1. A Plataforma Java

Vamos fazer uma introdução formal ao Java, que pode ser definido não apenas como uma linguagem de programação, mas como uma plataforma completa de desenvolvimento de software. Nesta seção, abordaremos um pouco da sua história, como funciona, e faremos a instalação do ambiente de desenvolvimento integrado (IDE), no qual criaremos nossos primeiros aplicativos.

1.1 Histórico e Evolução da Linguagem Java

1.1.1 A História da Linguagem de Programação Java

Para entender o surgimento da linguagem Java, é importante compreender a diferença entre dois paradigmas de programação: o estruturado e o orientado a objetos.

No paradigma **estruturado**, os programas são compostos por funções que operam sobre dados armazenados em variáveis. Não existe uma ligação direta entre os dados e as funções; essa relação é responsabilidade do programador, que deve passar os dados como parâmetros.

Por outro lado, no paradigma **orientado a objetos (OO)**, há uma relação mais próxima entre dados e funcionalidades. Os dados são agrupados em estruturas chamadas **objetos**, que também contêm as funções (ou métodos) que operam sobre eles. Isso aproxima a programação da forma como enxergamos o mundo real, onde objetos têm características e comportamentos.

A programação orientada a objetos oferece maior **produtividade** e **reutilização de código**. Linguagens como **SmallTalk** e **C++** foram pioneiras nessa abordagem. O C++ se destacou por sua compatibilidade com a linguagem C, mas apresentava desvantagens, como a sintaxe complexa e a necessidade de gerenciamento manual de memória.

Segundo *Deitel & Deitel (2010)*, em junho de 1991, James Gosling, Mike Sheridan e Patrick Naughton — engenheiros da Sun Microsystems (posteriormente adquirida pela Oracle) — iniciaram o desenvolvimento de uma nova linguagem de programação orientada a objetos. Seu objetivo era criar algo mais simples do que o C++, com gerenciamento automático de memória e capacidade de rodar em qualquer tipo de computador. Inspirados pelo café que consumiam, batizaram a linguagem de **Java**, em homenagem à ilha de onde vinha sua marca preferida da bebida.

A linguagem foi lançada oficialmente em **1995**, destacando-se por sua **simplicidade**, **portabilidade** e recursos para aplicações em rede e internet, o que era extremamente relevante com a expansão da web na época.

James Gosling, considerado o "pai do Java", definiu os cinco princípios fundamentais da linguagem no documento *The Java Language Environment* (GOSLING; MCGILTON, 1997):

1. Java deve ser uma linguagem simples, orientada a objetos e com conceitos familiares.

2. Java deve ser robusta e segura.

3. Java deve ser portável e neutra em relação à arquitetura.

4. Java deve ter alto desempenho.

5. Java deve ser interpretada, multithread e dinâmica.

Mais de duas décadas depois, esses princípios continuam sendo a base da linguagem.

A Sun Microsystems (e, depois, a Oracle) sempre disponibilizou o **JDK** gratuitamente na internet, o que colaborou para a popularização do Java, principalmente entre a comunidade **open source**. Em 2006, a Sun começou a liberar o código-fonte do Java sob a licença GPL. Atualmente, além de uma linguagem, o Java é uma plataforma completa de desenvolvimento, com uma comunidade ativa, mantida por meio do **Java Community Process (JCP)**.

Até a mascote oficial da linguagem, o **Duke**, está liberada sob termos de software livre, podendo ser usada livremente.

1.1.2 Principais Conceitos da Plataforma Java

Como mencionado, o Java é mais do que apenas uma linguagem de programação: trata-se de uma **plataforma de desenvolvimento**. Isso significa que, junto da linguagem, são oferecidos bibliotecas, ferramentas e recursos para criar aplicações para diversos dispositivos — desde computadores e servidores até smartphones e smart TVs.

A plataforma Java foi projetada desde o início para ser **multiplataforma**, ou seja, funcionar em diferentes tipos de dispositivos, sistemas operacionais e arquiteturas.

Mas como isso é possível?

Normalmente, um código-fonte é **compilado** para linguagem de máquina específica de um sistema, tornando-se dependente daquela plataforma. O Java adota uma abordagem diferente: ele compila o código-fonte para um formato intermediário chamado **bytecode**, que não é diretamente executado pelo sistema operacional, mas sim por uma **máquina virtual**: a **JVM (Java Virtual Machine)**.

Processo de Execução

1. O programador escreve o código em um arquivo `.java`.

2. O compilador `javac`, incluído no **JDK**, converte esse código para bytecode, gerando um arquivo `.class`.

3. Esse bytecode é interpretado pela **JVM**, que o traduz para a linguagem de máquina do sistema em que o programa será executado.

Esse processo garante a **portabilidade**: o mesmo programa Java pode ser executado em qualquer dispositivo que tenha uma JVM apropriada instalada.

JDK e JRE

- O **JDK (Java Development Kit)** inclui o compilador `javac`, a JVM e diversas ferramentas e bibliotecas necessárias para o desenvolvimento.

- O **JRE (Java Runtime Environment)** contém apenas o necessário para executar aplicativos Java (como a JVM), sendo suficiente para usuários finais que não vão desenvolver aplicações.

Cada sistema operacional precisa de uma JVM própria. Por isso, você deve instalar a versão adequada do Java para o seu sistema (Windows, Linux, macOS etc.).

1.2. IDE Eclipse

1.2.1. Ambiente de Desenvolvimento Java

O processo de compilação e execução de um programa em Java envolve dois utilitários instalados com o JDK: o compilador `javac` e o interpretador `java`.

Contudo, à medida que os programas Java crescem em tamanho e complexidade, trabalhar apenas com a linha de comando torna-se pouco produtivo. O ciclo de desenvolvimento passa a consumir tempo excessivo com tarefas repetitivas. Para otimizar esse processo, surgiram os IDEs (*Integrated Development Environments* ou *Ambientes de Desenvolvimento Integrado*).

Essas ferramentas integram diversos recursos que auxiliam o desenvolvedor durante a criação de um software, como:

- Editor de código com destaque de sintaxe e autocompletar para comandos e identificadores;

- Interface que facilita a compilação, bastando um clique para compilar todos os arquivos-fonte do projeto;

- Execução automática do programa (caso compilado com sucesso) ou exibição de erros de compilação com indicação precisa dos trechos de código que **precisam de**

correção;

- Recursos visuais de depuração (debug), permitindo a execução linha a linha do programa e a inspeção de variáveis;

- Acesso integrado à documentação da linguagem.

Devido à popularidade da linguagem Java, há diversos IDEs disponíveis no mercado, cada um com suas vantagens e desvantagens. Os mais populares entre os desenvolvedores são o Eclipse (gratuito e mantido por uma comunidade global) e o NetBeans (também gratuito, mas mantido pela Oracle). Ambos oferecem recursos avançados de edição, compilação e testes, sendo considerados equivalentes em funcionalidades.

Neste curso, utilizaremos o Eclipse, o IDE mais utilizado por desenvolvedores Java, tanto em ambientes acadêmicos quanto corporativos.