

Aspectos Fundamentais

Tipos de Dados Primitivos

Tipo	Quantidade de Bits	Exemplo
char	16	'a'
byte	8	00000001
int	32	1
short	16	1
long	64	1
float	32	2.99
double	64	2.99
boolean	8	true

Palavras reservadas

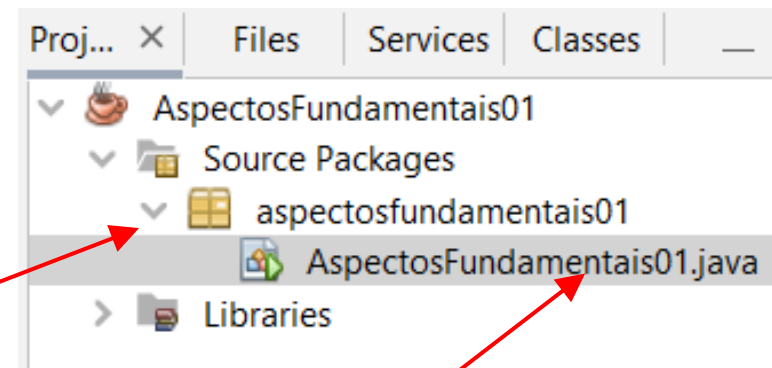
- Como já sabemos palavras reservadas não podem ser nomes de variáveis.
- Java possui diversas palavras reservadas.

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

Declaração de variáveis

- É possível declarar variáveis de quaisquer tipos primitivos: char, byte, int, long, float, double ou boolean.
- Assim como em C programas feitos em Java devem ter suas variáveis inicializadas antes de serem utilizadas.
- No próximo slide veremos um exemplo de declaração de variáveis.

Exemplo:



```
package aspectosfundamentais01;
```

```
public class AspectosFundamentais01 {  
    public static void main(String[] args) {  
        int x = 1, y = 2;  
        double z = 2.99;  
        System.out.println(x);  
        System.out.println(y);  
        System.out.println(z);  
    }  
}
```

Declaração de constantes

- Não existem constantes em Java, mas existe um tipo de variável com comportamento semelhante ao de outras linguagens.

- Em C podemos declarar uma constante da seguinte forma:

```
#define PI 3.1416
```

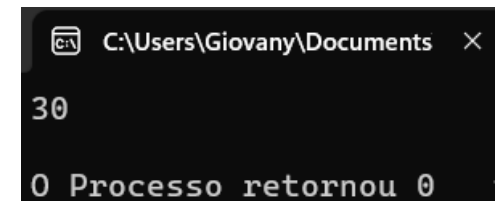
```
#define X 10
```

- Em Java devemos fazer:

- final double pi = 3.14;

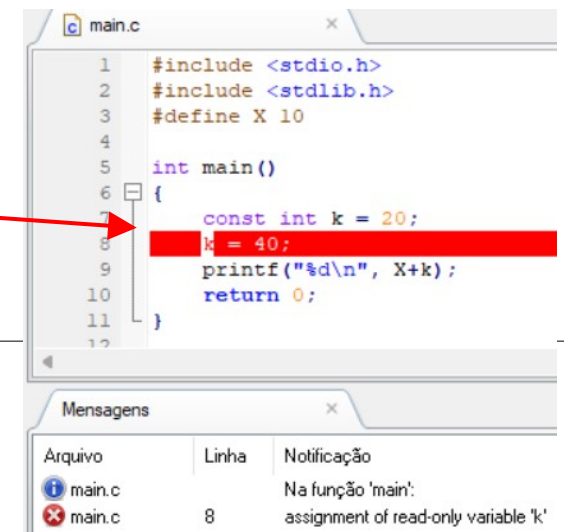
```
#include <stdio.h>
#include <stdlib.h>
#define X 10

int main()
{
    const int k = 20;
    printf("%d\n", X+k);
    return 0;
}
```



```
C:\Users\Giovany\Documents
30
0 Processo retornou 0
```

- Variáveis declaradas com a palavra **final** não podem ser alteradas, se comportando como constantes.
- Em C podemos ter o mesmo efeito utilizando a palavra **const**.



Comentários

- Java aceita 3 tipos de comentários:
 - Comentando uma única linha: `//`
 - Para comentar várias linhas: `/* */`
 - Para comentar várias linhas e gerar documentação com Javadoc: `/** */`
- Javadoc é um programa gerador de documentação em HTML, fornecido pela Sun junto com o SDK.

Exemplo – Comentários e variáveis final

```
package aspectosfundamentais02;
public class AspectosFundamentais02 {

    /*
     * Comentário de várias linhas
     * abaixo vemos um comentário Javadoc
     */

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Comentário de uma linha
        final double z = 2.99;
        int x = 1, y = 2;
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
    }
}
```

cannot assign a value to final variable z
The assigned value is never used

(Alt-Enter shows hints)

z = 1.99;

Run	Debug	Profile	Team	Tools	Window	Help
▶	Run Project (AspectosFundamentais02)					F6
	Test Project (AspectosFundamentais02)					Alt+F6
🔧	Build Project (AspectosFundamentais02)					F11
🧹	Clean and Build Project (AspectosFundamentais02)					Shift+F11
	Build Batch Projeto... (AspectosFundamentais02)					
	Set Project Configuration					>
	Set Project Browser					>
	Set Main Project					>
🪞	Open Java Shell for Project (AspectosFundamentais02)					
	Generate Javadoc (AspectosFundamentais02)					

Method Details

main

public static void main(String[] args)

Parameters:

args - the command line arguments

Operadores

- Os operadores:

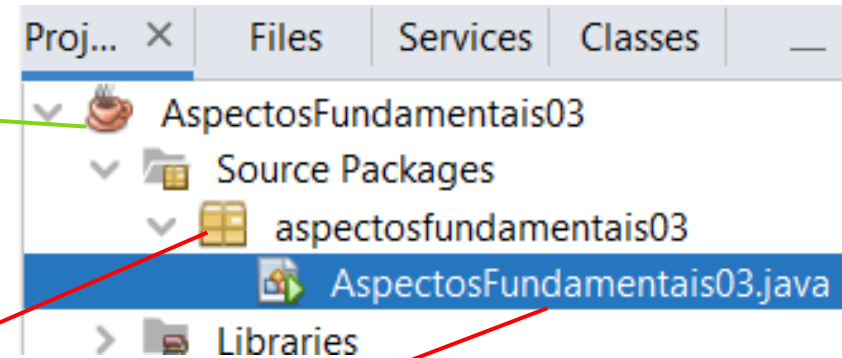
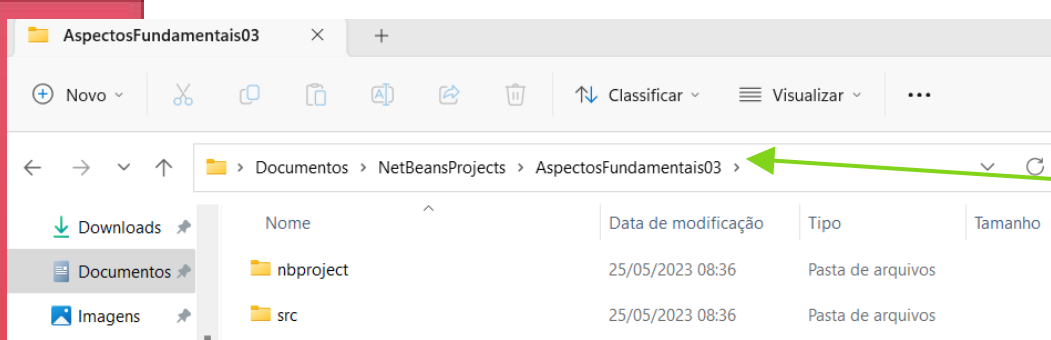
- Aritméticos
- Relacionais
- Lógicos

**São os mesmos
da linguagem C.**

Operador	Ação
+	Soma
-	Subtração ou Troca de sinal
*	Multiplicação
/	Divisão
%	Resto de divisão (de inteiros)
++	Incremento
--	Decremento

Operador	Ação
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

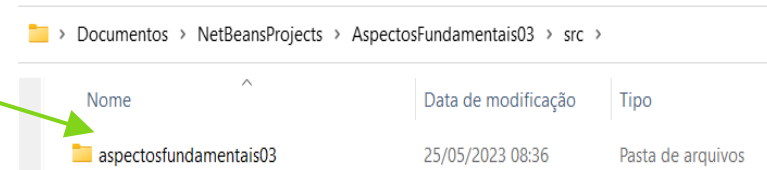
Operador	Ação
&&	AND (E)
	OR (OU)
!	NOT (NÃO)



```
package aspectosfundamentais03;

public class AspectosFundamentais03 {

    public static void main(String[] args) {
        int x = 10; int y = 3;
        System.out.println("X = " + x);
        System.out.println("Y = " + y);
        System.out.println("-X = " + (-x));
        System.out.println("X/Y = " + (x/y));
        System.out.println("O resto de X por Y = " + (x%y));
        System.out.println("Inteiro de X por Y = " + (int) (x/y));
        System.out.println("X + 1 = " + (x++));
    }
}
```



Exemplo - Operadores

Passagem de Parâmetros

- Uma aplicação em Java, assim como em C, pode receber valores a partir da linha de comando (prompt).
- Para executar um programa Java com parâmetros basta utilizar o seguinte comando no prompt:

```
java -jar <nome_do_.jar> <argumentos>
```

Ex:

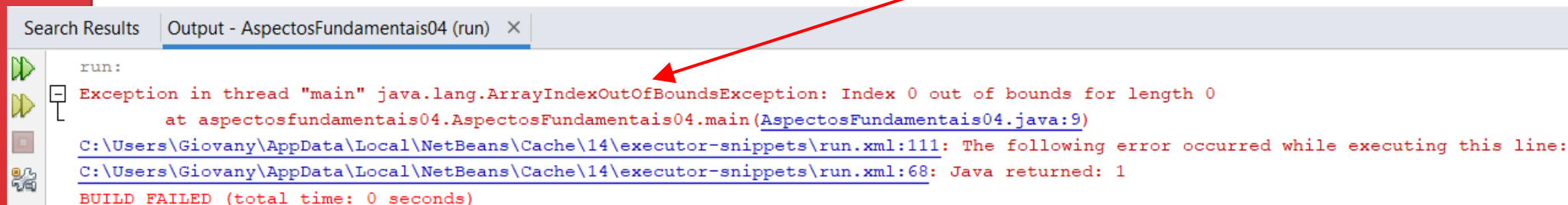
```
java -jar AspectosFundamentais.jar X Y
```

Exemplo Passagem de Parâmetros

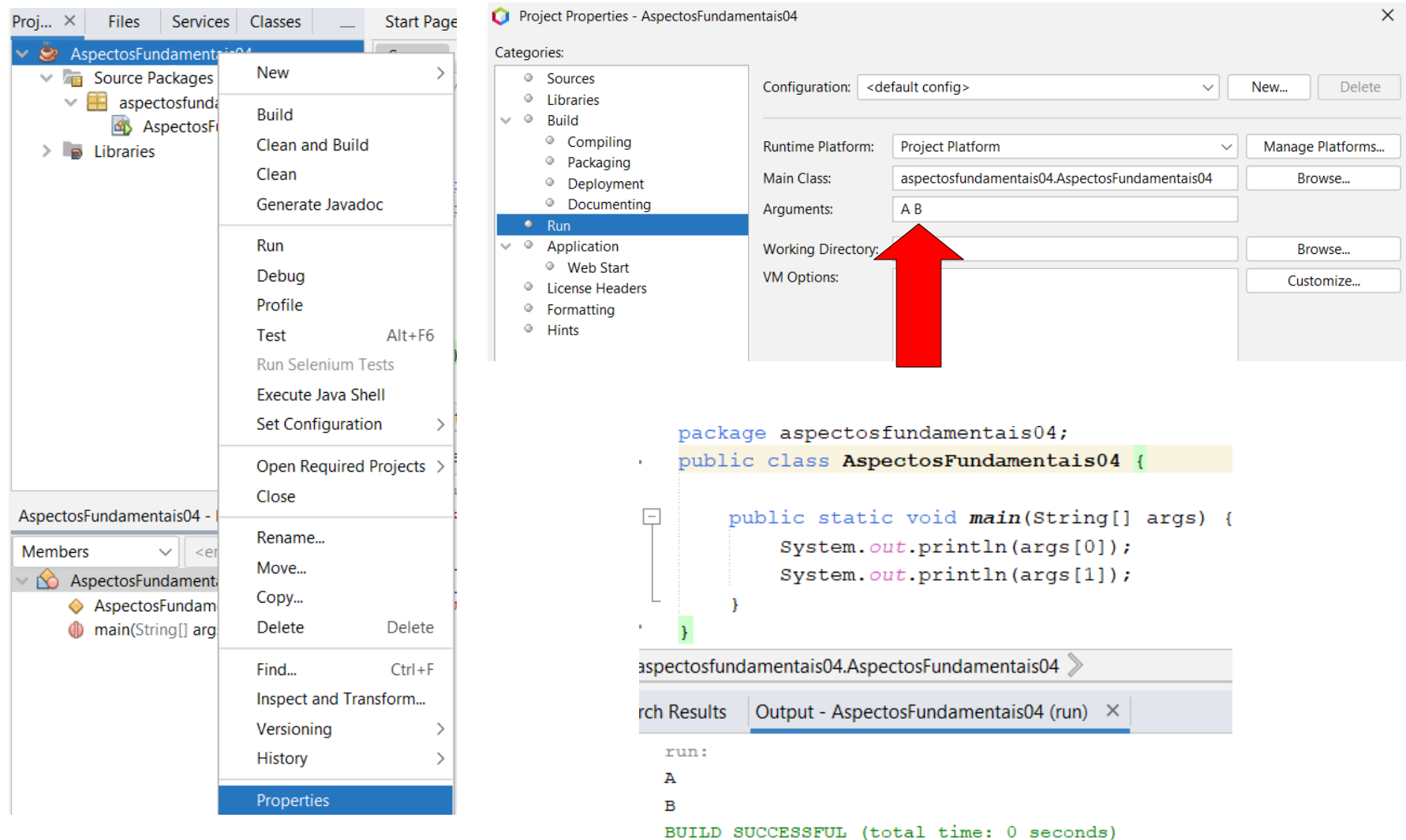
```
package aspectosfundamentais04;  
public class AspectosFundamentais04 {  
  
    public static void main(String[] args) {  
        System.out.println(args[0]);  
        System.out.println(args[1]);  
    }  
}
```

Invasão de Memória

Acessando posição inválida no Array (vetor)



Exemplo Passagem de Parâmetros



The screenshot illustrates the process of passing parameters to a Java application. It shows the IDE's 'Run' menu, the 'Project Properties' dialog with the 'Run' category selected, and the 'Arguments' field set to 'A B'. The code editor displays the following Java code:

```
package aspectosfundamentais04;

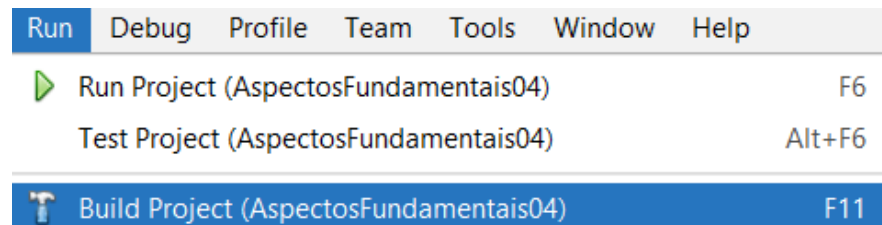
public class AspectosFundamentais04 {

    public static void main(String[] args) {
        System.out.println(args[0]);
        System.out.println(args[1]);
    }
}
```

The output console shows the results of the run:

```
run:
A
B
BUILD SUCCESSFUL (total time: 0 seconds)
```

Exemplo Passagem de Parâmetros



A screenshot of the NetBeans 'Output' window for the project 'AspectosFundamentais04 (jar)'. The window shows the following text:

```
ant -f C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04 -Dnb.internal.action.name=build jar
init:
Deleting: C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\build\\built-jar.properties
deps-jar:
Updating property file: C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\build\\built-jar.properties
compile:
Copying 1 file to C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\build
Nothing to copy.
To run this application from the command line without Ant, try:
java -jar "C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\dist\\AspectosFundamentais04.jar"
deploy:
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Two red arrows point to the command line output: one points to the full command `java -jar "C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\dist\\AspectosFundamentais04.jar"`, and another points to the `jar:` section.

A screenshot of a Windows Command Prompt window titled 'Prompt de Comando'. It shows the following text:

```
Microsoft Windows [versão 10.0.22621.1702]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\\Users\\Giovany>java -jar "C:\\Users\\Giovany\\Documents\\NetBeansProjects\\AspectosFundamentais04\\dist\\AspectosFundamentais04.jar" XYZ KTT
XYZ
KTT
```

O “tipo” String

- Em Java existe um “tipo” String, na realidade String, em Java, é uma classe, mas devido ao sua facilidade de uso começaremos a tratá-la antes mesmo de falar de classes.
- Por enquanto podemos visualizar uma String como um vetor de caracteres, assim como havíamos aprendido em C.

Conversão de tipos

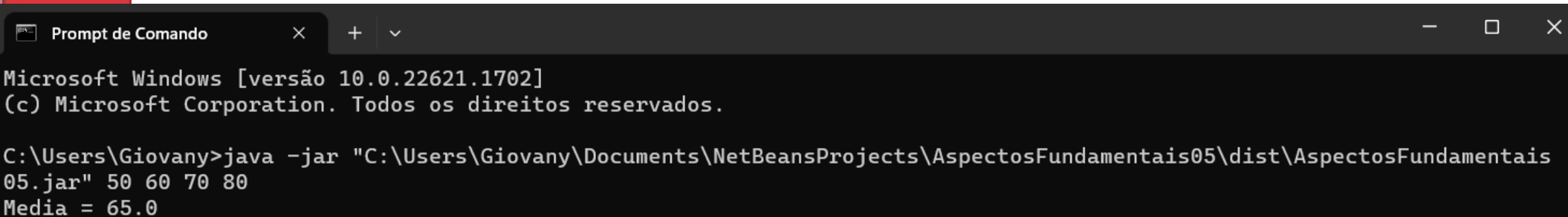
- É muito comum a conversão de tipos. Por exemplo se quiséssemos receber números e não `String[]` como foi passado no parâmetro `args`. Isso não seria possível pois os parâmetros de qualquer método em Java são tipados e somos obrigados a fornecer `String[]` como parâmetro.
- Podemos então para cada `String` em `String[]` fazer a conversão para o tipo numérico desejado.

Exemplo – Conversão de tipos

```
package aspectosfundamentais05;

public class AspectosFundamentais05 {

    public static void main(String[] args) {
        double nota1, nota2, trabalho1, trabalho2, media;
        nota1 = Double.parseDouble(args[0]);
        nota2 = Double.parseDouble(args[1]);
        trabalho1 = Double.parseDouble(args[2]);
        trabalho2 = Double.parseDouble(args[3]);
        media = ( nota1 + nota2 + trabalho1 + trabalho2 ) / 4;
        System.out.println("Media = " + media);
    }
}
```



The screenshot shows a Windows Command Prompt window with the title 'Prompt de Comando'. The window displays the output of running a Java program. The command entered is 'java -jar "C:\Users\Giovany\Documents\NetBeansProjects\AspectosFundamentais05\dist\AspectosFundamentais05.jar" 50 60 70 80'. The output is 'Media = 65.0'.

```
Prompt de Comando
Microsoft Windows [versão 10.0.22621.1702]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Giovany>java -jar "C:\Users\Giovany\Documents\NetBeansProjects\AspectosFundamentais05\dist\AspectosFundamentais05.jar" 50 60 70 80
Media = 65.0
```

Tabela de conversão de tipos

Declaração da variável a	Conversão para	b recebe a
int a = 20;	float	float b = (float) a;
int a = 20;	double	double b = (double) a;
int a = 20;	String	String b = String.valueOf(a);
float a = 20.1;	int	int b = (int) a;
float a = 20.1;	double	double b = (double) a;
float a = 20.1;	String	String b = String.valueOf(a);
double a = 20.1;	int	int b = (int) a;
double a = 20.1;	float	float b = (float) a;
double a = 20.1;	String	String b = String.valueOf(a);
String a = "23";	int	int b = Integer.parseInt(a);
String a = "23.3";	float	float b = Float.parseFloat(a);
String a = "23.3";	double	double b = Double.parseDouble(a);

Usando o Teclado para Entrada de Dados

```
package aspectosfundamentais06;
```

```
import java.io.DataInputStream;  
import java.io.IOException;
```

```
public class AspectosFundamentais06 {
```

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

```
        String s;
```

```
        float notal, nota2, media;
```

```
        DataInputStream dado;
```

```
        try {
```

```
            System.out.println("Entre com a nota 1");
```

```
            dado = new DataInputStream(System.in);
```

```
            s = dado.readLine();
```

```
            notal = Float.parseFloat(s);
```

```
            System.out.println("Entre com a nota 2");
```

```
            dado = new DataInputStream(System.in);
```

```
            s = dado.readLine();
```

```
            nota2 = Float.parseFloat(s);
```

```
            media = (notal + nota2) / 2;
```

```
            System.out.println("Media: " + media);
```

```
        } catch (IOException erro) {
```

```
            System.out.println("Erro na entrada de dados");
```

```
        } catch (NumberFormatException erro) {
```

```
            System.out.println("Houve erro na conversão, digite apenas caracteres numericos");
```

```
        }
```

```
    }
```

```
}
```

Bibliotecas necessárias (deixe o NetBeans importá-las)

Classe para leitura de dados via teclado

Criação do objeto associando-o a entrada de dados System.in

O que é isso?

E se o usuário digitar algo que não é número?

Usando o Teclado para Entrada de Dados – outra forma

```
package aspectosfundamentais07;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class AspectosFundamentais07 {
    public static void main(String[] args) {
        String s;
        float nota1, nota2, media;
        BufferedReader dado;
        try {
            System.out.println("Entre com a nota 1");
            dado = new BufferedReader(new InputStreamReader(System.in));
            s = dado.readLine();
            nota1 = Float.parseFloat(s);
            System.out.println("Entre com a nota 2");
            dado = new BufferedReader(new InputStreamReader(System.in));
            s = dado.readLine();
            nota2 = Float.parseFloat(s);
            media = (nota1 + nota2) / 2;
            System.out.println("Media: " + media);
        } catch (IOException erro) {
            System.out.println("Erro na entrada de dados");
        } catch (NumberFormatException erro) {
            System.out.println("Houve erro na conversão, digite apenas caracteres numericos");
        }
    }
}
```

Nessa outra forma de ler dados via teclado

utilizamos a classe **BufferedReader**.

Os códigos são bastante semelhantes.

É preferível o uso do **BufferedReader**, pois o método

readline() da classe **DataInputStream** é **deprecated** (obsoleto).

No NetBeans métodos **deprecated** são representados por um traço cortando o nome do método.

Dúvidas?

