Disciplina: LPOO
Prof. Anderson V. de Araujo

Aula 01: Introdução ao Java

andvicoso@facom.ufms.br
http://prof.facom.ufms.br/~andvicoso/

Histórico

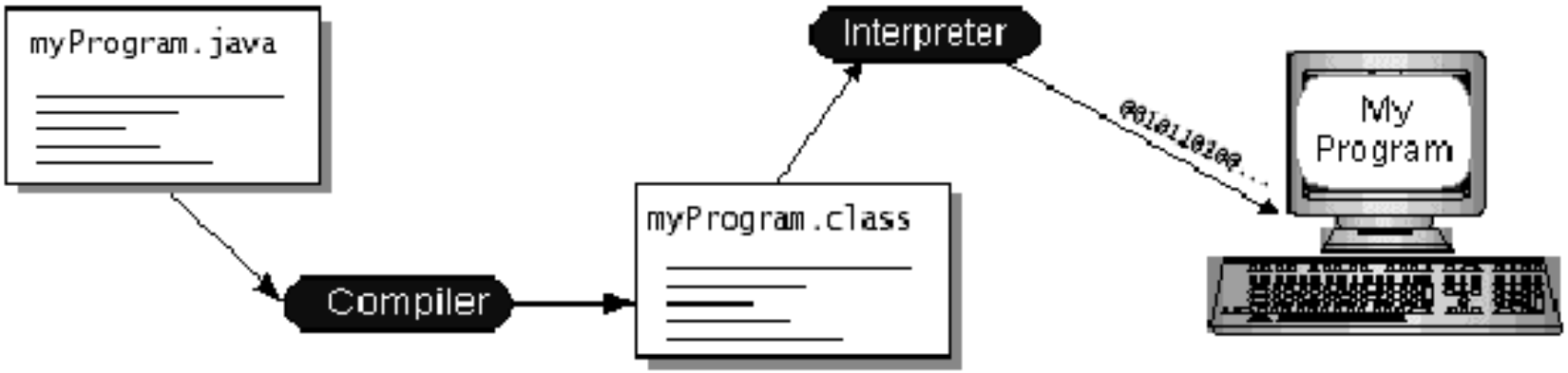
- 1991 – SUN MicroSystems (Projeto Green)
- 1993 – Com a web ganhou popularidade e passou então a potencializar o conteúdo dinâmico
- Anúncio oficial em 1995
  - 2015 fez 20 anos
- 2009 - Sun comprada pela Oracle

James Gosling, conhecido como o “pai” da linguagem Java

Características

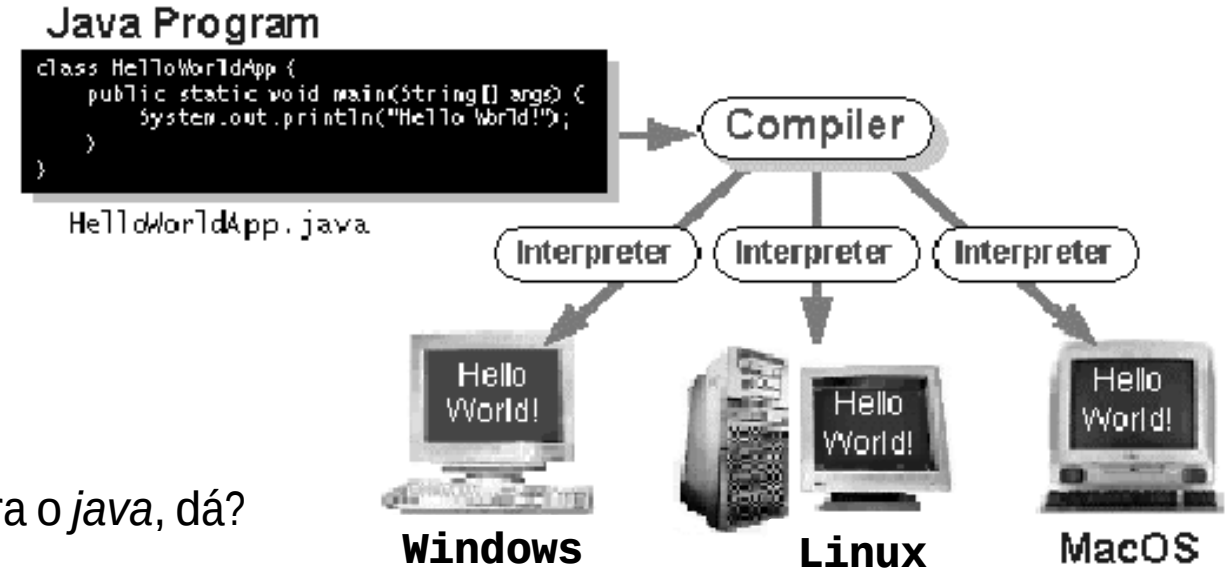
- Portabilidade
- Robustez
- Segurança
- Orientada a Objetos
- Dinâmica
- Alto desempenho

Linguagem Híbrida (Compilada e Interpretada)



Funcionamento

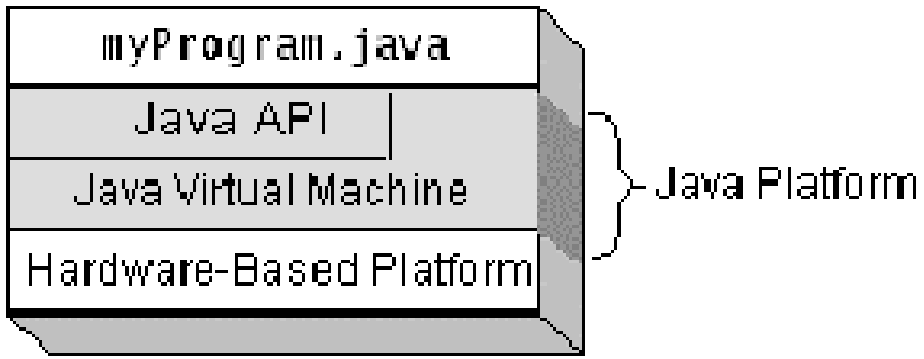
- A compilação gera o ***bytecode*** (independente de plataforma)
  - *Bytecode*: Instruções de máquina para a *Java Virtual Machine* (JVM)
  - "***W****rite* ***o****nce*, ***r****un* ***a****nywhere*" (WORA)



E voltar do *class* para o *java*, dá?

Plataforma Java

- A plataforma Java tem dois componentes:
  - ***J**ava **V**irtual **M**achine* (Java VM ou JVM)
  - *Java **A**pplication **P**rogramming **I**nterface* (Java API)



Mais Java

- **JRE** (***J**ava **R**untime **E**nvironment*): componentes necessários para execução de código Java (JVM + bibliotecas)
- **JDK** (***J**ava **D**evelopment **K**it*): componentes necessários para execução e compilação de código Java (JVM + bibliotecas + compilador)
- Outros: Java SE, Java Embedded, Java EE, Java ME, JavaFX, Java Card, Java TV e Java DB

http://www.oracle.com/technetwork/java

Ambiente Java Típico

- Fase 1 – Editor
  - O programa é criado no editor e armazenado em disco (.java)
- Fase 2 – Compilador
  - Compilador cria o arquivo *bytecode* (.class)
- Fase 3 – Carregador de Classe
  - Carrega o arquivo *bytecode* (.class) na memória
- Fase 4 – Verificador de *Bytecode*
  - Confirma se os dados do arquivo são válidos
- Fase 5 – Interpretador
  - A JVM lê e traduz para uma linguagem que o computador pode entender

Características da Linguagem Java

- Letras minúsculas e maiúsculas são diferentes
- Não há distinção de onde começar a digitar o programa
  - É possível adicionar espaços em pontos estratégicos para facilitar a leitura do programa (recuo ou identação)
- A maioria das instruções devem terminar com ponto e vírgula (**;**)
- Chaves são utilizadas para definir códigos internos (**{ }**)
- Comentários:
  - // comentário em uma linha
  - /* comentário em

múltiplas linhas*/

Palavras Reservadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

*	not used
**	added in 1.2
***	added in 1.4
****	added in 5.0

Praticando

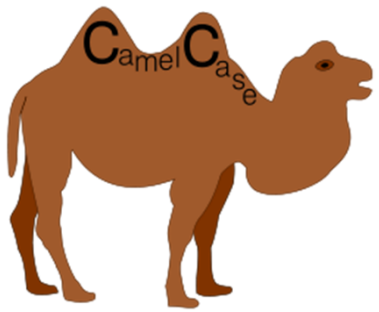
Exemplo

```
public class HelloWorld{

     public static void main(String[] args){

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

     }

}
```

Explicando...

```
public class HelloWorld
```

- Indica que estamos construindo um programa chamado **HelloWorld**
  - Com isso, o seu arquivo **DEVE** se chamar **HelloWorld.java**
  - Por padrão os nomes dos programas e dos arquivos são definidos em **CamelCase**
- **Por enquanto,** vamos criar todos os programas com o public class
  - Mais tarde vamos saber para que serve...



Explicando... (2)

```
public static void main(String[] args){
```

- Informa ao compilador onde o programa se inicia
  - Na verdade é a declaração de um **método**, mas vamos ver isso com calma depois...
- Indica que **main**, é o início do programa
- Palavra **main** deve ser sempre precedida pelas palavras **public**, **static** e **void**
- Palavra **main** deve sempre ser sucedida por (String[] args)
  - A palavra args pode ser diferente
- Essa linha só pode ser escrita uma única vez no programa

Explicando... (3)

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

- Chama a função **`println`** do objeto **`out`** que está dentro do objeto **`System`**
- Ao executar o método aparece a mensagem dentro das aspas duplas no console

Java x Bytecode

JAVA

```
public class HelloWorld {

        public static void main(String[] args) {
                System.out.println("Hello World");
        }
}
```

BYTECODE

```
Compiled from "HelloWorld.java"
public class HelloWorld {
  public HelloWorld();
    Code:
       0: aload_0
       1: invokespecial #8 // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #16 // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #22 // String Hello World
       5: invokevirtual #24 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

Executando (Linha de Comando)

1. Compilar

```
javac HelloWorld.java
```

(JVM vai gerar um arquivo .class com o *bytecode* através do seu compilador)

2. Executar

```
java HelloWorld
```

(JVM vai executar o código *bytecode* através do seu interpretador)

Exercício Prático

- Implementar no notepad (ou notepad++)
- Gerar o arquivo .*class* (compilar)
  - `javac nome_do_arquivo.java`
- Executar (interpretar)
  - `java nome_do_programa`
    - O nome do programa é o nome do arquivo por enquanto, depois vamos ver mais detalhes

1. Desenvolver código que imprima:
   - Idade, nome, peso, data de aniversário, número que calça e se é casado ou não
     - Cada um em uma linha