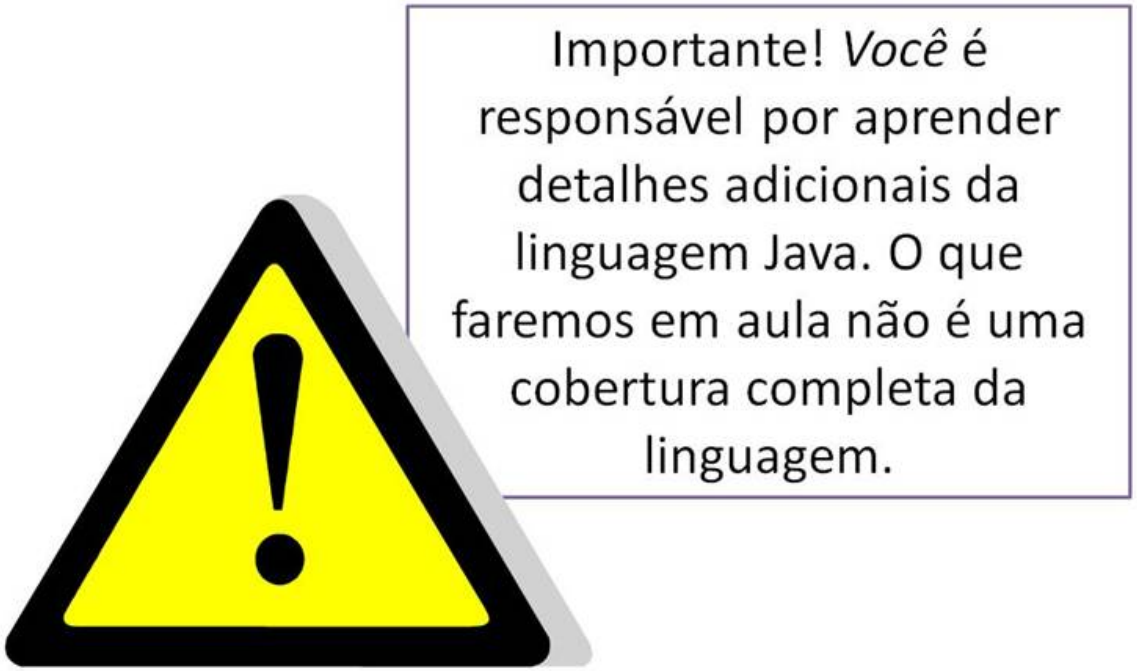
Introdução a Java através de exemplos

Programação 2 – Aulas 1, 2, 3 e 4



Por que Java?

- Linguagem orientada a objetos
- Linguagem poderosa embora simples (alto nível)
  - Permite que o programador faça menos “caca”
- Alguns indicadores de popularidade:
  - Indicador de popularidade das linguagens de programação (TIOBE):

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

- Indicador de popularidade (langpop.com): http://langpop.com/
- Popularidade no twitter: http://smartdatacollective.com/davidmsmith/30761/programming-languages-ranked-popularity

— Permite portabilidade

- Muito usada em programação em rede

  i) Server Programming (JBoss , Tomcat, Glassfish)

  ii) Aplicações de e-commerce, e-business, etc. (Java EE)

  iii) Aplicações para acesso via Internet, intranet, etc. (JSP/Servlets)

— Exemplos de programas escritos em Java: http://www.java.com/en/java_in_action/

— Mais informações sobre características de Java em: http://docs.oracle.com/javase/specs/

Primeiro programa

— Como seria um HelloWorld em python?

```
#
# O primeiro programa em Python: Hello World
#

print "Hello, world!"
```

— Agora em Java:

```java
package p2.exemplos;

/*
 * O primeiro programa em Java: Hello World
 */

// Todo programa tem um ponto de entrada:
// o "método" main de alguma "classe"

public class Hello {

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

— Notam diferenças????

— Vejamos um pouco sobre o que cada linha diz (java):

— A linha abaixo

```java
package p2.exemplos;
```

— É usada para indicar que este programa faz parte de um "pacote" que pode conter vários programas

— Agora o programa tem um nome completo, que inicia com o nome do pacote e tem que estar numa árvore de diretórios idêntica ao que foi definido no pacote (p2/exemplos/Hello.java)

— É uma forma de organizar vários programas, da mesma forma que "pastas" ou "diretórios" são usados para organizar arquivos num sistema de arquivos

Vemos 2 tipos de comentários

— (há mais um tipo muito importante a ser visto adiante)

```
/*
 * O primeiro programa em Java: Hello World
 * Autor: Jacques Sauvé
 */

// Todo programa tem um ponto de entrada: o
// "método" main de alguma "classe"
```

— Esqueça, por enquanto, o que significam “public”, “class”, “static” e “void”

— main é o nome de um método, que é o ponto de entrada do programa

- "Método" é semelhante a "module" em Python
- Outros nomes: sub-rotina, função, procedimento, *procedure*, ...

- O nome do programa é *Hello*
  - Por convenção, deve iniciar com uma letra maiúscula
  - Observe que o programa Hello está obrigatoriamente armazenado no arquivo Hello.java
  - Java é "case-sensitive" (reconhece diferença de caixa)

- Até entender detalhes, sempre use as primeiras duas linhas de código do exemplo, trocando apenas o nome do programa (Hello)

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

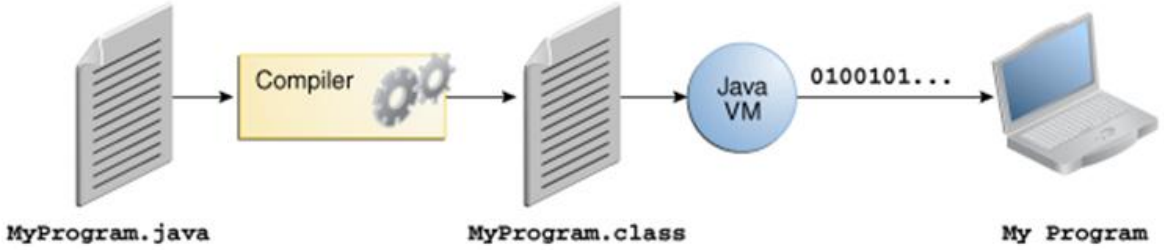
- "{" significa "BEGIN" e "}" significa "END". Isso vai ficar mais claro em outros exemplos...

- Observe a forma de imprimir (System.out.println)
- Observe a formação de uma constante do tipo string (`"Hello, world!"`)
- Observe que tudo termina em ";"

Novidade: Compilação e interpretação

- Como você executa o Hello.py??
  - > python Hello.py (correto??)
  - O que acontece nesse momento de execução?

- A linguagem Java é compilada e interpretada
  - Essa duplinha é o que garante bom desempenho e portabilidade!



Retirado de: http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

- Vamos compilar o programa (no Windows, UNIX, etc.)
- Isso é desnecessário em Python que é apenas interpretado

```
javac Hello.java
```

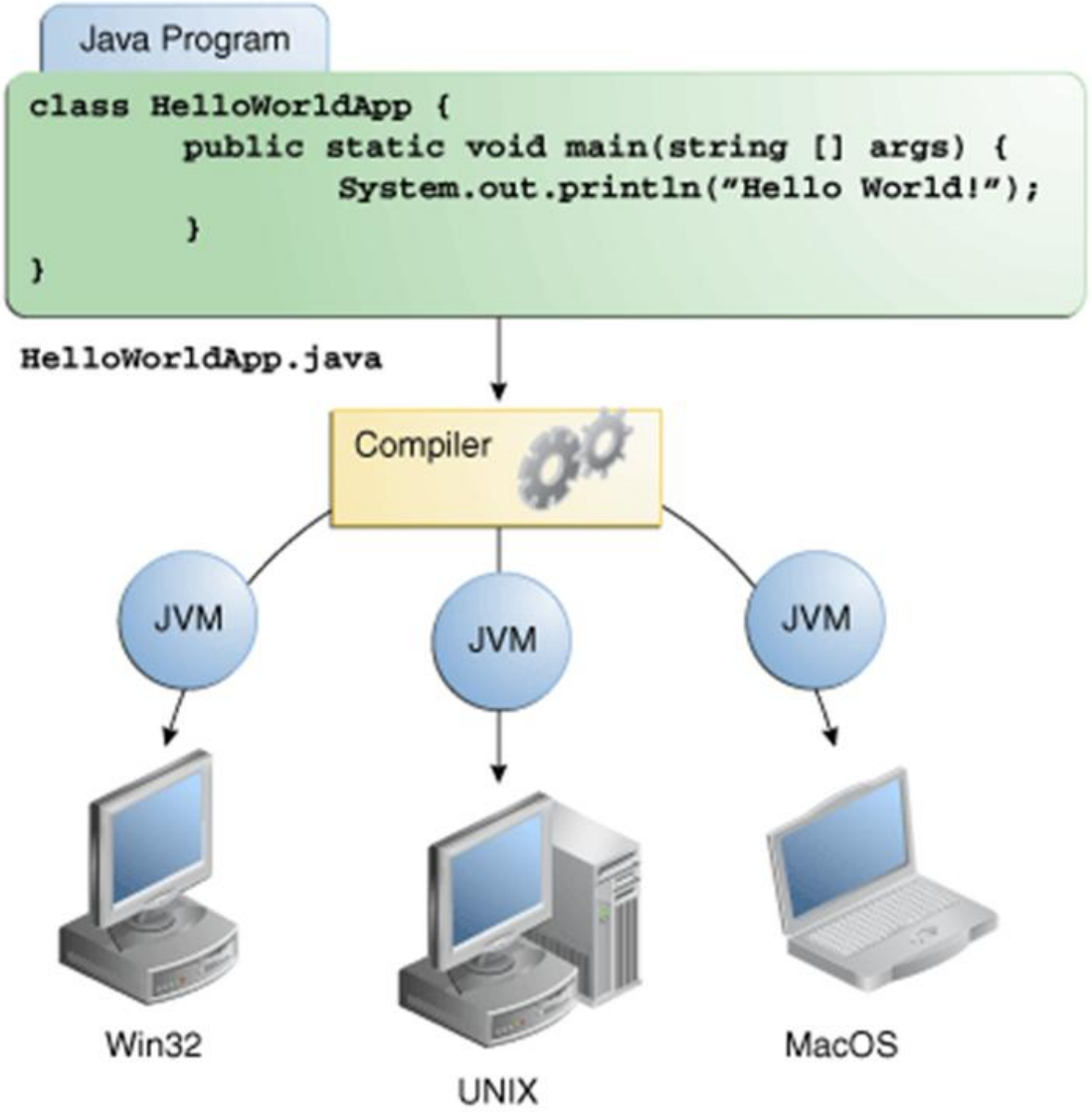
- O comando `javac` é o compilador Java
- O resultado deve estar no arquivo Hello.class (verifique)
- Agora, vamos executar o programa com o comando `java`:

```
java -cp . p2.exemplos.Hello
```

  - -cp indica a partir de onde na árvore de diretórios do sistema o programa p2.exemplos.Hello será procurado (o Hello.class)
  - Isso funciona se você estiver no diretório acima de p2
- O comando java é a "Java Virtual Machine" (JVM) que sabe interpretar um programa Java compilado com o comando `javac` e presente no arquivo

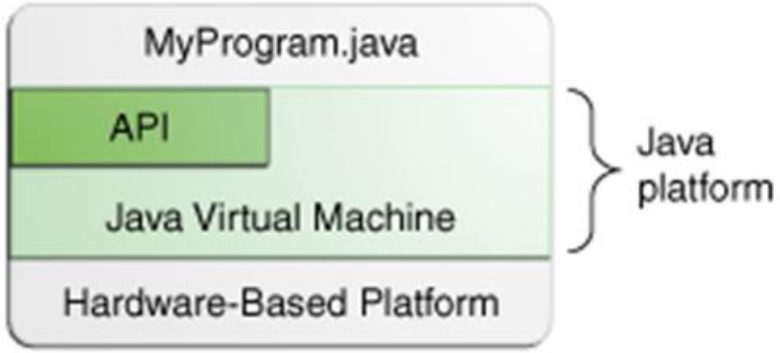
Hello.class

- Isso é diferente de outras linguagens (C, C++) que são diretamente executáveis após a compilação ou têm o código diretamente interpretado
- Motivo: independência de plataforma: Um programa em Java executa em qualquer lugar onde houver uma JVM, sem recompilação



Retirado de: http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

— Em resumo: O seu programa não vai executar no topo de uma plataforma de hardware + sistema operacional; o seu programa Java vai



executar sobre a plataforma Java

Retirado de: http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

— Problemas comuns na compilação e execução de programas Java: http://docs.oracle.com/javase/tutorial/getStarted/problems/index.html

A saída é:

```
Hello, world!
```

— A saída é "a caractere"

- É possível fazer interfaces gráficas com Java
- A disciplina de laboratório tratará do assunto

— É possível usar ambientes integrados de desenvolvimento (IDE) para

programar e depurar em Java
- Eclipse e NetBeans são particularmente populares

Nosso Segundo programa

— Ler 3 números inteiros da entrada, imprimir o menor e o maior
- Entrada de dados
- Tipos básicos
- Variáveis
- Decisões simples

— A primeira solução vem a seguir:

```
package p2.exemplos;

import java.util.Scanner;

public class MinMax1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1, n2, n3;

        System.out.print("Entre com o primeiro inteiro: ");
        n1 = sc.nextInt();
        System.out.print("Entre com o segundo inteiro: ");
        n2 = sc.nextInt();
        System.out.print("Entre com o terceiro inteiro: ");
        n3 = sc.nextInt();
```

```
if (n1 > n2) {
    if (n1 > n3) {
        if (n2 < n3) {
            System.out.println("O menor numero eh: " + n2);
        } else {
            System.out.println("O menor numero eh: " + n3);
        }
        System.out.println("O maior numero eh: " + n1);
    } else {
        if (n1 < n2) {
            System.out.println("O menor numero eh: " + n1);
        } else {
            System.out.println("O menor numero eh: " + n2);
        }
        System.out.println("O maior numero eh: " + n3);
    }
} else {
    if (n2 > n3) {
        if (n1 < n3) {
            System.out.println("O menor numero eh: " + n1);
        } else {
            System.out.println("O menor numero eh: " + n3);
        }
        System.out.println("O maior numero eh: " + n2);
    } else {
        if (n1 < n2) {
            System.out.println("O menor numero eh: " + n1);
```

```
                } else {
                    System.out.println("O menor numero eh: " + n2);
                }
                System.out.println("O maior numero eh: " + n3);
            }
        }
    }
}
```

— Compilando e rodando o programa:

```
javac MinMax1.java
java -cp ..\.. p2.exemplos.MinMax1
```

— Uma saída típica do programa:

```
Entre com o primeiro inteiro: 3
Entre com o segundo inteiro: 9
Entre com o terceiro inteiro: 123
O menor numero eh: 3
O maior numero eh: 123
```

— A linha:

```
import java.util.Scanner;
```

— É usada para dizer ao Java que usaremos alguma coisa externa ao nosso programa (a "classe" Scanner)

— A linha:

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

— Cria um "Scanner" que é usado para ler dados da entrada
  ◦ O que significa “new”, “System.in” serão explicados adiante no curso

— A linha:

```
int n1, n2, n3;
```

— Declara três variáveis inteiras para uso posterior
  ◦ Por convenção, variáveis iniciam com letra minúscula

— A linha...

```
n1 = sc.nextInt();
```

— ... lê um inteiro da entrada

— Também poderíamos ter feito assim:

```
int n1 = sc.nextInt();
```

— As linhas:

```
if (n2 < n3) {
  System.out.println("O menor numero eh: " + n2);
```

```
} else {
  System.out.println("O menor numero eh: " + n3);
}
```

- Mostram uma decisão
- Duas dessas linhas também mostram a concatenação de strings com o operador +
  - Falamos que, em Java, o operador "+" está *overloaded* porque ele significa adição de números e também concatenação de strings, dependendo dos seus operandos
  - É o único operador que sofre *overload*

- A seguir, uma segunda versão do programa

```
package p2.exemplos;

import java.util.Scanner;

/*
 * "Ler 3 números inteiros da entrada, imprimir o menor
 *  e o maior"
 *
 * Autor: Jacques Sauvé
 */

public class MinMax2 {
```

```
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  int n1, n2, n3;

  System.out.print("Entre com o primeiro inteiro: ");
  n1 = sc.nextInt();
  System.out.print("Entre com o segundo inteiro: ");
  n2 = sc.nextInt();
  System.out.print("Entre com o terceiro inteiro: ");
  n3 = sc.nextInt();

  int mínimo;
  int máximo;
  if (n1 > n2) {
    if (n1 > n3) {
      if (n2 < n3) {
        mínimo = n2;
      } else {
        mínimo = n3;
      }
      máximo = n1;
    } else {
      if (n1 < n2) {
        mínimo = n1;
      } else {
        mínimo = n2;
      }
```

```
        máximo = n3;
      }
    } else {
      if (n2 > n3) {
        if (n1 < n3) {
          mínimo = n1;
        } else {
          mínimo = n3;
        }
        máximo = n2;
      } else {
        if (n1 < n2) {
          mínimo = n1;
        } else {
          mínimo = n3;
        }
        máximo = n3;
      }
    }
    System.out.println("O menor numero eh: " + mínimo);
    System.out.println("O maior numero eh: " + máximo);
  }
}
```

- Observe que estamos usando variáveis com acentuação
  - Isso é possível porque Java usa Unicode como código de caracteres, mas não é uma boa prática sobretudo quando se escreve código pra o mundo

— Perguntas sobre este programa:
- Você achou o programa “bem escrito”?
- É fácil de entender?
- É fácil trocar as mensagens de saída por outras?
- É fácil assegurar-se de que não há bugs?
  i) Na realidade, um dos programas acima tem um bug: ache-o!
- É fácil estender para 4 números lidos na entrada?

— Que tal o seguinte programa que resolve o mesmo problema:

```
package p2.exemplos;

import java.util.Scanner;

/*
 * "Ler 3 números inteiros da entrada, imprimir o
 *  menor e o maior"
 *
 * Autor: Jacques Sauvé
 */

public class MinMax3 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int num;
    int mínimo = Integer.MAX_VALUE;
    int máximo = Integer.MIN_VALUE;
```

```
System.out.print("Entre com o primeiro inteiro: ");
num = sc.nextInt();
if (num < mínimo) {
  mínimo = num;
}
if (num > máximo) {
  máximo = num;
}

System.out.print("Entre com o segundo inteiro: ");
num = sc.nextInt();
if (num < mínimo) {
  mínimo = num;
}
if (num > máximo) {
  máximo = num;
}

System.out.print("Entre com o terceiro inteiro: ");
num = sc.nextInt();
if (num < mínimo) {
  mínimo = num;
}
if (num > máximo) {
  máximo = num;
}
```

```
    System.out.println("O menor numero eh: " + mínimo);
    System.out.println("O maior numero eh: " + máximo);
  }
}
```

— É muito mais simples, não é?
  ◦ Por quê?

— A solução pode ser ainda melhor!!! Veja o programa abaixo:

```
package p2.exemplos;

import java.util.Scanner;

/*
 * "Ler 3 números inteiros da entrada, imprimir o
 *  menor e o maior"
 *
 * Autor: Jacques Sauvé
 */

public class MinMax4 {
  public static void main(String[] args) {
    final int NÚMEROS_A_LER = 3;
    Scanner sc = new Scanner(System.in);
    int mínimo = Integer.MAX_VALUE;
    int máximo = Integer.MIN_VALUE;
```

```
    for (int i = 0; i < NÚMEROS_A_LER; i++) {
      System.out.print("Entre com o proximo inteiro:");
      int num = sc.nextInt();
      if (num < mínimo) {
        mínimo = num;
      }
      if (num > máximo) {
        máximo = num;
      }
    }

    System.out.println("O menor numero eh: " + mínimo);
    System.out.println("O maior numero eh: " + máximo);
  }
}
```

- **NÚMEROS_A_LER** é declarado como “final” para indicar que é uma constante
  - Como "const" em Pascal ou C
- Por convenção, usam-se letras maiúsculas para constantes
  - Como em Python, C, C++
- Melhor usar constantes simbólicas do que constantes numéricas
  - Programa fica mais simples de entender

- Observe como fazer um laço "for" na linha:

```
for (int i = 0; i < NÚMEROS_A_LER; i++) {
  //Corpo do for será repetido NÚMEROS_A_LER vezes!
```

```
  // Corpo executado até condição ser satisfeita
}
```

— A expressão **i++** significa **i = i+1**

— Observe também onde a variável “num” foi declarada:

```
int num = sc.nextInt();
```

— De forma geral, é bom declarar uma variável perto de onde ela é usada

  ◦ Fizemos a mesma coisa com a declaração da variável do laço (“i”)

— Ainda podemos deixar o programa em si mais limpo. Veja a versão abaixo:

```
package p2.exemplos;

import java.util.Scanner;

/*
 * "Ler 3 números inteiros da entrada, imprimir o
 *  menor e o maior"
 *
 * Autor: Raquel Lopes
 */

public class MinMax5 {

   public static void main(String[] args) {

      final int NÚMEROS_A_LER = 3;
```

```
    Scanner sc = new Scanner(System.in);
    int mínimo = Integer.MAX_VALUE;
    int máximo = Integer.MIN_VALUE;

    for (int i = 0; i < NÚMEROS_A_LER; i++) {
      int num = recebeProximoInteiro(sc);
      mínimo = menorNumeroEntre(mínimo, num);
      máximo = maiorNumeroEntre(máximo, num);
    }

    System.out.println("O menor numero eh: " + mínimo);
    System.out.println("O maior numero eh: " + máximo);
}

private static int maiorNumeroEntre(int numero1, int numero2) {
    return numero2 > numero1? numero2: numero1;
}

private static int menorNumeroEntre(int numero1, int numero2) {
    return numero2 < numero1? numero2: numero1;
}

private static int recebeProximoInteiro(Scanner sc) {
    System.out.print("Entre com o proximo inteiro:");
    int num = sc.nextInt();
    return num;
}
```

```
}
```

- Note a presença do operador ternário nos métodos maiorNumeroEntre e menorNumeroEntre

```
private static int maiorNumeroEntre(int numero1, int numero2) {
    return numero2 > numero1? numero2: numero1;
}
//Equivale a:
private static int maiorNumeroEntre(int numero1, int numero2) {
    if (numero2 > numero1) {
        return numero2;
    }
    return numero1;
}
```

- Esqueça por enquanto a palavra **`private`**
  - Criamos métodos, que são como funções que podem ser chamadas de dentro do main;
  - Também seria possível chamar estes métodos dentro de outros métodos, como veremos mais adiante;
- Criamos três métodos: **`recebeProximoInteiro`**, **`menorNumeroEntre`** e **`maiorNumeroEntre`**
  - Criação de métodos deixa o programa mais limpo, isto é, mais fácil de entender
  - Criação de métodos permite que eles sejam reusados em outras partes do mesmo programa ou até em outros programas
    i) Reuso é um requisito geralmente perseguido quando programamos no paradigma orientado a objetos

Nosso terceiro programa em Java

— Antes um pouco de teoria

- Não precisa decorar essas coisas! Basta saber que existem… (referências são feitas para se consultar!)

— Tipos primitivos, limites de representação e constantes:

Tipo primitivo	**Tamanho**	**Mínimo**	**Máximo**	**Exemplos de Constantes**
boolean	1 bit	-	-	true, false
char	16 bits	Unicode 0	Unicode 65.535	'a' (letra a)
byte	8 bits	-128	+127	97, -23 0x65 (hexadecimal)
short	16 bits	-2^{15} (-32.768)	$+2^{15}-1$ (32.767)	17569, -21875
int	32 bits	-2^{31} (uns -2 bi)	$+2^{31}-1$ (uns 2 bi)	1876345240, -2000000000
long	64 bits	-2^{63} (uns -9 quintilhões)	$+2^{63}-1$ (uns 9 quintilhões)	123981723971982318273L, -12381726387613678688L, 97L, -23L, 0L (Observe o 'L' final)
float	32 bits	aprox -10^{38} (6-7 dígitos significativos)	aprox $+10^{38}$ (6-7 dígitos significativos)	-3.4F 45.78E+23F (Observe o 'F' final)

double	64 bits	aprox -10^{308} (15 dígitos significativos)	aprox $+10^{308}$ (15 dígitos significativos)	-3.4 45.78E+23
void	-	-	-	indica ausência de tipo

— A conversão entre tipos, quando necessária, é feita com *casts* explícitos

```
double x = 8.89;
int n = (int)x; // n terá valor 8
```

— Operadores
- + (soma)
- - (subtração)
- (multiplicação)
- / (divisão)
- % (módulo)
- Há operadores unários - e +
- Operadores binários podem ser seguido de =

```
soma += nota*peso;
// equivalente a soma = soma + nota*peso
```

— Outros operadores:
- Operador de String + (overloaded)
- Operadores de auto-incremento e auto-decremento ++ e --

```
númeroDeAlunos++;
// equivalente a númeroDeAlunos = númeroDeAlunos + 1
númeroDeAlunos--;
// equivalente a númeroDeAlunos = númeroDeAlunos - 1
if(númeroDeAlunos-- > 0)
// equivalente a testar númeroDeAlunos e depois decrementar
if(--númeroDeAlunos > 0)
// equivalente a decrementar e depois testar númeroDeAlunos
```

- Operadores relacionais:
  - < (menor)
  - <= (menor ou igual)
  - > (maior)
  - >= (maior ou igual)
  - == (igual)
  - != (não igual)
- Operadores lógicos
  - && (AND)
  - || (OR)
  - ! (NOT)
  - Exemplos:

```
if(númeroDeAlunos > MAX_ALUNOS || númeroDeProfessores > MAX_PROFS) …

if(ano % 4 == 0 && ano % 100 != 0 || ano % 400 == 0)…
// ano bissexto
```

Operadores de bits e de deslocamento

- &, &=, |, |=, ^, ^=, ~
- Não falaremos deles aqui

Operador ternário

- Para escrever uma operação condicional sem usar **`if-else`**
- Exemplo segue

```
// a linha seguinte
média = númeroDeNotas == 0 ? 0.0 : soma/númeroDeNotas;
// é equivalente às linhas seguintes
if(númeroDeNotas == 0) {
    média = 0.0;
} else {
    média = soma/númeroDeNotas;
}
```

De forma geral, as precedências "esperadas" funcionam

```
// a linha seguinte
if( númeroDeAlunos > MAX_ALUNOS ||
    númeroDeProfessores > MAX_PROFS) …
// não precisa de parênteses, pois é equivalente a
if((númeroDeAlunos > MAX_ALUNOS) || (númeroDeProfessores > MAX_PROFS)) …
```

Existe uma tabela de precedências:

Operadores	**Precedência**
[] . ()	Mais alta

! ~ ++ -- + (unário) - (unário) (cast) new	
* / %	
+ -	
<< >> >>>	
< <= > >= instanceof	
== !=	
&	
^	
|	
&&	
||	
?:	
= += -= *= /= %= &= |= ^= <<= >>= >>>=	Mais baixa

- Na dúvida, use parênteses
- Não decore a tabela de precedências!

- Abaixo segue um programa exemplo
  - Este programa auxilia o planejamento da aposentadoria

```
package p2.exemplos;

import java.util.Scanner;

/*
```

```
 * Planejamento de aposentadoria
 * Autor: Raquel Lopes
 */

public class Aposentadoria {

   public static void main(String[] args) {
      final double QUANTIA_MINIMA_TOTAL = 10000;
      final double MENOR_CONTRIBUICAO = 200;

      System.out.print("Quanto dinheiro voce quer para se aposentar? ");
      double alvo = recebeDoubleMaiorQue(QUANTIA_MINIMA_TOTAL);

      System.out.print("Quanto dinheiro voce vai contribuir todo ano?");
      double contribuicaoAnual = recebeDoubleMaiorQue(MENOR_CONTRIBUICAO);

      System.out.print("Taxa de juros (ex.: digite 0,075 para 7,5%): ");
      double juros = recebePercentual();

      System.out.println("Voce pode se aposentar em "
            + computaAnosDeContribuicao(alvo, contribuicaoAnual, juros)
            + " anos.");
   }

   private static int computaAnosDeContribuicao(double alvo,
                                                double contribuição,
                                                double juros) {
      int anos = 0;
      double saldo = 0;
      while (saldo <  alvo) {
            saldo = (saldo + contribuição) * (1 + juros);
```

```
            anos++;
        }
        return anos;
    }

    private static double recebePercentual() {
        Scanner sc = new Scanner(System.in);
        if (!sc.hasNextDouble()) {
            sc.next();
            System.out.println("Voce deve digitar um valor real entre 0 e 1.");
            return recebePercentual();
        }
        double valor = sc.nextDouble();
        if (valor > 1 || valor < 0) {
            System.out.println(" Voce deve digitar um valor real entre 0 e
1.");
            return recebePercentual();
        }
        return valor;
    }

    private static double recebeDoubleMaiorQue(double minimo) {
        Scanner sc = new Scanner(System.in);
        if (!sc.hasNextDouble()) {
            sc.next();
            System.out.println("Voce deve digitar um valor real maior que "
                               + minimo);
            return recebeDoubleMaiorQue(minimo);
        }
        double valor = sc.nextDouble();
        if (valor < minimo) {
```

```
            System.out.println("Voce deve digitar um valor real maior que "
                               + minimo);
            return recebeDoubleMaiorQue(minimo);
        }
        return valor;
    }
}
```

Nosso quarto programa

- Vai nos ajudar a lidar com arrays
- Java tem arrays unidimensionais e multidimensionais
- Segue um programa que lê 10 números e os imprime em ordem inversa

```
package p2.exemplos;

import java.util.Scanner;

/*
 * Ler 10 números inteiros da entrada, imprimir em ordem inversa
 * Autor: Raquel Lopes
 */

public class Inverte1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] números = new int[10];
        // criação do array de 10 inteiros
```

```
        for (int i = 0; i < 10; i++) {
          System.out.print("Entre com o proximo inteiro: ");
          números[i] = sc.nextInt();
        }
        for (int i = 10 - 1; i >= 0; i--) {
          System.out.print(números[i] + " ");
        }
    }
}
```

- O que acharam deste programa?
  - Será que ele pode ser melhorado?

```
package p2.exemplos;

import java.util.Scanner;

/*
 * Ler 10 números inteiros da entrada, imprimir em ordem inversa
 * Autor: Jacques Sauvé
 */

public class Inverte {
  public static void main(String[] args) {
    final int NÚMEROS_A_LER = 10;
    Scanner sc = new Scanner(System.in);
    int[] números = new int[NÚMEROS_A_LER];
    // criação do array de 10 inteiros
```

```
    for (int i = 0; i < números.length; i++) {
      System.out.print("Entre com o proximo inteiro: ");
      números[i] = sc.nextInt();
    }
    for (int i = números.length - 1; i >= 0; i--) {
      System.out.println(números[i]);
    }
  }
}
```

- Observe que arrays são sempre indexados a partir de zero
- Precisa saber o tamanho para criar o array
- **números.length** é o número de elementos no array números
- No laço, é preferível usar **números.length** a usar o número 10
  - Por quê?
- Segue um programa que ecoa os argumentos de linha de comando

```
package p2.exemplos;

/*
 * Ecoar argumentos de linha de comando
 * Autor: Jacques Sauvé
 */

public class Eco {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
```

```
            System.out.print(args[i] + " ");
        }
        System.out.println();
        for (int i = 0; i < args.length; i++) {
            System.out.println(args[i]);
        }
    }
}
```

- Observe que **`args`** é um array normal composto de Strings
- Segue um programa que mostra uma forma de inicializar arrays

```
package p2.exemplos;

/*
 * Imprime o dia da semana correspondendo ao argumento de linha de
 * comando
 *
 * Autor: Jacques Sauvé
 */

public class Dia {
  public static void main(String[] args) {
    final int DIAS_NA_SEMANA = 7;
    final String[] diasDaSemana = { "", "Domingo", "Segunda",
                "Terca", "Quarta", "Quinta", "Sexta", "Sabado" };

    if (args.length != 1) {
```

```
      System.err.println("Sintaxe: Dia numero");
      System.exit(1);
    }

    int dia = Integer.parseInt(args[0]);
    if (dia < 1 || dia > DIAS_NA_SEMANA) {
      System.err.println("O dia da semana deve estar entre 1 e " +
                         DIAS_NA_SEMANA);
      System.exit(1);
    }

    System.out.println(diasDaSemana[dia]);
  }
}
```

- A inicialização do array **diaDaSemana** já calcula o tamanho necessário
- É recomendado usar **System.*err*** em vez de **System.*out*** para imprimir erros
- **System.*exit*(0)** é usado para terminar o programa "bem"
- **System.exit(1)** é usado para terminar o programa "mal"
- **Integer.parseInt(xpto)** converte o string **xpto** para um inteiro
- Teste: retire o teste que começa com a seguinte linha ...

```
if (dia < 1 || dia > DIAS_NA_SEMANA) {
```

- E veja o que acontece quando passamos um argumento maior que 7!

— Você pode obter mais informação e dicas sobre entrada e saída em Java aqui
  - Você poderá não entender tudo que está aí agora, mas volte a ler este material sobre entrada e saída de vez em quando ao longo da disciplina. Você entenderá mais a cada leitura.
  - Recomendamos fortemente a leitura da seção Scanning and Formatting

— Exercícios sugeridos:
  - http://docs.oracle.com/javase/tutorial/getStarted/QandE/questions.html