**Estudo de Java**

**1º Conceito Inicial (Breve História)**

A Sun criou um time (conhecido como Green Team (Equipe Verde)) para desenvolver inovações tecnológicas em 1992. Esse time foi liderado por James Gosling, considerado o pai do Java. O time voltou com a ideia de criar um interpretador (já era uma máquina virtual, veremos o que é isso mais a frente) para pequenos dispositivos, facilitando a reescrita de software para aparelhos eletrônicos, como vídeo cassete, televisão e aparelhos de TV a cabo.

A ideia não deu certo. Tentaram fechar diversos contratos com grandes fabricantes de eletrônicos, como Panasonic, mas não houve êxito devido ao conflito de interesses e custos. Hoje, sabemos que o Java domina o mercado de aplicações para celulares com mais de 2.5 bilhões de dispositivos compatíveis, porém em 1994 ainda era muito cedo para isso.

Com o advento da web, a Sun percebeu que poderia utilizar a ideia criada em 1992 para rodar pequenas aplicações dentro do browser. A semelhança era que na internet havia uma grande quantidade de sistemas operacionais e browsers, e com isso seria grande vantagem poder programar numa única linguagem, independente da plataforma. Foi aí que o Java 1.0. foi lançado: focado em transformar o browser de apenas um cliente magro (thin client ou terminal burro) em uma aplicação que possa também realizar operações avançadas, e não apenas renderizar html.

Atualmente os \*applets realmente não são o foco da Sun. É curioso notar que a tecnologia Java nasceu com um objetivo em mente, foi lançado com outro, mas, no final, decolou mesmo no desenvolvimento de aplicações do lado do servidor. Em 2009 a Oracle comprou a Sun, fortalecendo a marca.

\*applet: **Applet** é um pequeno [software](https://pt.wikipedia.org/wiki/Software) que executa uma atividade específica, dentro (do contexto) de outro [programa](https://pt.wikipedia.org/wiki/Programa_de_computador) maior (como por exemplo um [web browser](https://pt.wikipedia.org/wiki/Web_browser)), geralmente como um [Plugin](https://pt.wikipedia.org/wiki/Plugin" \o "Plugin).

**2º Programação Estruturada com Java**

* Comentário de Bloco
* Comentário de Linha
* Java é Case Sensitive
* Palavras chaves não podem ser usadas como identificadores

**Tipos de Dados**

* **Boolean - Verdadeiro \ Falso**
* **String** – **Cadeia de Caracteres**
* **Inteiro –** **Representação Numérica com tipo Primitivos int, short**
* **Ponto Flutuante - Representação Numérica com tipo Primitivos double, float**

**Nota: Variáveis é um espaço utilizado para armazenar informações as mesma é inicializada com o operador de atribuição (** = **).**

**Estrutura de Controle**

* **Estrutura de Decisão**
* **If**
* **If – else**
* **If – else – if**
* **Switch**
* **Estrutura de Repetição**
* **While**
* **Do – While**
* **For**
* **Estrutura de Interrupção**
* **Break**
* **Continue**
* **Return**

**3º Programação Orientada a Objetos com Java**

* Programação Orientada a Objetos (POO) é uma metodologia de programação que vê um programa como um conjunto de objetos que pode atuar só ou interagir uns com outros
* Todos os objetos de uma classe possuem os mesmas características e o mesmo comportamento
* Um objeto é um programa que possui dados, chamados atributos, e que executa certas ações conhecidas como comportamentos
* Uma classe define um certo tipo de objeto
* Objetos de uma mesma classe possuem os mesmo tipo de dados e os mesmo comportamentos
* Em tempo de execução, objetos podem agir de forma independente ou interagir com outros objetos para alcançar os objetivos do programa
* Princípios da POO
  + Encapsulamento
  + Polimorfismo
  + Associação

É uma arquitetura, ou melhor é um tipo de lógica em PHP, é ter duas classe e em vez de uma herdar da outra elas se associam, logo depois ocorrerá a interação entre objetos.

* + Herança
* Encapsulamento uso de get e set
  + Em outras palavras, colocar coisas em uma cápsula
  + Apenas parte do que está na cápsula é visível
  + O programador descreve a sua porção de código de forma que outros programadores saibam como usá-la, porém omite todos os detalhes de como o código funciona
  + As classe são responsáveis pelo seus atributos
  + Modificadores de acesso fazem parte do encapsulamento são eles
* Public

**Características**

O modificador **public** deixará visível a classe, método e atribut para todas as outras classes.

**Vantagens**

Fácil manipulação na classe ou junto do objeto

**Desvantagens**

Falta de responsabilidade, aceita qualquer modificação

* **Private**

**Características**

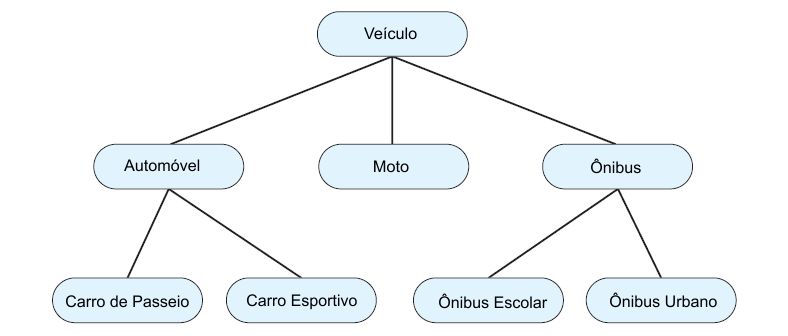
Este é mais recomendado, seus atributos so podem ser acessado pela própria classe, já os seus métodos podem ser acessado pelo objeto e pela classes filhas

**Protected**

**Características**

O modificador protected deixará visível o atributo para todas as outras classes e subclasses que pertencem ao mesmo pacote. A principal diferença é que apenas as classes do mesmo pacote tem acesso ao membro. O pacote da subclasse não tem acesso ao membro. Os atributos e os métodos protegidos vao poder ser acessados apenas pelas classes filhas e nunca pelo objeto

* Polimorfismo
  + Muitas formas
  + A mesma instrução do programa pode significar diferentes coisas em diferentes contextos
  + Significa que um método executa diferentes ações dependo do tipo de objeto a que pertence
  + Em polimorfismo as subclasses possuem os mesmos métodos e características da superclasse, porém nas subclasses os métodos e as características são (implícitos), e (explícitos quando se quer sobre-escrever (overrid) o método da superclasse)
  + (overrid) Mesmo método, mesmo parâmetros , e apenas comportamentos diferentes
  + Overload()mesmo método , com parâmetros e tipos de retorno diferentes
* Herança próxima aula 08/08/2016
  + É uma forma de organizar classes
  + O programador define características e comportamento uma única vez, e então os aplica a toda uma coleção de classes
  + Em herança as subclasses possuem os mesmos métodos e características da superclasse(implicitamente),e este métodos fazem a mesma função na classe teste
* Interface serve par ditar quais os método devem ser implementados por uma classe, esta também trabalha com métodos abstratos
* Classes Abstratas(Polimorfismo)
* Estas classes não podem ser instanciadas, somente as classes filhas podem ser instanciadas, estas também possuem métodos e comportamentos indefinidos, para podermos implementá-los em nossas subclasses
* Em classes abstratas existem métodos sem corpo, ou seja, (métodos abstrato, a serem implementados nas classe filhas), porém os métodos abstratos só podem existir em classes abstrata.
* Herança
* CLASSES FINAL NÃO PODEM SER INSTANCIADA
* OS METODOS FINAL TAMEBM NÃO PODEM SER SOBREESCRITO
* Interface só trabalha com métodos abstratos, porem não é preciso declara no inicio do método (abstract voi algumacoisa...)pois esses método já vem implicitamente na interface



* A classe veículos possui certar propriedades
* As classes automóveis, moto e ônibus herdam estas propriedades e podem adicionar novos atributos
* A herança possibilita que o programador evite a repetição
* A relação entre classe e objeto: São estruturas que definem as características e comportamentos dos elementos do mundo real que se objetiva modelar

Classes e objetos



* Métodos estáticos – podem ser invocados sem que o objeto seja instanciado
* O que é um método? Métodos correspondem a ações realizadas por um determinado objeto, o mesmo é iniciados com letras minúscula e seguido por ()
* Qual a relação existente entre classes e objetos? São estruturas responsáveis pelas características e comportamentos dos objetos do mundo real que se deseja modelar
* Todos os objetos da mesma classe possuem os mesmos métodos? Não
* O que é encapsulamento? No encapsulamento apenas parte do que está na cápsula é visível. Ou seja , o programador define uma porção de código de maneira que outros programadores saibam como usá-los, porém omiti detalhes de como este funciona.
* O que significa esconder complexidade? Corresponde ao encapsulamento.
* O que é polimorfismo?Refere-se as varias formas de comportamento ou assinatura de determinado método, ou seja a mesma instrução pode conter diferentes coisas em diferente contextos.
* O que é herança? Corresponde a forma de organizar classes, ou seja o programador define características e comportamentos uma única vez e aplica a uma coleção de classes.A classe evita a duplicidade de código.
* O que é um algoritmo? Corresponde a declarações de instruções de forma bem definida e organizada.
* O que é um pseudocódigo?
* Quais atributos possuem um objeto que represente uma música? Tom, acorde, letra.....etc.
* *Garbage Collection* 
  + Responsável pela liberação automática do espaço em memória
* Classe é a definição de um conjunto de objetos
* Todo programa em Java está dentro de uma estrutura lógica de código chamada classe
* As classes são compostas de atributos, métodos e construtores

Comentário ajudam a entender o código fonte: podem ser de três tipos.

1º de uma única linha com duas barras: / odenilson marques/

2º de multiplas linhas com duas barras e dois asteriscos:/\*odenilson marques

araújo\*/

3º de multilas linhas para desenvolver documentos de comentários com duas barras e três asterisco/\*\* odenilsonmarques...\*/

* Em Java, ***new*** é um operador unário usado para criar objetos de uma classe
* O operador ***new*** retorna o endereço de memória do objeto
* Existem dois tipos de métodos
  + Métodos que retornam um único item
  + Métodos que executam alguma ação e não retornam um item (métodos ***void***)

Variáveis ​​locais

* Dizemos que a variável é local porque seu significado é local
* Parâmetro
* Um parâmetro formal na definição de um método representa o argumento real
* Java passa argumentos para um método usando chamada por valor

Comentários de pré-condição e de pós-condição

* Uma pré-condição determina os requisitos de um método
* Uma pós-condição indica efeito de um método

Modificadores *public* e *private*

* Qualquer classe pode usar uma classe, método ou variável de instância pública
* O modificador público, quando aplicado a uma variável de classe, método ou instância, significa que qualquer outra classe pode usar diretamente ou acessar a classe, método ou variável de instância pelo nome

Metódos modificadores e de Acesso

Um método público que retorna dados de uma variável de instância *private* é chamado um método de acesso, método *get*

* Um método público que muda os dados armazenados em uma ou mais variáveis ​​de instância *private* é chamado um método modificador, método *set*, ou *setter*
* Construtores criam e inicializam novos objetos
* Um construtor é um método especial que é chamado quando você usa o operador *new* para criar um novo objeto
* Pareiem variaveis e metodos estaticos

Variáveis e Métodos Estáticos

* Variáveis e métodos estáticos pertencem a classe como um todo e não a um objeto individual
* Variáveis estáticas são compartilhadas por todos os objetos de uma classe
* Variáveis estáticas são também chamadas de variáveis de classe
* Métodos estáticos continuam membros da classe mas o método pode ser invocado sem que seja utilizado um objeto
* Sobrecarga: A sobrecarga de um nome de método significa dar o mesmo nome a mais de um método dentro de uma classe, porém este possuem assinaturas diferentes.
* Enumerações: Uma enumeração lista os valores que uma variável pode ter
* ***enum*** Avaliacao {A, B, C}
* Uma enumeração age como um tipo classe
* Os itens listados entre as chaves são objetos que pode ser atribuídos ao objeto do tipo de classe
* Uma enumeração é normalmente definida dentro de uma classe, porém sempre fora das definições dos métodos
* *Arrays: Arrays são tipos especiais de objetos usados para armazenar uma coleção de dados do mesmo tipo*
* *Todos os dados armazenados em um array devem ser do mesmo tipo*
* Um indice é um inteiro que indica um elemento do array
* Indices de arrays iniciam em 0

Parei em herança

Collection: coleções Java são estrutura de dados, interface e métodos para manipular esse dados ex:cartas de baralhos,musicas favoritas ...ect

Esta coleção possui as seguintes interfaces: set(sem duplicidae),list(ordenado),queue(filas),map(chaves-valor)

As classe que implementam a interface list são: vector, ArrayList, linkedLIst

As classe que implementam a interface set são:

SPEEDTEST.NET e velocimetro(SITEs PARA MEDIR A VELOCIDADE DA INTERNERT)

Coesão em Java

Refatoração: alterar a estrutura do código mais não o sentido deste

O método construtor de uma classe serve para executar algum comportamento(atribuição de valor, execução de método, etc) logo no momento em que uma instancia da mesma for criada. Isso traz vantagens pois evita a execução de tarefas repetitivas e obrigatórias