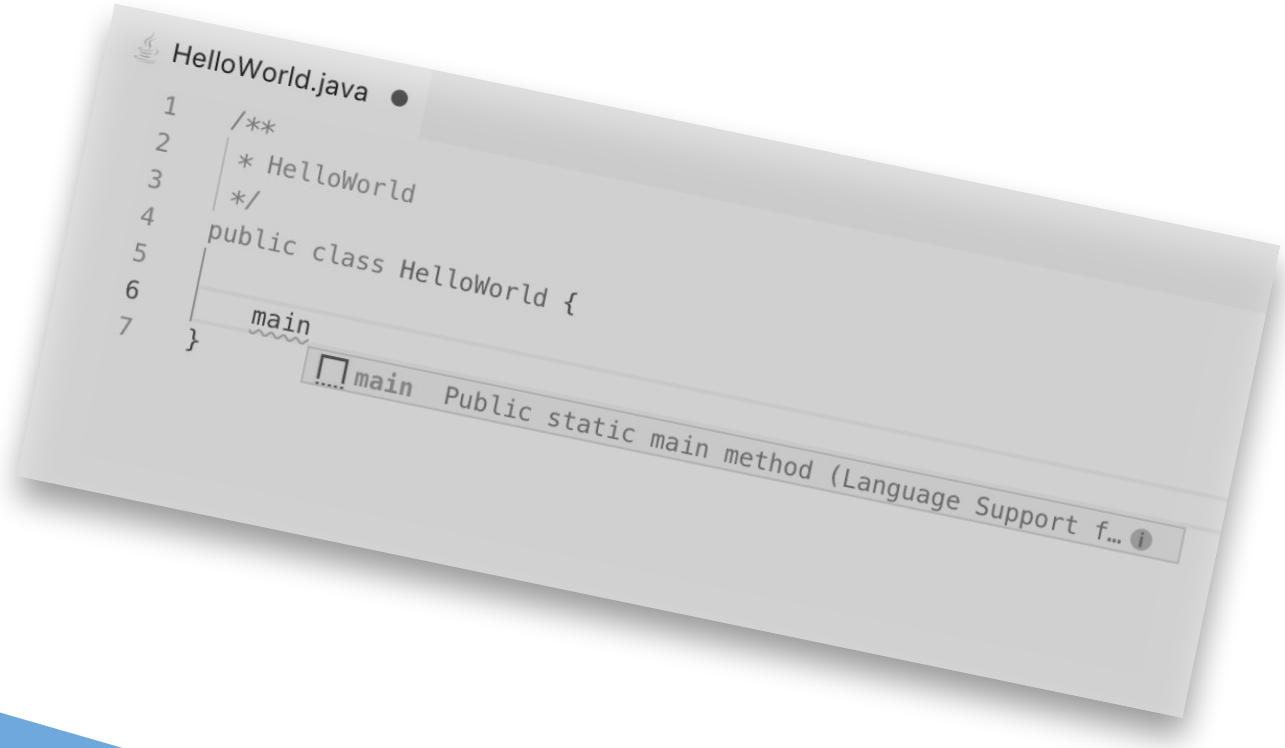

Aula 2

Conceitos Básicos Sobre Programas em Java

Código para imprimir uma linha de texto em Java

```
1   //Programa para imprimir uma mensagem no terminal.
2
3   public class ImprimirMensagem {
4
5       //Método principal - inicia a execução de um aplicatio Java
6       public static void main(String[] args){
7           System.out.println("Imprimindo a primeira mensagem na tela em java.");
8       }
9       //Fim do método principal
10
11  }
12  // Fim da classe ImprimirMensagem
13
```

Comentando Programas

- Para comentar um única linha em java inicie a linha com //
- Para comentar várias linha, inicie a primeira linha com "/*" e finalize a última linha com "*/"

Exemplo de um comentário em múltiplas linhas em JAVA

```
/* Este é um comentário
que ultrapassa uma linha */
```

Declarando uma Classe

- **Todo programa Java consiste em pelo menos uma classe**
- A **palavra-chave** `class` introduz uma declaração de classe e é imediatamente seguida pelo **nome da classe**
- **Por convenção, os nomes de classes iniciam com uma letra maiúscula e apresentam a letra inicial de cada palavras que eles incluem em maiúscula**
- Exemplos de nomes de classes
  - Teste1
  - ManipuladorArquivos
  - ClassificadorInstancias

Declarando uma Classe

- **O nome de uma classe é um identificador e consiste em uma série de caracteres que consistem em:**
  - Letras
  - Dígitos
  - Sublinhados (_)
  - Sinais de cifrão ($)

- Identificadores de classes **não devem começar com dígitos** e não **devem conter espaços**

Declarando uma Classe

- **Exemplos de identificadores de classe válidos:**
  - `Bemvindo1`
  - `$Valor`
  - `_Valor`
  - `m_CampoDeEntrada1`

- **Exemplos de identificadores de classe inválidos:**
  - `7button`
  - `Mãe do Céu`

Declarando uma Classe

- **OBSERVAÇÃO 1:** normalmente, um identificador que não inicia com uma letra maiúscula não é um nome de classe

- **OBSERVAÇÃO 2:** o Java faz distinção entre letras maiúsculas e minúsculas $\rightarrow$ `a1` e `A1` são diferentes

Declarando uma Classe

- **POR ENQUANTO** vamos assumir que cada classe inicia com a palavra-chave **public**
- O nome do arquivo `.java` corresponde ao nome do arquivo da classe principal contida no arquivo
  - A classe `Welcome` deve ser declarada em um arquivo `Welcome.java`
  - A classe `ImprimirMensagem` deve ser declarada em um arquivo `ImprimirMensagem.java`
- Uma chave esquerda "{" inicia o corpo da declaração da classe
- Uma chave direita "}" (correspondente à chave esquerda de declaração da classe) finaliza a declaração da classe

Declarando um Método

- Declarações de classe em Java normalmente contém um ou mais métodos
- No exemplo de código para imprimir uma linha, temos os método `main`

Declaração do método `main` em um classe

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

- Tipo de acesso: `public`
- Tipo de carregamento: `static`
- Tipo de retorno: `void`
- Nome do método: `main`

Declarando um Método

Declaração do método `main` em um classe

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

- Argumentos do método: `String[] args`
- Conteúdo do método: `...`

- O método `main` declarado como acima sempre será executado ao executar a classe em uma JVM
- **Portanto, em um aplicativo ou projeto, ao menos uma das classes deve possuir o método `main` para que o mesmo possa ser executado**

Gerando uma saída no dispositivo de saída padrão

Imprimindo no terminal

```
System.out.println(‘‘Welcome to Java Programming!’’)
```

- Imprime a *string* de caracteres contidas entre aspas duplas no dispositivo padrão e insere uma quebra de linha
- `System.out` é conhecido como **objeto de saída padrão**
- **Por padrão**, o `System.out` exibirá as strings passadas como parâmetros do comando `print` no terminal, prompt de comando ou shell

Gerando uma saída no dispositivo de saída padrão

Imprimindo no terminal

```
System.out.println(‘‘Welcome to Java Programming!’’);
```

- **OBSERVAÇÃO 1**: cada comando em java deve terminar com um “;”
- **OBSERVAÇÃO 2**: caso não queira quebrar a linha após a impressão de uma *string* na tela, basta chamar o método `print(...)`

Gerando uma saída no dispositivo de saída padrão

Imprimindo no terminal

```
System.out.println(‘‘Welcome to Java Programming!’’)
```

- **OBSERVAÇÃO 3**: caso queira imprimir várias linhas basta chamar o comando `println` repetidas vezes ou inserir o caractere especial "\n" no texto (ex: `System.out.println(‘‘Welcome \n to \n Java \n Programming!’’);`
- **OBSERVAÇÃO 4**: em uma *string*, o caractere \ é chamado de **caractere de escape** e indica a presença de um caractere especial

Imprimindo no terminal

Sequência de Escape	**Descrição**
\t	Insere uma tabulação no texto
\b	Insere um *backspace* no texto
\n	Insere uma nova linha no texto
\r	Insere um retorno de carro (posiciona o curso no início da linha atual)
\'	Insere uma aspas simples no texto
\"	Insere uma aspas duplas no texto
\ \	Insert uma barra invertida no texto

Imprimindo no terminal

- Também pode-se imprimir no dispositivo de saída padrão utilizando o comando `printf`, o qual é utilizado para exibir dados formatados

Imprimindo com o comando `printf`

```
System.out.printf(‘‘%s\n%s\n’’,‘‘Welcome to’’, ‘‘Java
Programming!’’);
```

Programa Básico para Somar Dois Inteiros

```java
//Programa para ler dois números, somá-los e apresentar o resultado da soma
import java.util.Scanner; // O programa faz uso de uma classe Scanner presente
                          // na biblioteca padrão da linguagem Java
                          // A classe escaner permite realizar leitura de fluxos
                          // de dados.

public class Somador {

    public static void main(String[] args){

        //Criando um objeto Scanner para realizar a leitura do teclado
        Scanner teclado = new Scanner(System.in);

        int number1;
        int number2;
        int soma;

        System.out.print("Digite o primeiro número: ");
        number1 = teclado.nextInt(); //Lê um token (unidade de texto) correspondente a
                                     //um inteiro e armazena em number1
        System.out.print("Digite o primeiro número: ");
        number2 = teclado.nextInt();

        soma = number1 + number2;
        System.out.println("O resultado da soma é: " + soma);
    }

}
```

```
Somador > main >
Saída × | Resultados da Pesquisa ×
Console do Depurador × | Teste (run) × | Teste (run) #2 ×
run:
Digite o primeiro número: 5
Digite o primeiro número: 10
O resultado da soma é: 15
CONSTRUÍDO COM SUCESSO (tempo total: 12 segundos)
```

Importando uma classe

- A declaração **import** ajuda o compilador a localizar uma classe utilizada dentro de outra classe
- **OBSERVAÇÃO 1**: Java possui um rico conjunto de classes pré-definidas
- **OBSERVAÇÃO 2**: as classes javas são agrupadas em **pacotes** que juntos formam a **biblioteca de classes Java** ou **Java Application Programing Interface (Java API)**

Importando uma classe

Declaração de um objeto do tipo Scanner

```
Scanner input = new Scanner(System.in);
```

- Na declaração de um objeto têm-se o **tipo do objeto** (classe) o nome do objeto seguido por "=", a palavra-chave "`new`" e novamente o tipo do objeto
- Pode-se passar parâmetros na declaração do objeto (ex: `new Scanner(System.in)`)

Importando uma classe

Declaração de um objeto do tipo Scanner

```
Scanner input = new Scanner(System.in);
```

- Um `Scanner` permite a um programa ler os dados para utilização em um programa
- Antes de utilizar o `Scanner` deve-se especificar a origem dos dados
- Por padrão, **System.in** refere-se ao teclado
- O método `readInt()` da classe `Scanner` converte a entrada em inteiros

Declarando Variáveis

Declarando variáveis inteiras

```
int number1;
int number2;
int sum;
```

Outra forma de se declarar variáveis inteiras

```
int number1, number2, sum;
```

Definindo valores iniciais para variáveis inteiras

```
int number1 = 1;
int number2 = 5;
```

Aritmética em Java

Operação Java	Operador	Expressão Algébrica	Expressão Java
Adição	+	$f + 7$	f + 7
Subtração	-	$p - c$	p - c
Multiplicação	*	bm	b * m
Divisão	/	x / y ou $x \div y$	x / y
Resto (divisão)	%	r *mod* s	r % s

- **OBSERVAÇÃO**: a divisão de inteiros irá resultar em um quociente do tipo inteiro (ex: 7 / 4 irá resultar em 1)

Expressões Aritméticas em Linha Reta

- Expressões em Java devem ser escritas na **forma de linha reta** para facilitar a codificação

- Portanto, expressões como $\frac{a}{b}$ devem ser escritas como a/b, de modo que as contantes, variáveis e operadores apareçam em uma linha reta

Parênteses para Agrupar Subexpressões

- Os parênteses são utilizados para agrupar termos em expressões Java da mesma maneira como em expressões algébricas
  - Ex: para multiplicar a vezes a quantidade $b + c$ escrevemos $a * (b + c)$
- Em casos de parênteses aninhados, executa-se primeiro as expressões dos parênteses mais internos
  - Ex: em $((a + b) * c) + d$, primeiro executa-se $(a + b)$,o resultado da soma será multiplicado por c, e por fim, o resultado da multiplicação será somado com d

Regras de precedência de operadores

Nível de Precedência	Operador	Operação	Ordem de avaliação
1º	*	Multiplicação	Avaliado primeiro. Se houver vários operadores desse tipos, eles são avaliados da esquerda para a direita.
	/	Divisão	
	%	Resto	
2º	+	Adição	Avaliado em seguida. Se houver vário operadores desse tipo, eles são avaliados da esquerda para a direita.
	-	Subtração	
3º	=	Atribuição	Avaliado por último.

Exemplos de Expressões Algébricas em Java

- **Álgebra**: $y = mx + b$
- **Java**: $y = m * x + b$

- **Álgebra**: $m = \frac{a+b+c+d+e}{5}$
- **Java**: $m = (a + b + c + d + e)/5$

Operadores de Igualdade e Operadores Relacionais

- Uma **condição** é uma expressão que pode ser `true` ou `false`
- As condições podem ser formuladas utilizando **operadores de igualdade** (== e !=) e **operadores relacionais** (>, <, >= e <=)
- Operadores de igualdade têm o mesmo nível de precedência entre sí e os operadores de igualdade são associados da esquerda para a direita
- Operadores de igualdade têm precedência mais baixa que operadores relacionais
- Todos os operadores relacionais têm o mesmo nível de precedência e também são associados da esquerda para a direita

Operadores de Igualdade e Operadores Relacionais

```java
import java.util.Scanner;

public class Comparador {

    public static void main(String[] args){
        Scanner teclado = new Scanner(System.in);
        int numero1;
        int numero2;

        System.out.print("Digite o primeiro numero: ");
        numero1 = teclado.nextInt();
        System.out.print("Digite o segundo numero: ");
        numero2 = teclado.nextInt();

        if(numero1 == numero2){
            System.out.println("Os número são iguais");
        }
        if(numero1 != numero2){
            System.out.println("Os número são diferentes");
        }
        if(numero1 > numero2){
            System.out.println("O primeiro número é maior que o segundo número");
        }
        if(numero1 <= numero2){
            System.out.println("O primeiro número é menor ou igual ao segundo número");
        }

    }

}
```

Operadores de Igualdade e Operadores Relacionais

```
Digite o primeiro numero: 5
Digite o segundo numero: 1
Os número são diferentes
O primeiro número é maior que o segundo número
```

```
Digite o primeiro numero: 8
Digite o segundo numero: 13
Os número são diferentes
O primeiro número é menor ou igual ao segundo número
```

```
Digite o primeiro numero: 100
Digite o segundo numero: 100
Os número são iguais
O primeiro número é menor ou igual ao segundo número
```

Exercício

- Faça um programa que:
  - Peça para o usuário digitar o seu salário
  - Peça para o usuário digitar o número de dependentes
  - Imprima o imposto de renda que o usuário deve pagar

Base de Cálculo (R$	Alíquota (%)
Até 1.903,98	-
De 1.903,99 até 2.826,65	7,5
De 2.826,66 até 3.751,05	15
De 3.751,06 até 4.664,68	22,50
Acima de 4.664,68	27,50
Dedução por dependente: R$ 189,59	

Material Complementar

- **Documentação**: **Java Documentation**

  `http://docs.oracle.com/javase/8/docs/api/`

- **Tutorial**: **Java Tutorials**

  `https://docs.oracle.com/javase/tutorial/java/data/characters.html`

- **Vídeo**: Curso de Java #07 - Operadores Aritméticos e Classe Math

  `https://youtu.be/W9V5wtOOZHs?list=PLHz_AreHm4dkI2ZdjTwZA4mPMxWTfNSpR`

- **Vídeo**: Curso de Java #08 - Operadores Lógicos e Relacionais

  `https://youtu.be/xHgnlic7fj8?list=PLHz_AreHm4dkI2ZdjTwZA4mPMxWTfNSpR`

Imagem do Dia [1]

Imagem do Dia [2]

Programação Orientada a Objetos
http://lives.ufms.br/moodle/

Rafael Geraldeli Rossi
rafael.g.rossi@ufms.br

Slides baseados em [Deitel and Deitel, 2010]

Referências Bibliográficas I

Deitel, P. and Deitel, H. (2010).
Java: How to Program.
How to program series. Pearson Prentice Hall, 8th edition.