

**JGrasp**

Roberto Rocha

# Linguagem Java

# Instalando o JGrasp

Acesse o site do [jgrasp.org](https://jgrasp.org/)

<https://jgrasp.org/>

**jGRASP**

Um ambiente de desenvolvimento integrado com visualizações para melhorar a compreensão do software

[Home](#)  
[Download](#)  
[Fale Conosco](#)  
[Membros da Equipe](#)  
[Recursos](#)  
[Arquivo](#)  
[Política de Privacidade](#)  
[Suporte jGRASP](#)

Documentação  
jGRASP Hello

A versão atual do jGRASP é a versão 2.0.6\_07 (25 de novembro de 2020).

O Android Studio 3.6.2 é compatível com o plug-in jGRASP para IntelliJ (as versões anteriores do Android Studio não eram).

A versão atual do [plugin jGRASP para IntelliJ](#) é a versão 1.0.0 (11 de junho de 2020).

A versão atual do [plugin jGRASP para Eclipse](#) é a versão 1.0.0 Beta 8 (29 de janeiro de 2020).



[Baixar jGRASP](#)



[Instituições que usam jGRASP](#)

Clique em **Download**

# Instalando o JGrasp

Escolha a ultima versão disponível, de preferência com o pacote OpenJDK incluído!

**GRASP 2.0.6\_07 (25 de novembro de 2020)**

**Incluído com OpenJDK 15.0.1, Checkstyle 8.35 e JUnit 4.13**

**jGRASP pacote exe**

**Windows 64 bits (Vista ou superior)** : executável de extração automática (183.093.848 bytes).

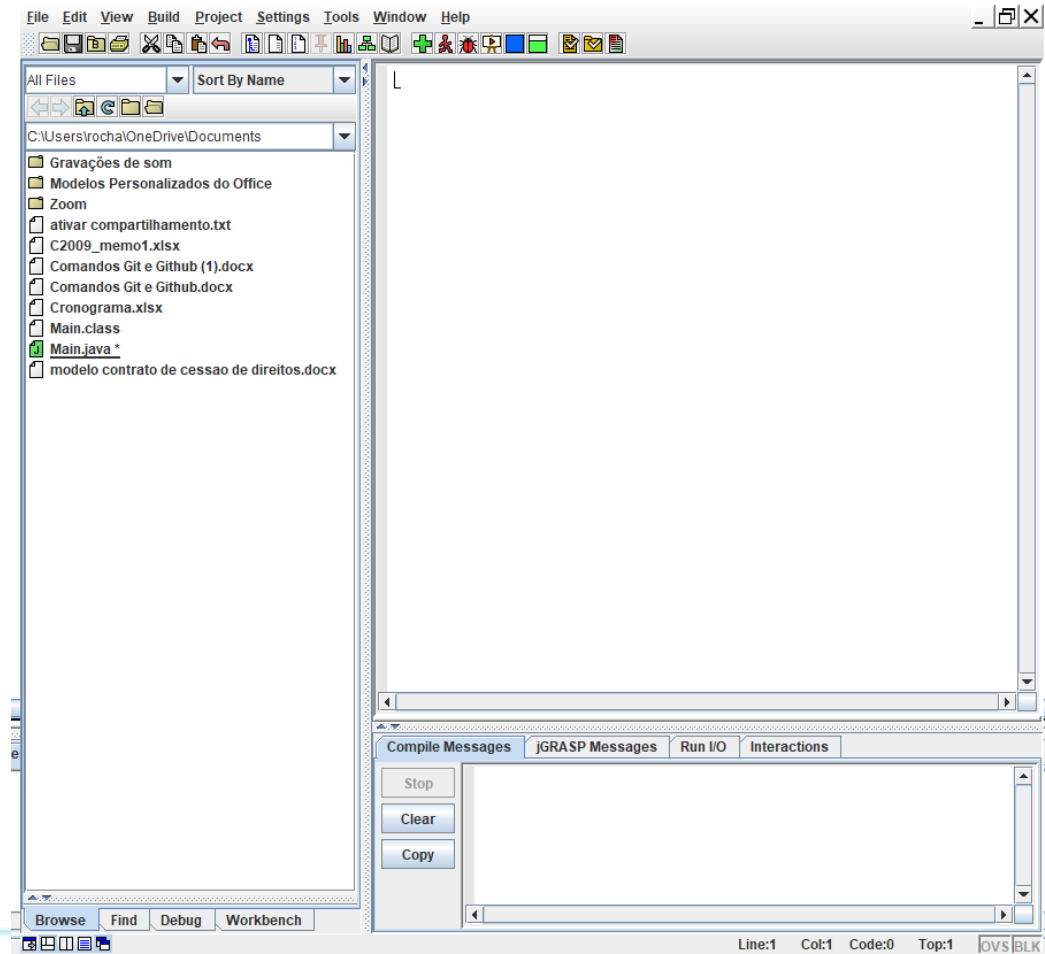
**Pacote de pacote jGRASP**

**macOS (High Sierra or Higher)** : arquivo de instalação do pacote (requer acesso de administrador para instalar) (211.575.868 bytes).

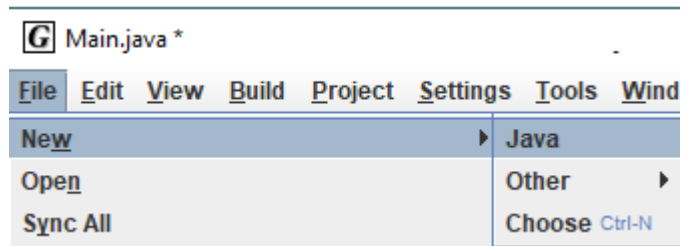
**jGRASP compactado zip**

**Intel Linux de 64 bits** : arquivo zip (219.911.765 bytes).

# Abra o JGrasp

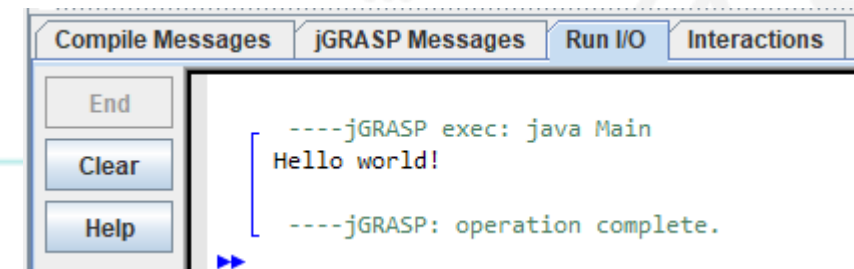
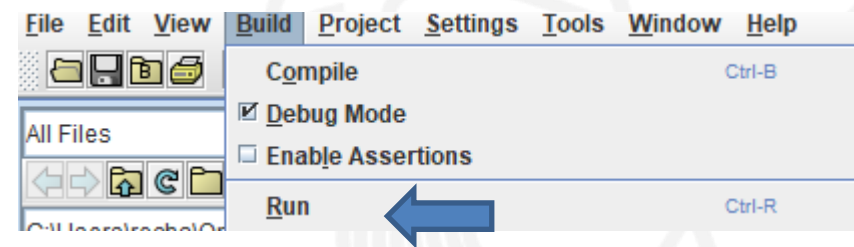
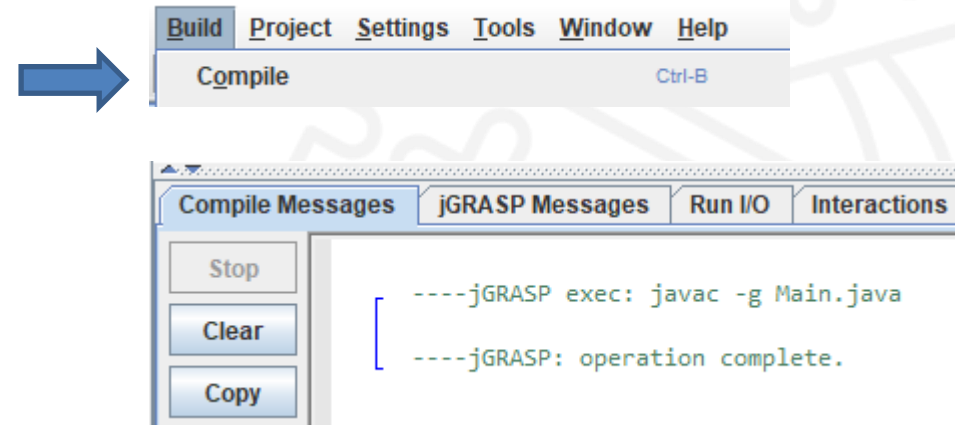


# Criando um projeto



Digite o seguinte programa

```
import java.io.*;
import java.util.*;
class Main {
    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }
}
```



# Altere o programa para

```
/* Um Primeiro Programa */  
import java.io.*;  
import java.util.*;  
class PrimeiroPrograma {  
    public static void main(String[] args)  
    {  
        System.out.println("Primeiro Programa em Java!");  
    }  
}
```

# Estrutura de um programa Java

// definições para pré-processamento

**class** <nome> {

// definições globais

// definições de armazenadores e métodos (*OPCIONAL*)

<tipo> <nome> (<lista de parâmetros>) {

// definições locais

// comandos

}

// parte principal

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

// definições locais

// comandos

}

} // fim da classe

```
/* Um Primeiro Programa */
import java.io.*;
import java.util.*;
class PrimeiroPrograma {
    public static void main(String[] args)
    {
        System.out.println("Primeiro Programa em Java!");
    }
}
```



# Primeiro Programa

```
/* Um Primeiro Programa */  
class PrimeiroPrograma {  
    public static void main(String[] args)  
    {  
        System.out.println("Primeiro Programa em Java!");  
    }  
}
```

## Comentando programas

Inserir comentários para documentar programas e aprimorar sua legibilidade. O compilador Java ignora os comentários e não realiza qualquer ação envolvendo os comentários.

Tipos de comentários:

// indica que será considerado comentário até o final da linha

O Java também possui comentários que se estendem por diversas linhas inicia-se com /\* e termina com \*/

# Primeiro Programa

```
/* Um Primeiro Programa */  
class PrimeiroPrograma  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Primeiro Programa em Java!");  
    }  
}
```

## Declarando uma classe

```
class PrimeiroPrograma
```

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** (PrimeiroPrograma). O nome da **classe** deve pertencer a um arquivo com um nome na forma NomeDaClasse.java, assim a **classe** PrimeiroPrograma é armazenada no arquivo **PrimeiroPrograma.java**

Por convenção, os nomes de classes iniciam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles incluem em maiúscula (por exemplo **PrimeiroPrograma**, **NumerosPrimos**).

A **{** da esquerda inicia a **classe**. Uma chave direita **}** correspondente deve terminá-la.

# Primeiro Programa

```
/* Um Primeiro Programa */  
class PrimeiroPrograma {  
    public static void main(String[] args)  
    {  
        System.out.println("Primeiro Programa em Java!");  
    }  
}
```

## Declarando um método

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

Método **main** inicia a execução do aplicativo Java, é o ponto de partida. Os parênteses depois do identificado **main** indicam que é um bloco de construção do programa denominado método.

Declarações de classe java normalmente contêm um ou mais métodos. Para um aplicativo Java, um dos métodos deve ser chamado **main**.

A **{** da esquerda inicia o corpo da declaração do método. Uma chave direita **}** correspondente deve terminá-la.

# Primeiro Programa

```
/* Um Primeiro Programa */  
class PrimeiroPrograma {  
    public static void main(String[] args)  
    {  
        System.out.println("Primeiro Programa em Java!");  
    }  
}
```

Gerando saída com System.out.println

```
System.out.println("Primeiro Programa em Java!");
```

Instrui ao computador executar uma ação de exibir os caracteres entre as aspas duplas. As aspas e os caracteres entre elas são uma **string** – também denominada de **string de caracteres** ou **string literal**. **System.out** (predefinido) é denominado como **objeto de saída padrão**. Permite que um aplicativo Java exiba informações na **janela de comando** a partir da qual ele é executado.

# Primeiro Programa

print e println

```
/* Um Primeiro Programa */
class PrimeiroPrograma {
    public static void main(String[] args){
        System.out.print("Primeiro ");
        System.out.println("Programa em Java!");
    }
}
```

```
----jGRASP exec: java PrimeiroPrograma
Primeiro Programa em Java!
----jGRASP: operation complete.
```

print e println

```
/* Um Primeiro Programa */
class PrimeiroPrograma {
    public static void main(String[] args){
        System.out.println("Primeiro ");
        System.out.println("Programa em Java!");
    }
}
```

```
----jGRASP exec: java PrimeiroPrograma
Primeiro
Programa em Java!
----jGRASP: operation complete.
```

# Primeiro Programa

Utilizando o printf

O método `System.out.printf` (f significa “formato”) exibe os dados formatados.

```
/* Um Primeiro Programa */
class PrimeiroPrograma {
    public static void main(String[] args)
    {
        System.out.printf("%s\n%s\n", "Primeiro", "Programa em Java!");
    }
}
```

```
----jGRASP exec: java PrimeiroPrograma
Primeiro
Programa em Java!
----jGRASP: operation complete.
```

A chamada do método `System.out.printf` especificou 3 argumentos, quando o método exige múltiplos argumentos, esses são colocados em uma **lista separada por vírgulas**. Chamar um método também é referido como **invocar** um método.

# Instruções de Programa

- Toda instrução em Java deve terminar com um **ponto-e-vírgula**.  
**Obs: toda instrução não quer dizer toda linha!!!**
- O **ponto-e-vírgula** é crucial da sintaxe da linguagem e facilmente esquecido o que acarreta a apresentação de um erro de compilação.
- Uma **função** pode ter qualquer número de instruções.
- As instruções devem ser escritas entre as **chaves** que delimitam o corpo da função e executadas na ordem em que as escrevemos.

# Instruções de Programa

## Letras Minúsculas

Em Java as letras maiúsculas e minúsculas são consideradas distintas.

Assim, a função principal que inicia o programa deve ser grafada em letras minúsculas.

Os nomes main, Main, mAaIN, MAIN, MAIn são diferentes.



## Exercício

Crie um novo programa denominado SegundoPrograma que escreva a mensagem:  
“segundo programa em Java”.

# Variáveis



## Linguagem Java

### Tipos de dados:

**Inteiro:** palavra reservada **int** (exemplos: 3, 6, 9, 27)

**Real:** palavra reservada **double** (exemplos: 3.1416, 8.8)

**Caractere:** palavra reservada **char** (exemplos: 'a', '8')

**Literal:** palavra reservada **String** (exemplos: "nome", "rua x")

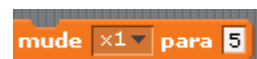
# Comandos básicos

## Linguagem Java

Comando de Atribuição. ,  
Para a atribuição de um valor a uma variável,  
símbolo =

Sintaxe: variável = expressão

### Exemplos:



x1 = 5;



a = 1.5;



soma = 0;



soma = soma + a;

ou



soma +=a;



# Comandos básicos - saída

## Linguagem Java



```
class ExemploSaida
{
    public static void main (String[] args)
    {
        int a;
        double b;
        char c;
        String nome;
        a=10;
        b=3.14;
        c='x';
        nome="Maria";
        System.out.println("Valor de a:"+a+".");
        System.out.printf("Valor de b:%4.2f.%n",b);
        System.out.println("Valor de c:"+c+".");
        System.out.printf("%s%s%s%n","Valor de nome ",nome,".");
    }
}
```

----jGRASP exec: java ExemploSaida  
Valor de a:10.  
Valor de b:3,14.  
Valor de c:x.  
Valor de nome Maria.  
----jGRASP: operation complete.

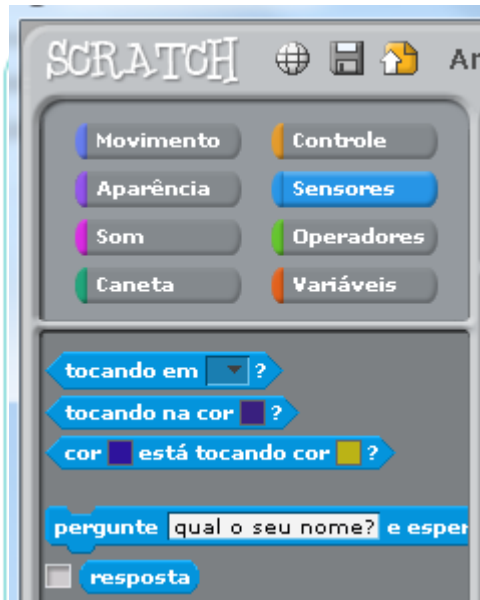
# Comandos básicos - entrada

## Linguagem Java

### Entrada de dados:

Incluir a classe **Scanner** no programa

Criar um objeto da classe **Scanner**



```
import java.util.Scanner;
public class ExemploEntrada
{
    public static void main (String[] args)
    {
        Scanner leia = new Scanner (System.in);
        int a;
        double b;
        String nome;
        String linha;
        System.out.println("Informe um valor para a:");
        a= leia.nextInt();
        System.out.println("Informe um valor para b:");
        b= leia.nextDouble();
        System.out.println("Informe um valor para nome:");
        nome= leia.next();
        leia.nextLine();// limpar o buffer de entrada
        System.out.println("Informe uma linha:");
        linha= leia.nextLine();
        System.out.println("Valor de a:"+a+".");
        System.out.printf("Valor de b:%4.2f.%n",b);
        System.out.printf("%s%s%s%n","Valor de nome ",nome,".");
        System.out.printf("valor de linha:"+linha);
    }
}
```

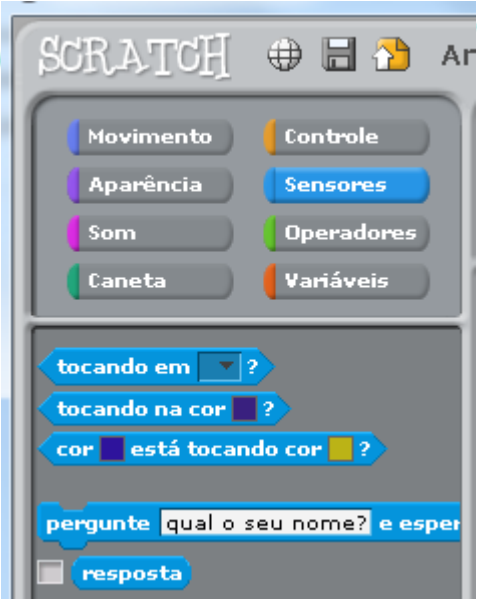
```
----jGRASP exec: java ExemploEntrada
Informe um valor para a:
10
Informe um valor para b:
15,87
Informe um valor para nome:
josé
Informe uma linha:
linha completa
Valor de a:10.
Valor de b:15,87.
Valor de nome josé.
valor de linha:linha completa
----jGRASP: operation complete.
```

# Comandos básicos - entrada

## Linguagem Java

### Entrada de caractere:

Diferentemente do que ocorre com os tipos base `int` e `double`, a classe `Scanner` não oferece um método específico para a leitura de dados do tipo caractere (`char`). Assim, para ler um caractere deve-se utilizar o método `read()` da classe `System` através do fluxo de entrada de dados padrão `System.in`



```
import java.io.IOException;
public class EntradaCaractere
{
    public static void main (String[] args) throws IOException
    {
        char c;
        System.out.println("Digite um caractere:");
        // lendo um caractere
        c= (char)System.in.read(); // tentar ler um caractere
        System.out.println("Valor atual de c:"+c);
    } // fim main()
} // fim classe
```

The image shows the Scratch interface on the left with input-related blocks like 'when clicked', 'when color clicked', 'when flag clicked', 'ask for name', and 'wait for answer'. A blue arrow points from the 'ask for name' block to the Java code. Another blue arrow points from the text 'Diferentemente do que ocorre com os tipos base int e double...' to the 'throws IOException' line in the code.

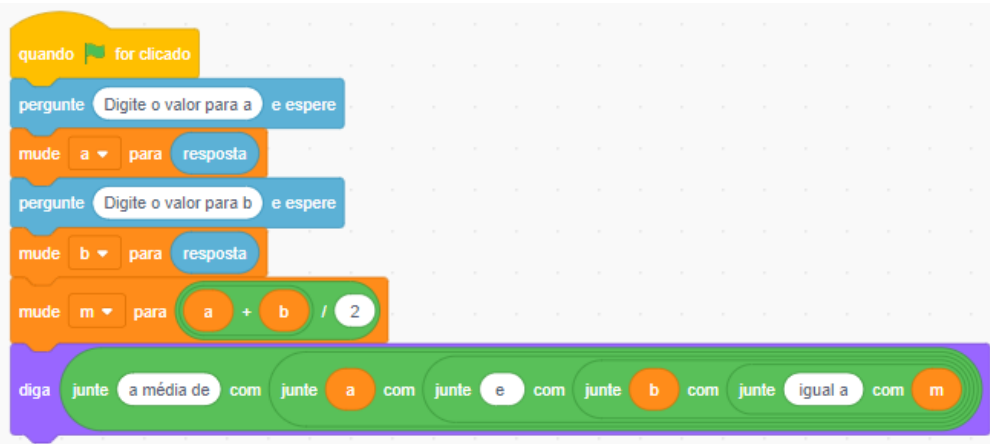
```
----jGRASP exec: java EntradaCaractere
Digite um caractere:
Valor atual de c:
----jGRASP: operation complete.
```

A blue arrow points from the 'Valor atual de c:' line in the terminal output to the corresponding line in the Java code.

# Comandos básicos – entrada / saída

## Linguagem Java

### Calcular a média entre 2 valores



```
----jGRASP exec: java CalculaMedia
>> Digite o valor para a:3
>> Digite o valor para b:8
    A media de 3 e 8 igual a 5.5
    A media de 3 e 8 igual a 5,50

----jGRASP: operation complete.
```

```
import java.util.*;
/*
  Autor:
  Data:
  Programa para calcular a média entre dois valores
*/
public class CalculaMedia
{
    public static void main (String[] args)
    {
        Scanner leia = new Scanner (System.in);
        int a,b;
        double m;
        System.out.print("Digite o valor para a:");
        a=leia.nextInt();
        System.out.print("Digite o valor para b:");
        b=leia.nextInt();
        m=(a+b)/2.0;
        System.out.printf("A media de "+a+" e "+b+" igual a "+m);
        System.out.printf("\nA media de %2d e %2d igual a %5.2f\n",a,b,m);
    }
}
```

# Exercícios de fixação:

1. Escreva um programa que solicite ao usuário a altura e o raio de um cilindro circular e imprima o volume do cilindro.

O volume de um cilindro circular é calculado por meio da seguinte fórmula:

$$\text{Vol} = \text{PI} * \text{raio}^2 * \text{altura}$$

Obs: em algoritmo o operador de potência é o ^ em Java utiliza-se a função `Math.pow(a,b)`. exemplo  $a^2$  `a^2`  
`Math.pow(a,2)`

2. Uma empresa contrata um encanador a R\$ 20.00 por dia. Crie um programa que solicite o número de dias trabalhados pelo encanador e imprima a quantia líquida que deverá ser paga, sabendo-se que são descontados 8% de impostos.
3. O cardápio de uma lanchonete é dado abaixo. Prepare um programa que leia seu nome e a quantidade de cada item que você consumiu e calcule a conta final.

Hambúrguer..... R\$ 30,00

Cheeseburger..... R\$ 35,50

Fritas..... R\$ 20,50

Refrigerante..... R\$ 10,00

Milkshake..... R\$ 30,00





**PUC Minas**  
**Virtual**