JAVA – Aula 2

Existem 2 momentos no desenvolvimento do JAVA

1- Programadores JAVA – **JDK (JAVA Development Kit – Kit de Desenvolvimento JAVA)**

2- Pessoas que usam JAVA – **JRE (JAVA Runtime Environment – Ambiente de Execução JAVA)**

JRE

- É composto por 2 partes:

1- JVM – é a máquina virtual JAVA, fazendo com que o código em Bytecode seja enviado para o computador

Possui algumas partes internas:

Loader/verificador – **Loader** é a parte interna da JVM que vai carregar o Bytecode na memória da máquina virtual, já o **verificador** vai verificar se esse código pode ser executado sem problema algum

Interpretador/gerenciador – **Interpretador** é aquele que vai pegar o código em Bytecode e vai transformar diretamente para o código nativo da máquina em questão. Exemplo: se estou na JVM para Windows, ele vai converter o código bytecode numa instrução que o Windows irá compreender. O **gerenciador** de memória vai tratar como os códigos e as variáveis vão ser gerenciadas na memória da JVM

Compilador JIT (Just in Time – Em Tempo Real) – Ganho de performance (antigamente era muito lento o processo)

2- Bibliotecas – são APIs que poderão ser executados dentro do programa para tornar mais atrativo e com mais funcionalidade

JDK

- Dentro do JDK, já temos o JRE (ou seja, já vem com JVM e todos seus componentes e com as bibliotecas)

- Se você instala o JDK já vem o JRE

- Vem com a linguagem JAVA **(JavaLeng)** e com um conjunto de ferramentas **(JavaTools)**

Dentro do JavaTools temos várias ferramentas, como:

JavaC (Java Compiler) – é o compilador JAVA, que transforma o código fonte em bytecode

Debugger – permite que você verifique como o seu programa está sendo executado em tempo real, inclusive verificando e testando conteúdos de variáveis ou acesso a banco de dados

APIs – são APIs de desenvolvimento

OBS: na hora de instalar o JDK, também podemos instalar o **IDE**, que é um ambiente de desenvolvimento que facilita na hora de programar em JAVA (plataforma de exemplo: NetBeans)

Lembrando da AULA 1:

Compilador – transforma o **código fonte** em **código objeto**

Linker – transforma o **código objeto** em **código executável**

Interpretador – faz a tradução direta do **código fonte** para o **código executável**

OBS: o **compilador** passa por 2 processos, porém ele analisa tudo antes, vai interpretar todos os comandos antes, vai verificar se existe alguma inconsistência e depois vai traduzir isso tudo para o linker gerar o código executável. Esse código gerado executável vai consumir mais memória, pois o código todo vai estar na memória do computado, já o **interpretador** vai pegando cada uma das linhas, jogando na memória e executando ela. Depois de executada, ela é apagada da memória e colocado uma outra instrução, assim ocupando **menos memória**. Porém o **compilador** já compilou, já traduziu, já gerou o código executável no final, então esse código leva menos tempo para executar, enquanto o **interpretador** vai ter que traduzir cada uma das linhas, ele leva mais tempo para isso

Ou seja:

Código compilado – ocupa mais memória, porém leva menos tempo

Código interpretado – ocupa menos memória, porém leva mais tempo