**Introdução à Linguagem Java**

O que é java?

* Linguagem de programação (Regras sintáticas)
* Plataforma de desenvolvimento e execução
* **Bibliotecas (API):** Fornece um conjunto vasto de funcionalidade que nós podemos utilizar nos nossos aplicativos.
* **Ambientes de execução:** Consiste que quando você vai executar um programa em java é preciso instalar no dispositivo (computador) o ambiente de execução, permite que os programas executem, também tem que instalar o quite de desenvolvimento java com esse ambiente com essa API para que ele possa construir as aplicações em JAVA.

Histórico

Problemas resolvidos e motivo de seu sucesso:

* **Ponteiros/ gerenciamento de memória:** Des de que a programação começou a gente tinha muitos problemas, no final da década de 1980 e início da década de 1990, com relação a programação de ponteiros gerenciamento de memória. Muitas linguagens amplamente utilizadas como a linguagem C, você tinha que definir explicitamente quais variáveis eram ponteiros e quais variáveis eram valor e você tinha que controlar isso manualmente, esse tipo de controle manual de memória que tinha nas linguagens como C e C++ faziam que essas linguagens fossem de um nível mais baixo. A gente chama de linguagem de baixo nível e isso gerava uma dificuldade para o programador e os softwares eram mais suscetíveis a terem erros, bugs. Então o mercado de programação (Comunidades) estava precisando de linguagens que eliminasse essa dificuldade.
* **Portabilidade falha:** reescrever parte do código ao mudar de SO: Embora uma linguagem como C, tivesse um padrão e ela tivesse compiladores para várias plataformas, quando o programador tivesse que compilar esse código ou plataformas diferentes, eventualmente ele tinha que reescrever parte do código e alterar algumas coisas para adaptar aquele programa para sistema operacional, isso também sempre foi uma dificuldade de linguagens compiladas.
* **Utilização em dispositivos diversos:** Muitas vezes era desejado rodar o mesmo programa em vários dispositivos, no computador, dispositivo embarcado...Essa dificuldade de portabilidade aumentavam os custos e a dificuldade para se fazer isso.
* **Custo**
* Criada pela Sun Microsystems no meio da década de 1990: No meio da década de 1990 foi lançada a linguagem java que foi uma revolução no mercado corporativo principalmente porque é uma linguagem totalmente orientada a objetos e é uma linguagem de mais alto nível, ela resolvia os problemas em relação a linguagem de baixo nível.
* Adquirida pela Oracle Corporation em 2010

Aspectos notáveis:

* Código compilado para Bytecode e executado em máquina virtual (JVM):
* Portável, segura, robusta
* Roda em vários tipos de dispositivos
* Domina o mercado corporativo desde o fim do século 20
* Padrão Android por muitos anos

Plataforma Java SE

* JVM – java virtual Machine
* Máquina virtual do java – necessário para executar sistemas java

Compilação e Interpretação

* Linguagens **compilada:** C, C++

Compilador é aquele programa, aquele utilitário que transforma o código fonte que o programador escreveu em um código compilado que pode ser executado pelo computador.

* Linguagens **interpretada:** PHP, JavaScript

Interpretador é um utilitário que pega o código fonte e ele vai gradualmente interpretando aquele código e transformando ele para código executável.

* Linguagens **pré-compiladas + máquina virtual:** JAVA, C#

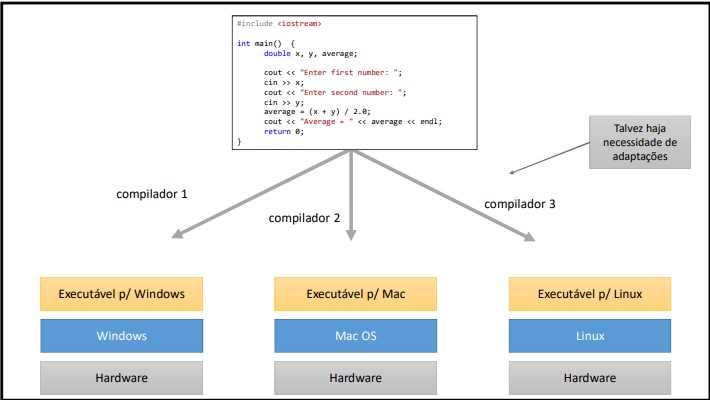
Abordagem hibrida são pré-compiladas e depois o código gerado pela pré-compilação ele é executado por uma máquina virtual.

Linguagem Compilada: O código fonte que é escrito pelo programador tem que passar pelo compilador, esse compilador junto com outras ferramentas que geram o arquivo executável, vai gerar um arquivo específico para a plataforma que você estiver usando. Exemplo: Você tem um computador que é um Hardware, nesse computador foi instalado o Windows que é o sistema operacional em cima desse sistema vai ter que rodar um programa executável específico para Windows, o que isso significa? Significa que se eu quiser pegar esse mesmo programa que foi escrito no código fonte em C ou C++ e quiser executar esse programa na plataforma MAC OS eu vou ter que ter outro compilador para traduzir esse código para executável para MAC e aí sim esse executável vai rodar em cima do MAC OS, o mesmo raciocínio vai para o LINUX e para outra plataforma qualquer.

**Problema:** portabilidade, esse código não necessariamente via funcionar certinho para todas as plataformas, bastando apenas trocar o compilador eventualmente vai ter que fazer alguma alteração nesse código.

**Vantagem:** as linguagens compiladas são muito rápidas, porque o código gerado é específico e roda diretamente em cima do sistema operacional, por outro lado tem o problema da portabilidade, porque tem o trabalho um débito técnico para resolver mudar de uma plataforma para outra. A linguagem compilada é menos suscetível a bugs, porque o compilador ajuda a verificar o código, os tipos e isso torna o sistema menos a erros depois que for rodar no cliente.

Essa figura mostra o senário da linguagem compilada:

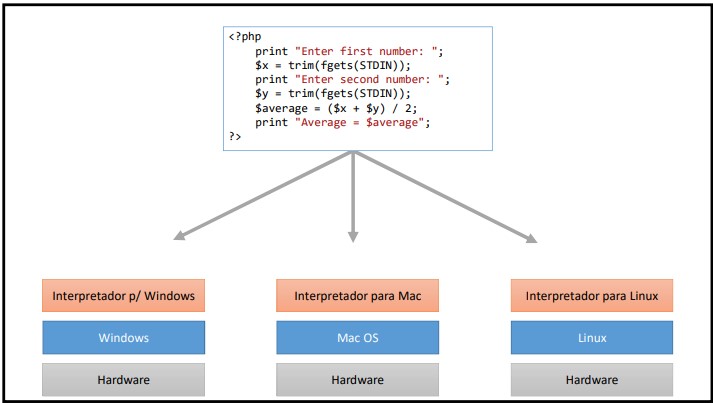


**As linguagens interpretadas** funcionam de uma forma diferente, ele distribui o interpretador para cada uma das plataformas de sistema operacional, então existe um interpretador para Windows, MAC OS, LINUX e dentre outras. Isso permite que o mesmo código sem nenhuma alteração execute em sistemas operacionais diferentes isso é muito bacana, porém como a análise do código e a geração do executável, ela é feita **just-in-time** ou seja é feita no momento que você está executando, esses programas tendem a ser mais lentos.

**Vantagem:** portabilidade e manutenção.

**Problema:** As linguagens interpretadas são mais difíceis de programar, porque ela é mais relaxada, tipagem mais fraca e isso pode ser uma brecha para bugs e problemas se não for feita de forma correta.

Essa figura mostra o senário da linguagem interpretada:



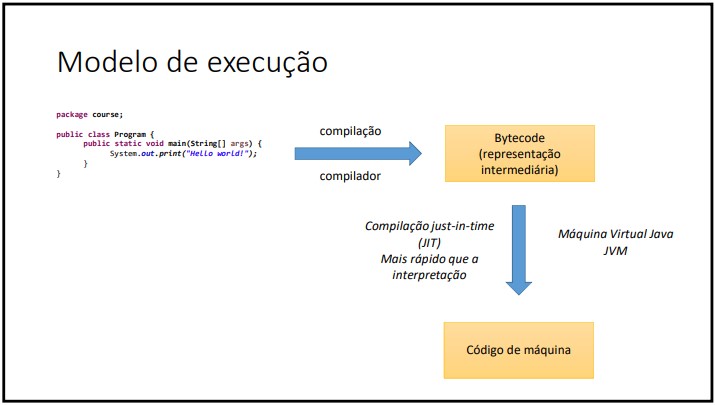
**Linguagem com abordagem hibrida**: O programa vai passar por um pré-compilador e vai gerar um BYTECODE que é uma representação intermediaria do código já compilada, garantida que esta correta do ponto de vista sintático, depois que o código foi verificado ele vai rodar em cima de uma máquina virtual (JVM). Se o sistema é Windows vou ter uma máquina virtual para Windows, se for MAC OS vou ter um JVM para MAC OS, se for para LINUX vou ter um JVM para LINUX...Essa abordagem hibrida conseguiu obter vantagens nos dois mundos, tem a compilação, verificação de tipos e tem a portabilidade muito facilitada, porque o mesmo BYTECODE que é gerado na compilação ele é executado sem nenhuma alteração em qualquer uma das máquinas virtuais (Sistema Operacional).

Essa figura mostra o senário da linguagem hibrida:

Diagrama

Descrição gerada automaticamente

**Modelo de Execução do JAVA:** é uma abordagem hibrida entre a compilação e a interpretação, o código fonte é compilado para BYTECODE que é uma representação intermediaria, porém já está verificada, depois o BYTECODE por sua vez vai ser executada pela máquina virtual Java especifica para o sistema operacional. Esse processo de executar o BYTECODE é chamado de **compilação just-in-time** é mais rápida que a interpretação, porque o BYTECODE já está otimizado, garantindo que está correto sintaticamente.



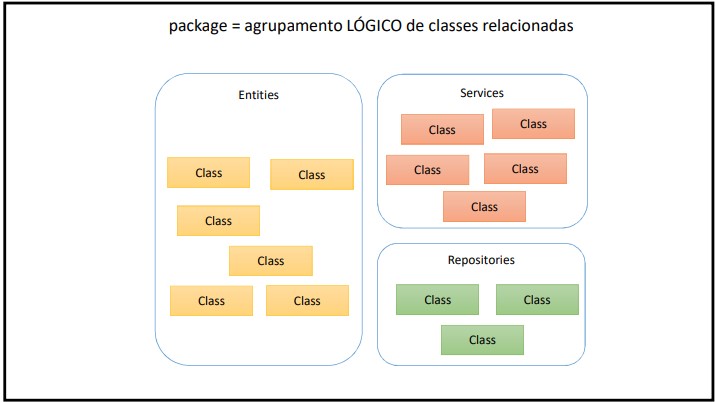
Estrutura de uma aplicação java

Java é uma linguagem orientada a objetos e a unidade lógica básica de um programa orientada a objetos é a classe. Todo código em java tem que estar dentro de classes.

Gráfico de barras

Descrição gerada automaticamente com confiança baixa

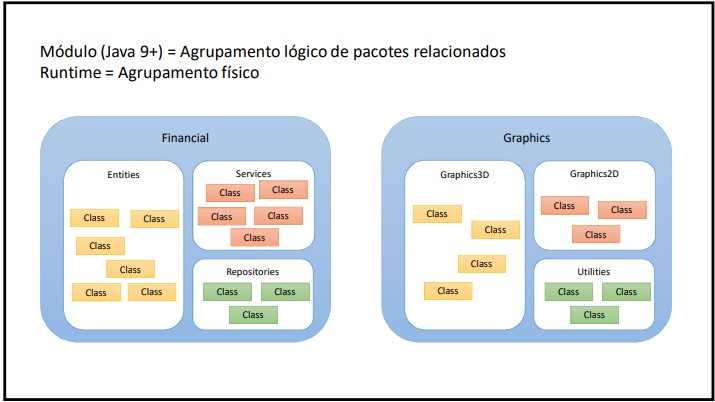
Package é um agrupamento LÓGICO de classes relacionadas. Na minha aplicação posso ter vários tipos de classes, exemplo: classes que representam entidades do negócio, como produto, cliente e pedido. Essas classes que são entidades, podem ficar em um pacote chamado ENTITIES, para agrupar e organizar a aplicação. Da mesma forma posso agrupar as classes que representam serviços chamado SERVICES em outro pacote. Classes de serviços seriam de e-mail, pedidos e login...As classes chamadas de REPOSITORIES, são para acessar os dados.



O **módulo** foi introduzido a partir do java 9, ele é um agrupamento lógico de pacotes relacionados, podendo agrupar pacotes e classes no conceito chamado módulo.

Nesse exemplo na figura abaixo, tem um modulo de financeiro que pode ter entidades, serviços e repositórios. Tudo relacionado a financeiro pode ser agrupado em um modulo. No outro exemplo é o modulo de gráficos, vamos supor que você tenha uma biblioteca gráfica com gráficos 3D, 2D e utilitários podendo agrupar em um modulo para organizar sua aplicação.

Quando a gente fala que o agrupamento é lógico, isso significa que o agrupamento está no nível conceitual e um nível de código, esses agrupamentos lógicos podem ser compilados e gerado o Bild (Runtime) que seria o agrupamento físico que estaria representado por arquivos. Então cada Bild ou Runtime é uma unidade física que pode ser instaladas par rodar nos dispositivos.



**Aplicação** é um agrupamento de módulos relacionados. Exemplo: Posso ter um sistema de comercio eletrônico que ele é composto por vários módulos.

