diretório ou também conhecido como namespaces.

Artigo 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.

Documentação Java:

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

### 2.6 – Variáveis

Variáveis são espaços na memória que guardam valores. Usadas para manipular valores.

Documentação Java:

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

### 2.7 – Métodos

São ações ou funções que um objeto pode executar.

#### 2.7.1 - Métodos construtores

São métodos que são invocados no momento que o seu respectivo objeto é instanciado.

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

Construtores de uma classe podem ser definidos com parâmetros ou sem parâmetros. Parâmetros estes tem o intuito de 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.

#### 2.7.4 - Métodos estáticos e de instância

Métodos de *instância* dependem do estado de uma instância de objeto específica para seus comportamentos. *Métodos estáticos* são aqueles cujos comportamentos não dependem de qualquer estado do objeto.

Documentação Java:

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

### 2.8 – Tipo de dados primitivos

Java é uma linguagem derivada de C. Tipos de dados primitivos são aqueles que não possuem estrutura de objetos e seus possíveis valores são literais, e não uma representação orientada a objetos.

Documentação Java:

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

### 2.9 – Strings

As Strings são bastante importantes. São cadeias de caracteres. Com a String podemos manipular textos.

#### 2.9.1 – Strings

*Strings* são 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

Para concatenar Strings usa-se o sinal de soma (+)

#### 2.9.3 - Encadeando chamadas de métodos

Encadeia-se objetos para que uma modificação sempre retorne à modificação e não a original.

Educador, Strings serão amplamente utilizadas neste curso, como também em qualquer programa. É importantíssimo que você tenha domínio sobre esta classe (String).

Documentação Java:

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

### 2.10 – Operadores

São operadores utilizados para operações aritméticas.

Documentação Java:

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

### 2.11 – Operadores condicionais e instruções de controle

São utilizados para realizar decisões condicionais. Estes tipos de decisões são ramificações que o fluxo do programa sofre quando uma asserção imposta pelo programador é válida.

#### 2.11.1 - Operadores relacionais e condicionais

Operações relacionais e condicionais, são operações que servem para tomadas de decisão frente a uma condição. A decisão pode ser tomada caso a condição retornar verdadeiro (true) e outra decisão pode ser tomada quando a condição retornar falso (false). A condição pode ser se uma relação, por exemplo, algébrica (confusa a frase).

#### 2.11.2 - A instrução if

O if avalia as condições booleanas entre parênteses e executa seu bloco caso forem verdadeiras.

### 2.12 – Escopo da variável

Modificadores de acesso restringem o acesso às informações de um atributo para maior segurança e integridade dos dados.

Documentação Java:

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

#### 2.12.1 - A instrução else

Caso if for falso, será executado o que estiver no bloco do else, ou se simplesmente omitirmos o else, será dado prosseguimento ao restante do código.

Documentação Java:

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

Documentação Java:

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

### 2.13 – Loops

Por último mas não menos importante (loops são importantes até demais), falaremos sobre os loops. Que são estruturas de repetição condicionais.

#### 2.13.1 - O que é um loop?

Quando queremos executar um bloco de instruções tantas vezes quanto uma condição retornar verdadeira, utilizamos um loop.

#### 2.13.2 - Loops while

É uma construção de programação que executa repetidamente certas ações enquanto uma condição não é atendida.

#### 2.13.3 - Ramificação de loop

Para que nosso código execute a mesma ação repetidas vezes, até um limite de vezes estabelecido, utilizamos os loops.

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>

### 2.14 - 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.

Nesta aula eles aprenderam sobre itens que servirão de referência para uma Programação Java, que também serão utilizados em aulas seguintes e que na linguagem Java existe um conjunto de palavras reservadas, e que cada uma terá um propósito específico na linguagem.

### 2.15 – Exercícios

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.

### 2.16 – TDP

#### 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.

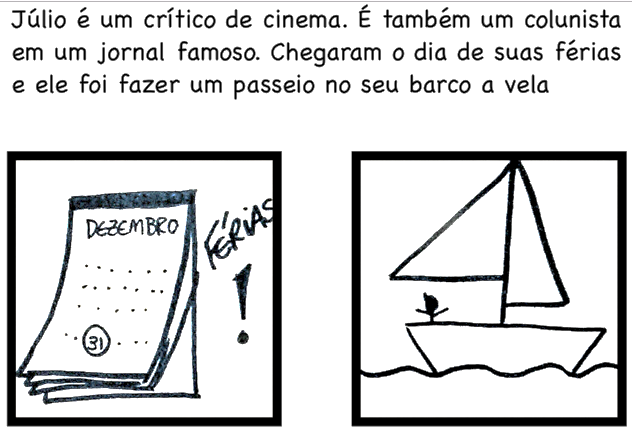
Dado um caso, ou problema, o storyboard deve narrar o mesmo caso, introduzindo o produto como uma solução.

#### Tarefa

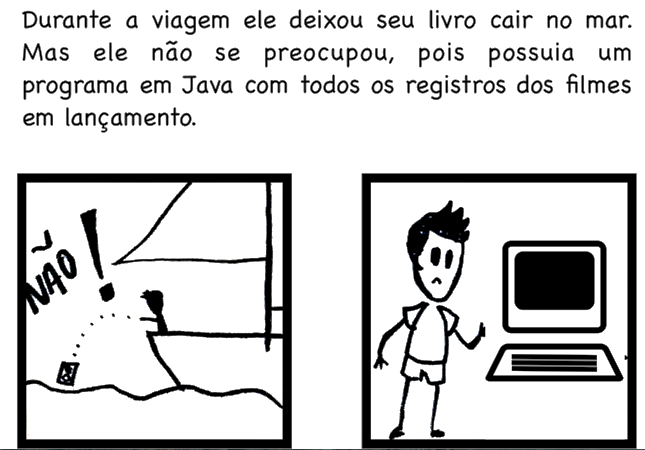
Crie um storyboard que narre o acidente de Júlio, mas que com a introdução do aplicativo, prive-o da necessidade de reescrever sua agenda.

### SUGESTÃO:

A sugestão de storyboard para este exemplo é:







O ideal é que o projeto siga usando este storyboad.

Observação. O Storyboard foi desenhado a mão. Se o educador tiver um quadro negro seria interessante que a história fosse desenhada nele. O desenho não precisa ser bonito, pois o intuito é criar um “rascunho”.

## 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 todas as funcionalidades e regras que o software implementará.

Como esclarecimentos dos requisitos não funcional citados no livro do aluno, abaixo segue uma breve explicação dos mesmos:

* **Performance**: Define o quão rápido e eficiente é a execução do app. Uma app que trava muito, é lento e faz com que haja um consumo de energia muito alto é um app de baixa performance.
* **Usabilidade**: Define o quão fácil e intuitivo é o uso do app. Um app que possui coisas difíceis de serem encontradas, cores que cansam a vista, botões e texto muito pequenos ou exageradamente grandes, ou que o usuário erra muito ao tentar utilizar uma das funcionalidades é um app com usabilidade ruim.
* **Manutenibilidade**: Nem sempre os próprios desenvolvedores do app serão aqueles que farão a manutenção. A Manutenibilidade define o quão fácil e rápido a manutenção do app será.
* **Segurança**: Software lida com dados, e muitas vezes dados importantes que não podem ser perdidos ou alterados deliberadamente, como contas bancárias ou dados de controle de um avião, por exemplo. A segurança define quão seguro serão os dados mantidos pelo software, sendo que seguro, quer dizer integro, coeso, protegido, verídico.

**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 a solução para os problemas. No decorrer do curso as fases de análise e design de software serão abordadas. Este é o diferencial deste curso para os outros cursos. Ele realmente expõe o aluno a situações reais. Os próximos passos serão uma sequência mecânica de etapas que devem ser seguidas com atenção.

### 3.2 – Mão na massa

Educador, sugerimos que este passo-a-passo longo seja feito junto com os alunos. Projete no telão a sua 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, isso agilizará a detecção dos 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, aqui uma boa prática seria obter exemplos de atributos com os alunos e escrevê-los em seu quadro. Mas para que o entendimento dos passos seguintes seja mais simples, 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, temos aqui um código simples, mas o aluno iniciante sempre encontrará erros, e são bem simples como falta de fechamento de bloco ou erros de ortografia. Se atente a isso e reforce tais pontos para que os erros 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.

### 3.3 – 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.

Nesta aula, implementamos um projeto prático onde pudemos simular o funcionamento e as características de um carro e falamos sobre a importância da fase de Análise de Requisitos no início do desenvolvimento de um projeto de software. É a partir daí que inicia a fase de desenvolvimento. Também foi visto sobre como os atributos devem ser aplicados a uma classe e também foi abordado o método toString().

### 3.4 – Exercícios

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";

### 3.5 – TDP

#### 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.

### SUGESTÃO:

Listando 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

**Educador,** para que o material de apoio não fique muito extenso, não iremos criar um cadastro de Cargo, País e Gênero. Estes três casos podem ser facilmente cadastrados automaticamente via código. Pode até ser considerado decisão de projeto, por exemplo. Ao invés de forçar o usuário a cadastrar país por país para que estes possam ser usados, nós (programadores) simplesmente podemos disponibilizar todos os países ao usuário. Para Gêneros e Cargo, podemos selecionar todos os mais utilizados, apenas para nível didático e deixa-los pré-cadastrados. Caso você queira implementar o cadastro destas entidades, sinta-se livre.

## 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.

Nesta 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.

### 4.5 – Exercícios

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

#### 4.6.1 - Identificando métodos e atributos

Você está se saindo muito bem. O seu time de desenvolvimento possui um arquiteto de software. 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)