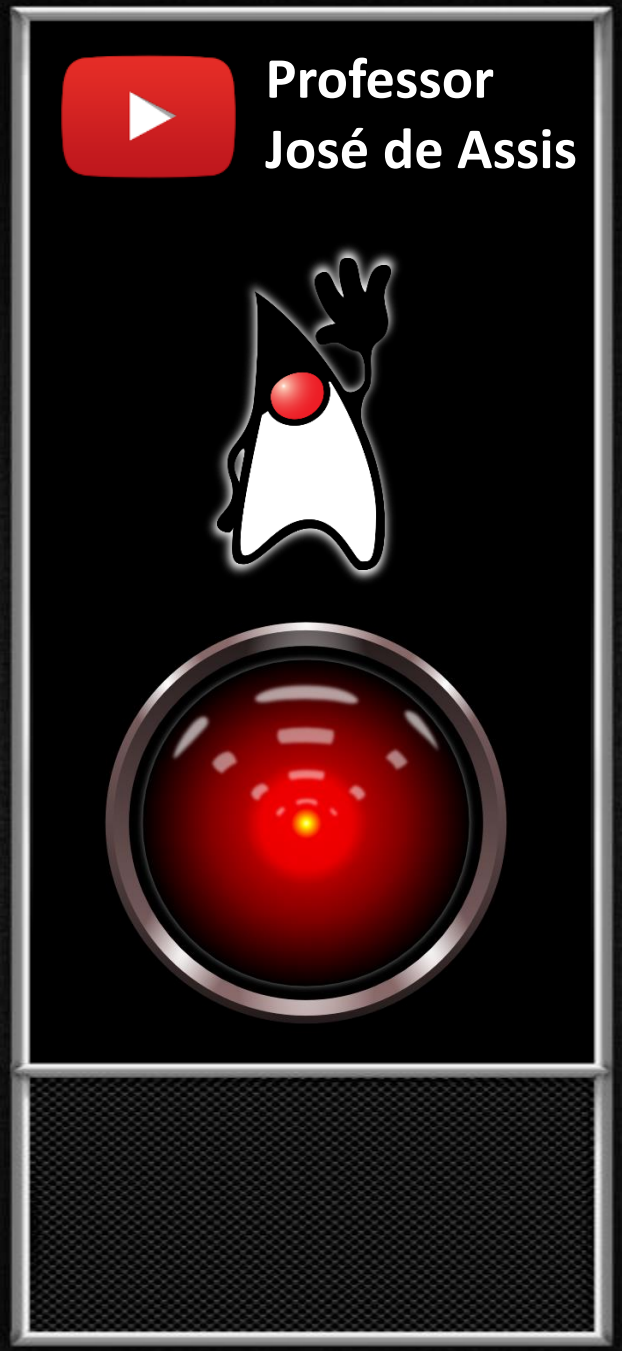
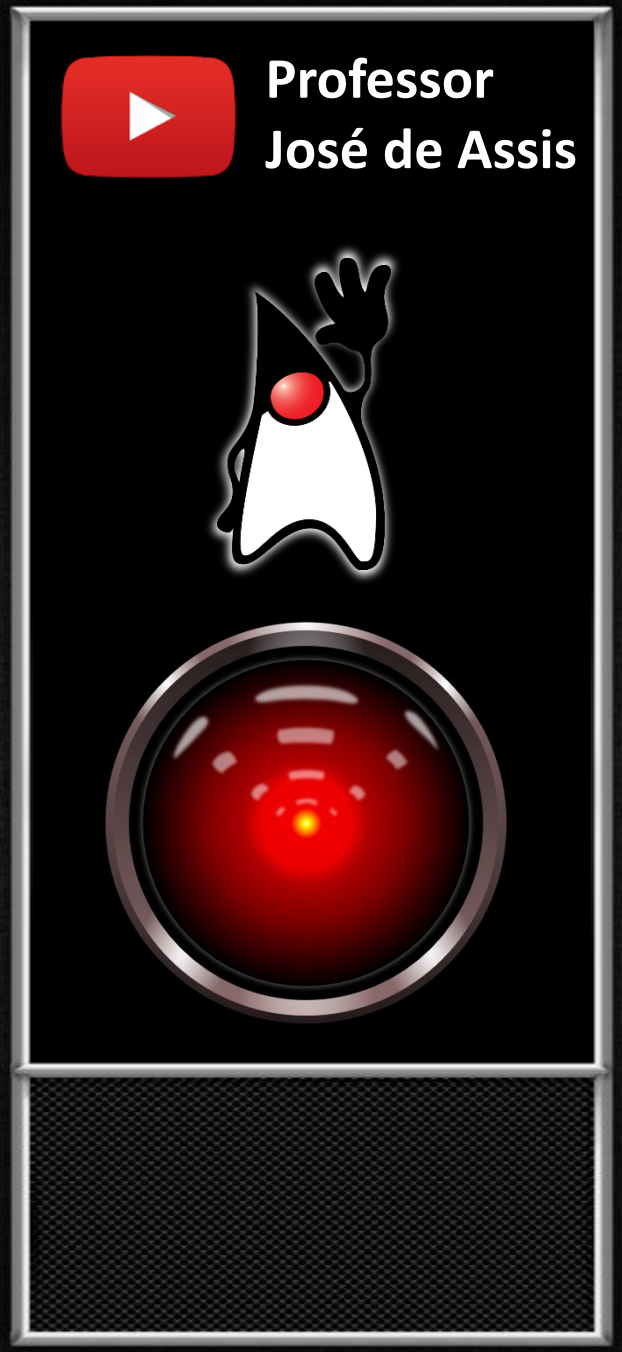
Java



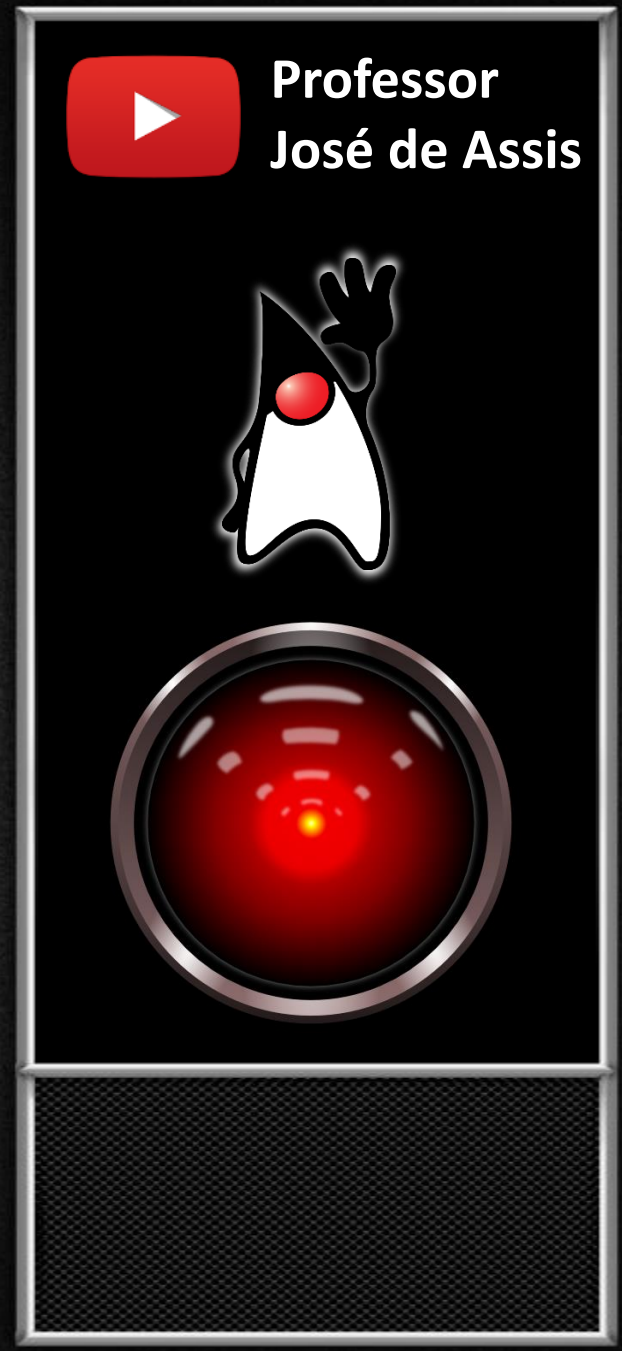
Um pouco da história – 1990

A ideia inicial era desenvolver um dispositivo wireless que usava uma tela colorida sensível ao toque para controlar a televisão, o som, alarmes e diversos dispositivos domésticos, além de interagir com computadores. A linguagem **Java** foi desenvolvida especialmente para este dispositivo.



Um pouco da história - 1995

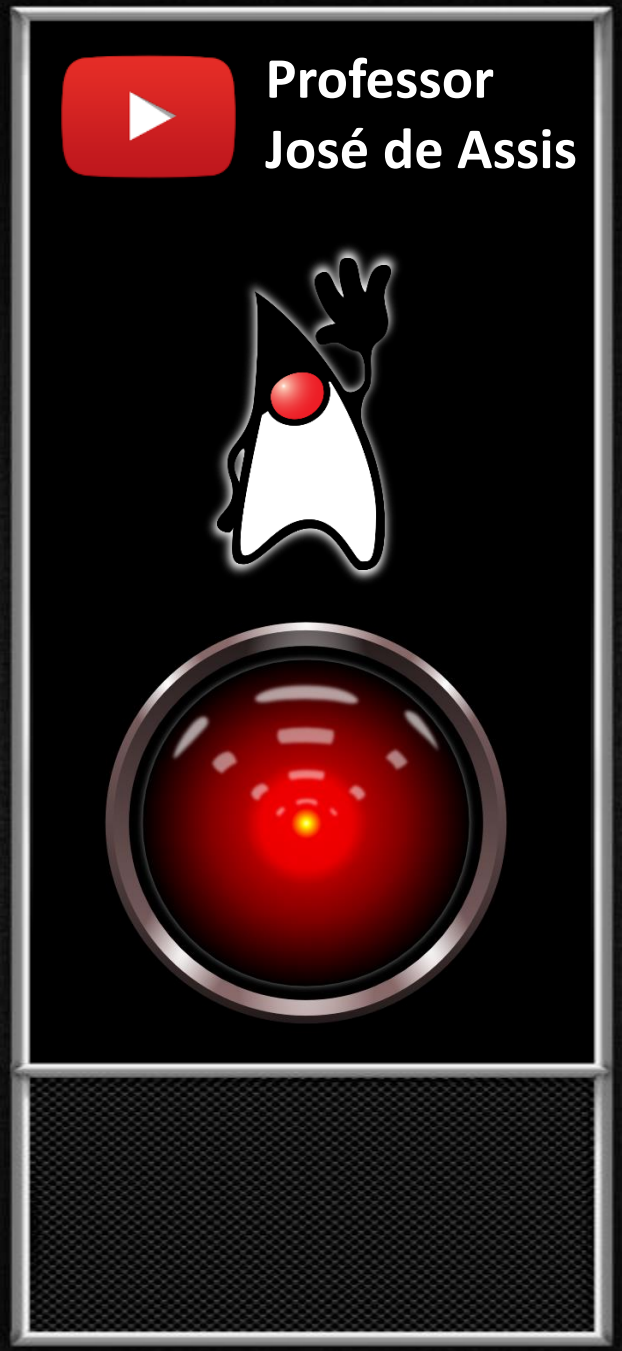
Por estar muito a frente da época, este projeto não foi adiante, o time então voltou a procurar alternativas para uso desta tecnologia. A equipe decidiu entrar no mundo da internet, criando um navegador capaz de suportar a linguagem **Java**. Este navegador ficou conhecido como **hotJava**.

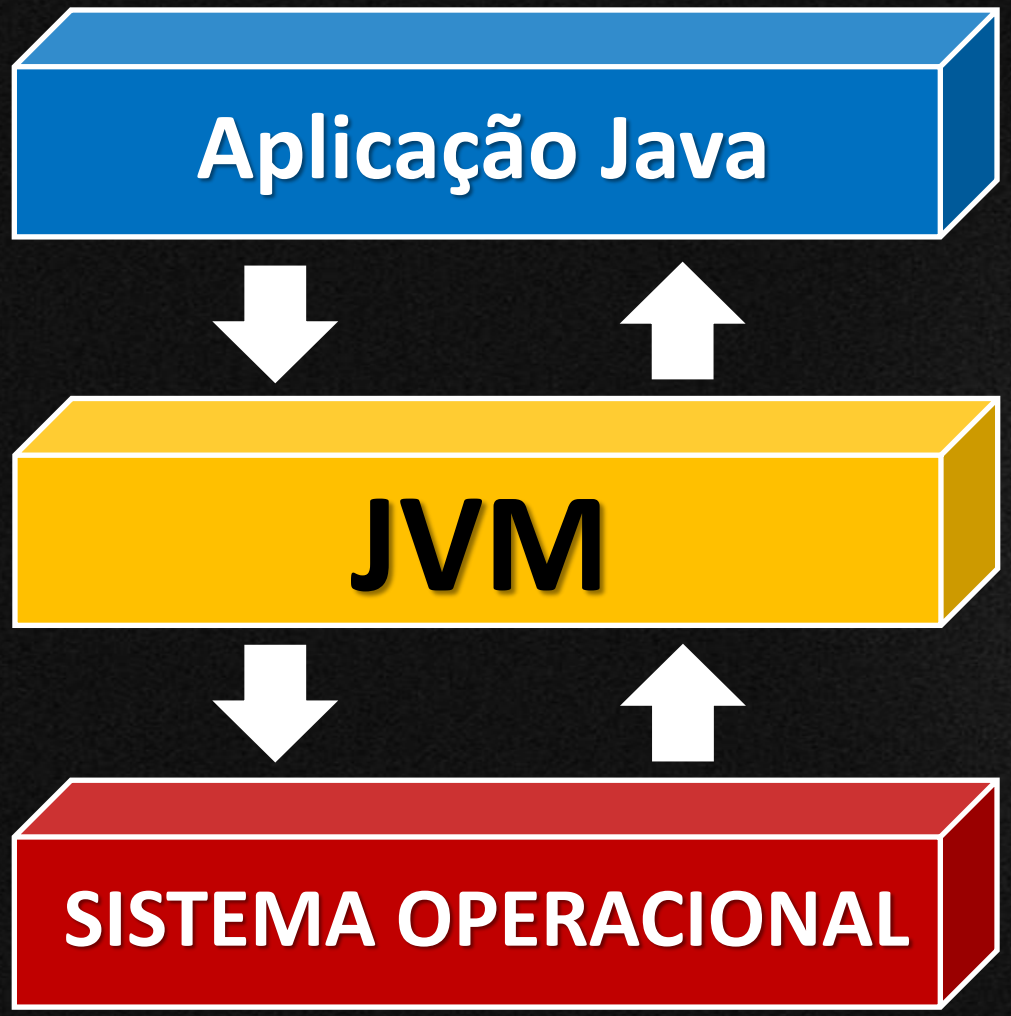


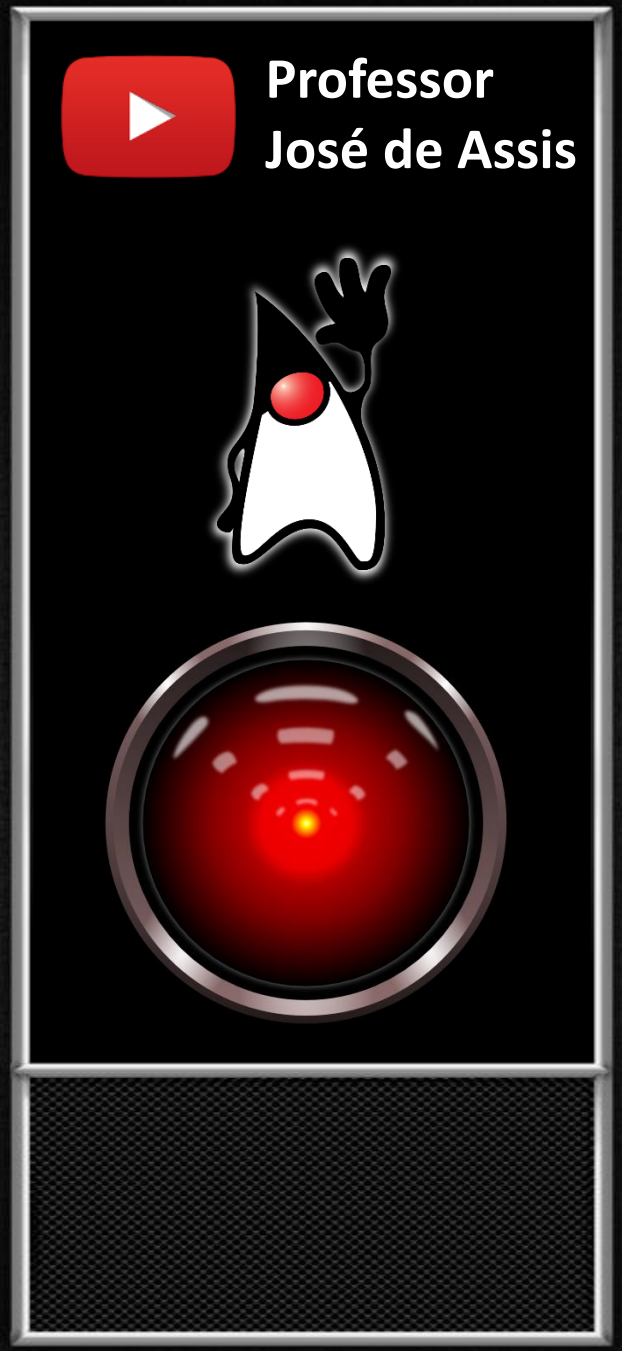
WORA

"Write once, run anywhere"

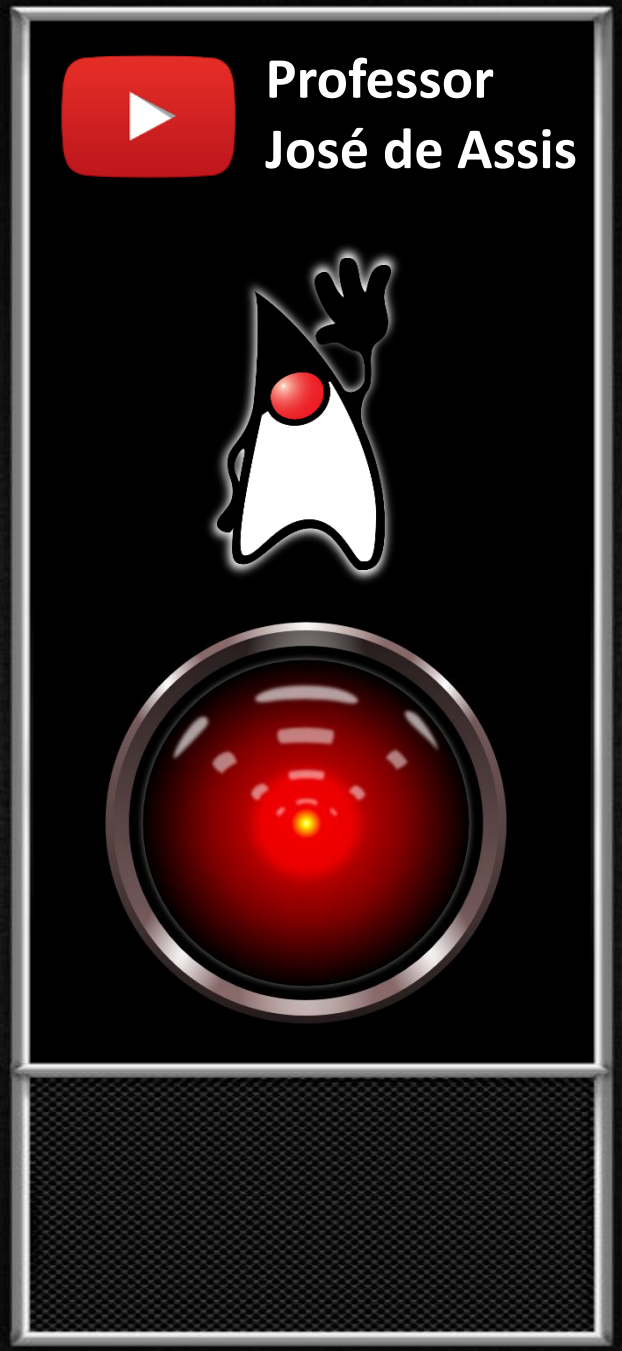
“Escreva uma vez, rode em qualquer lugar”. Este é o slogan do Java e representa a sua maior vantagem que é a portabilidade. Você escreve o código em Java e pode executar no Windows, no Linux, no MAC, no sistema Android, em Smartcards etc.



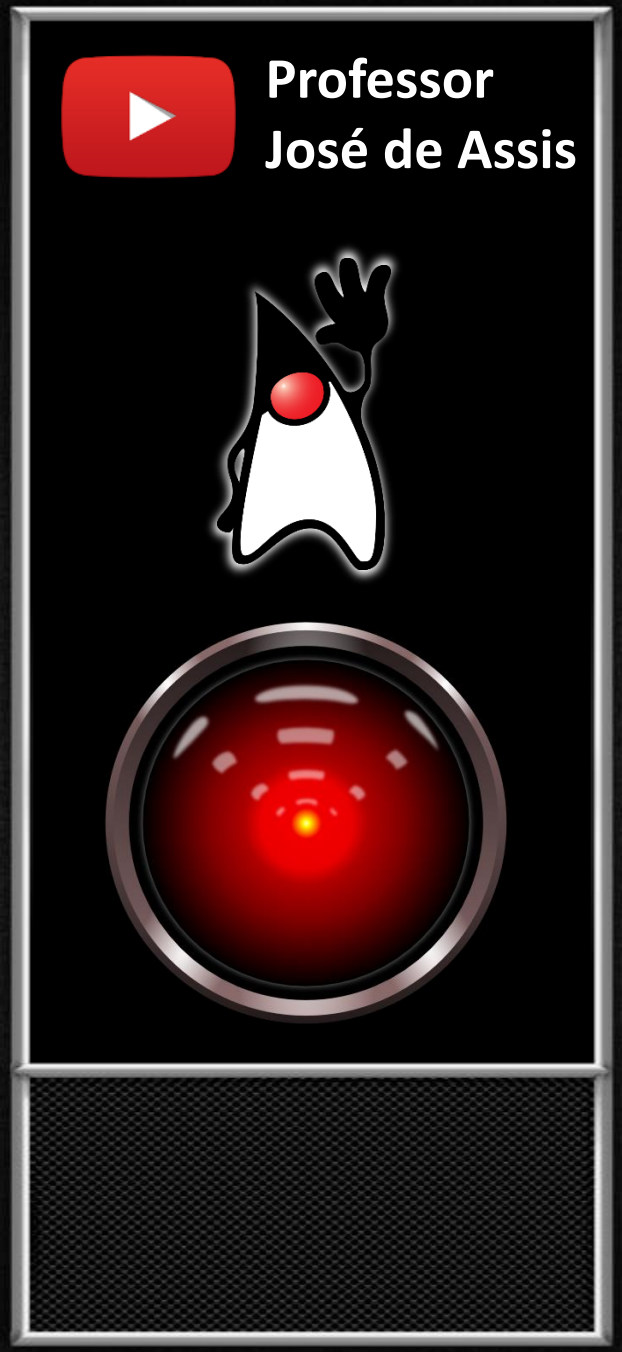

Aplicação Java
JVM
SISTEMA OPERACIONAL


Professor
José de Assis

Popularidade



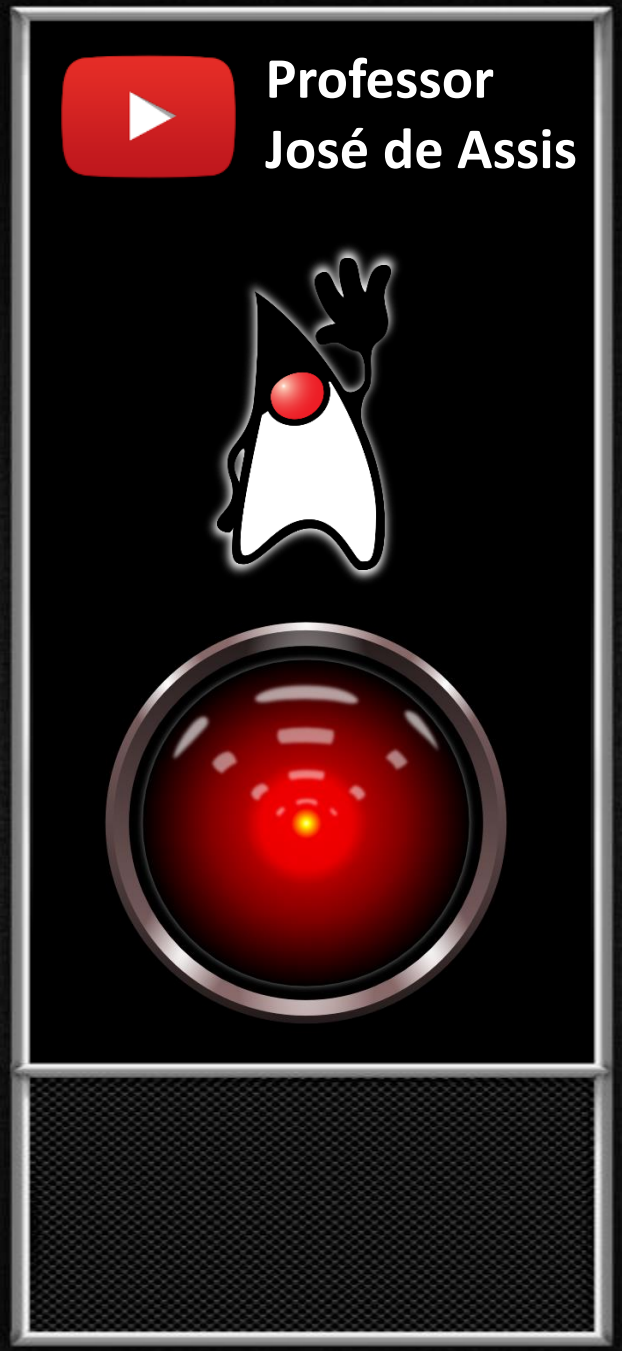
Quem usa Java?



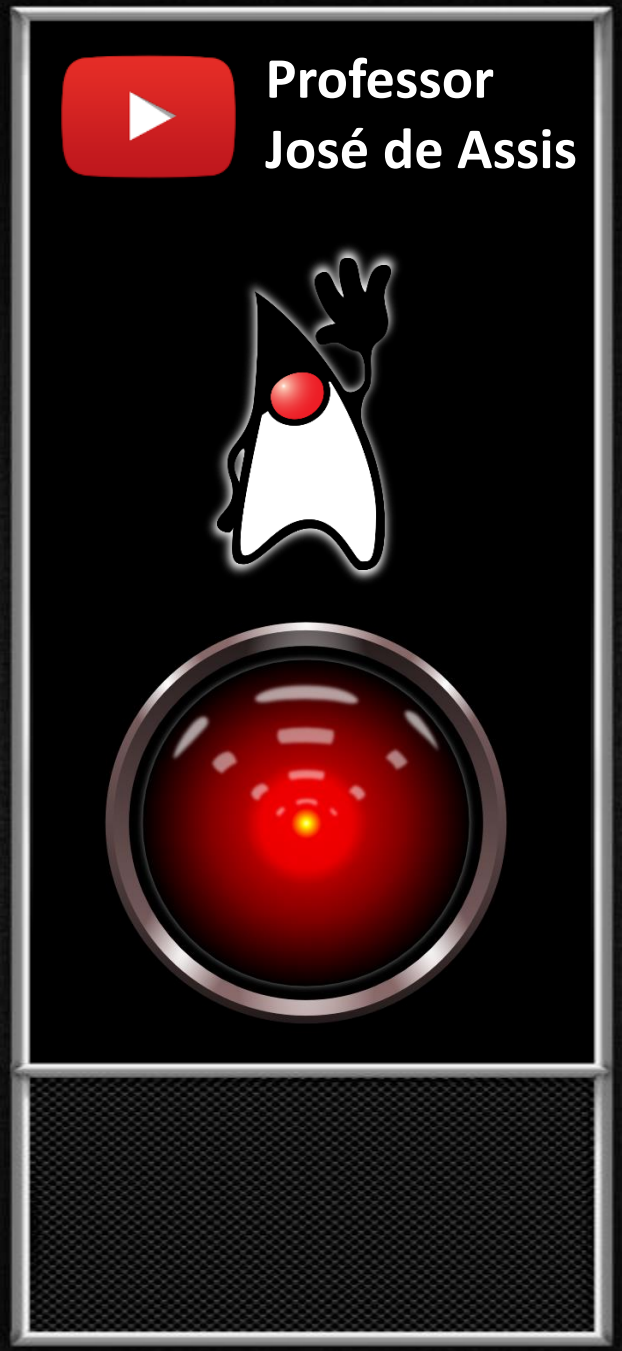
Back-end

Servlet

A linguagem **Java** é muito usada no mercado corporativo para o desenvolvimento de aplicações WEB no back-end.



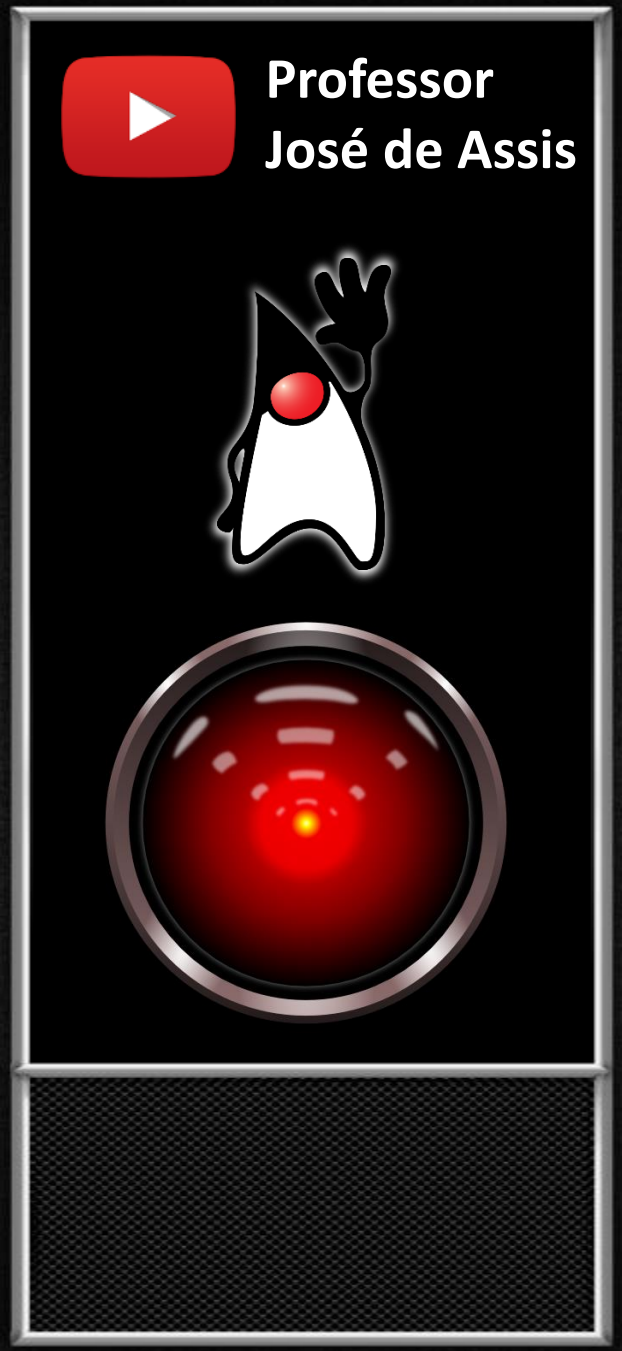
Android



A linguagem Java é usada para desenvolvimento de aplicações nativas do sistema Android.

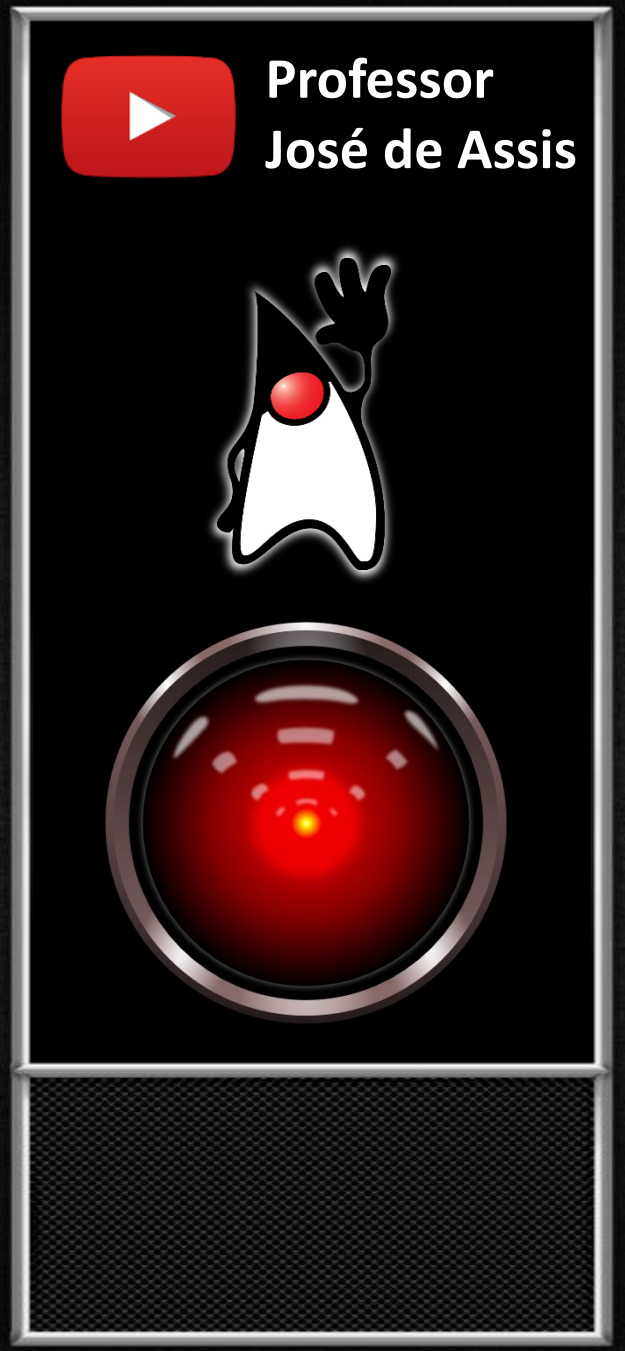
Sistema Desktop

O programa de declaração de imposto de renda é um exemplo de sistema desktop desenvolvido em Java.

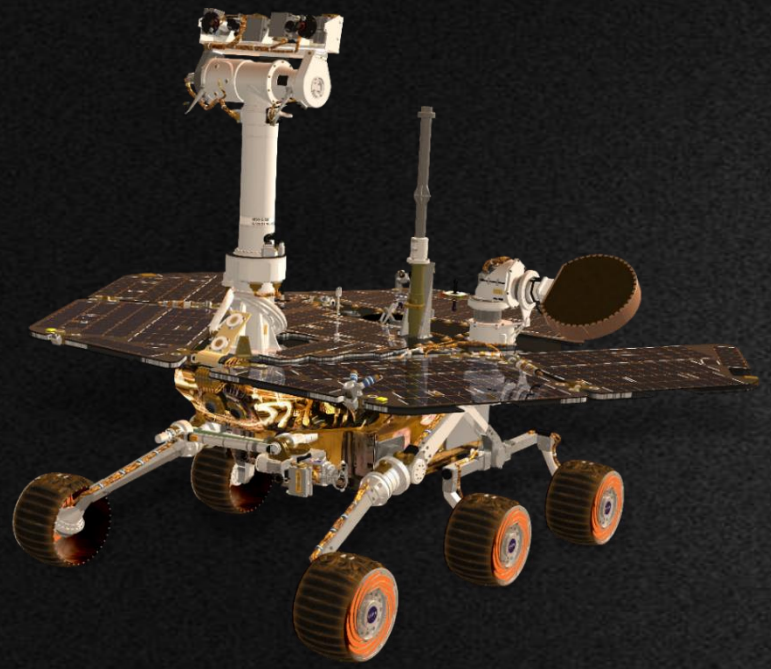


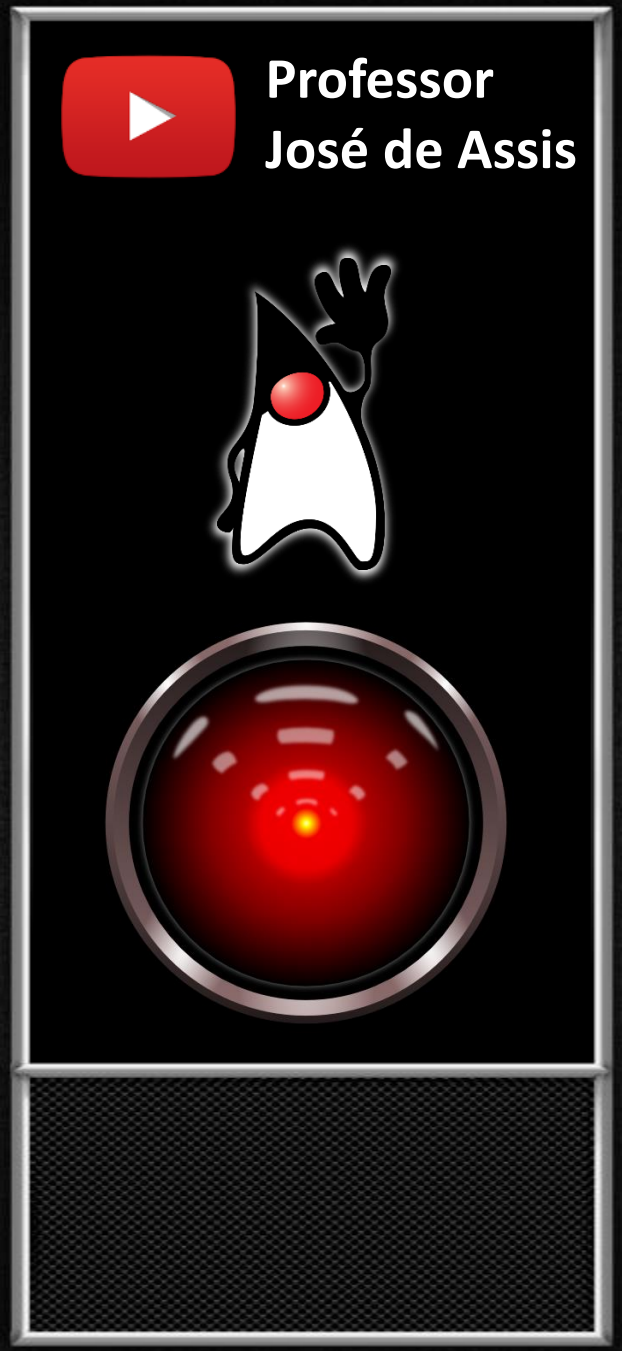
Games

Minecraft é um exemplo de jogo desenvolvido com a linguagem Java.



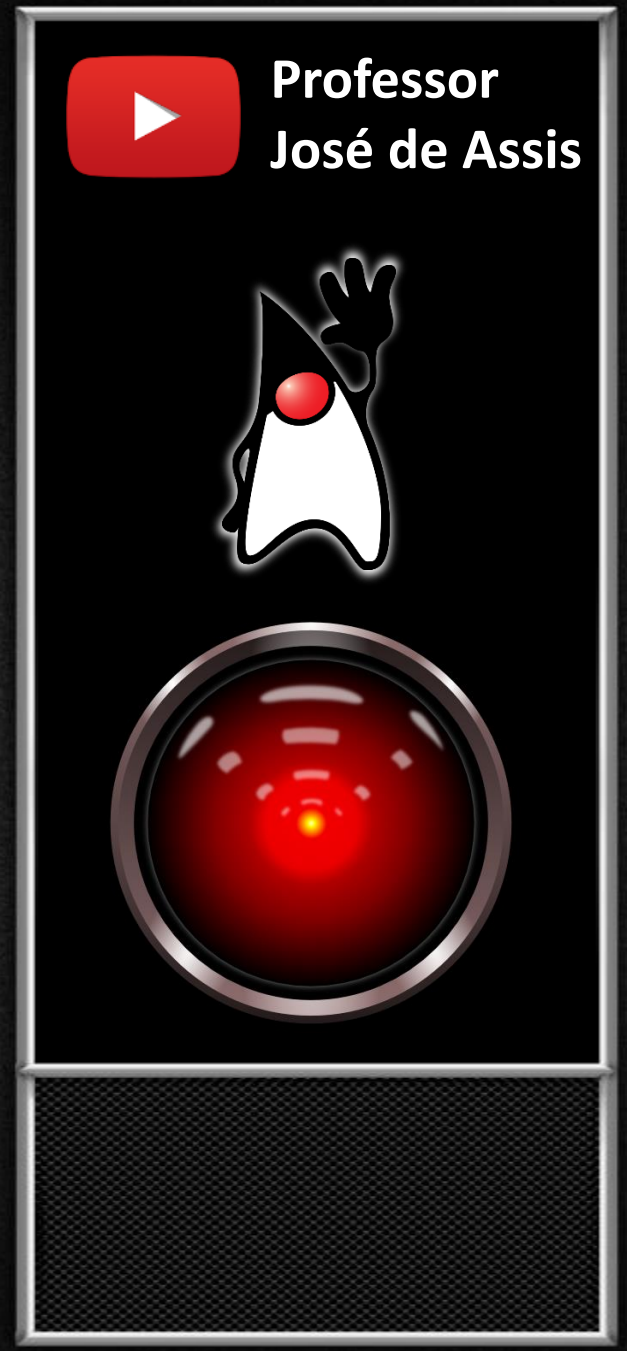
Outros





O Java também é encontrado em Smartcards e em diversos outros dispositivos como por exemplo sistemas de som, e até mesmo na robótica.

Por onde começar ?

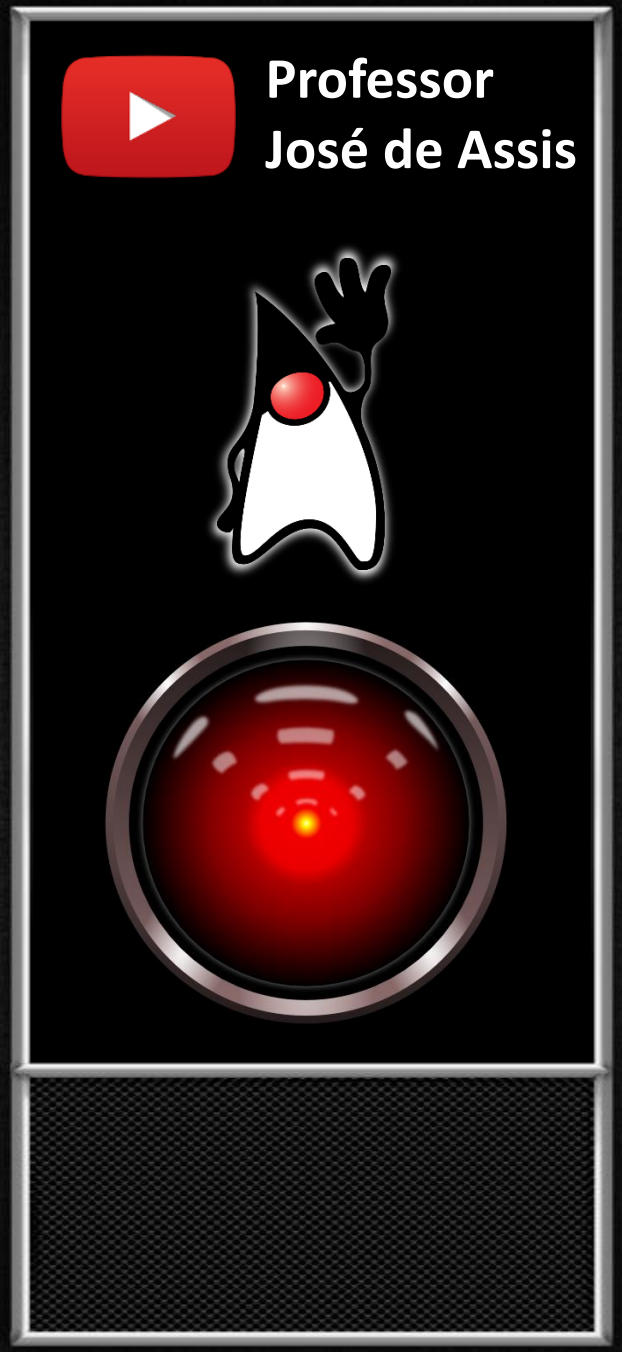


Para começar a desenvolver o Java é necessário instalar o kit do desenvolvedor Java, também conhecido como **JDK**.

IDE

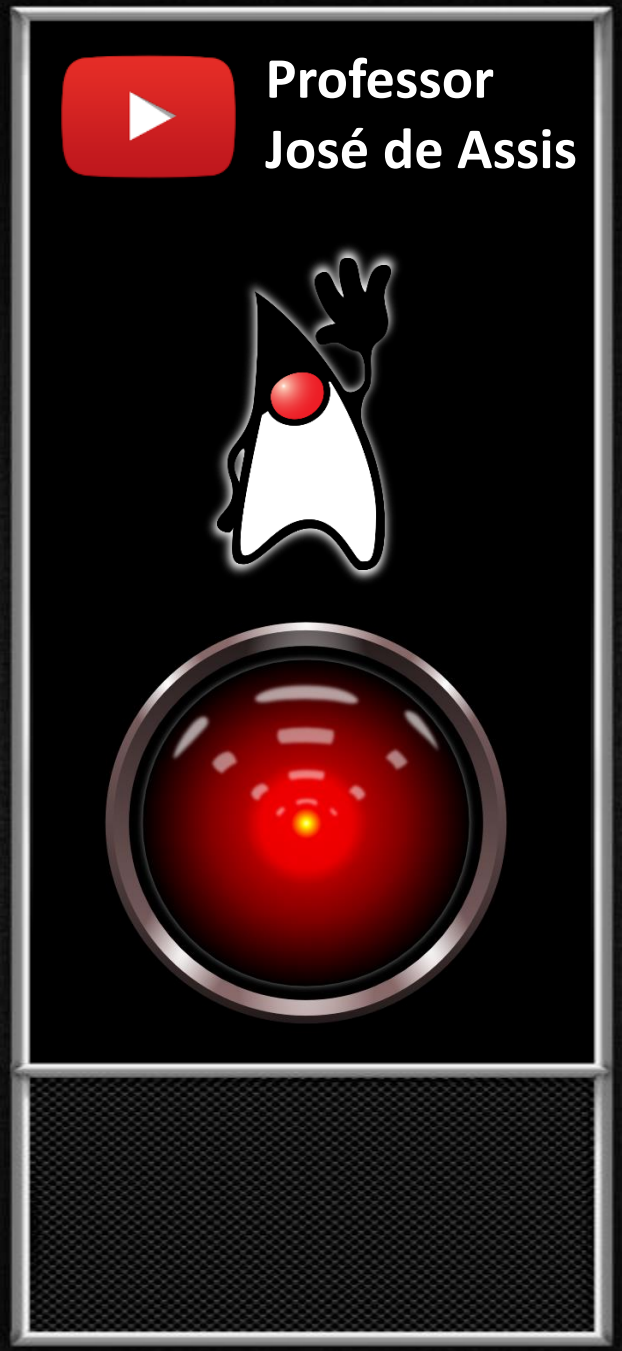
Netbeans

Eclipse



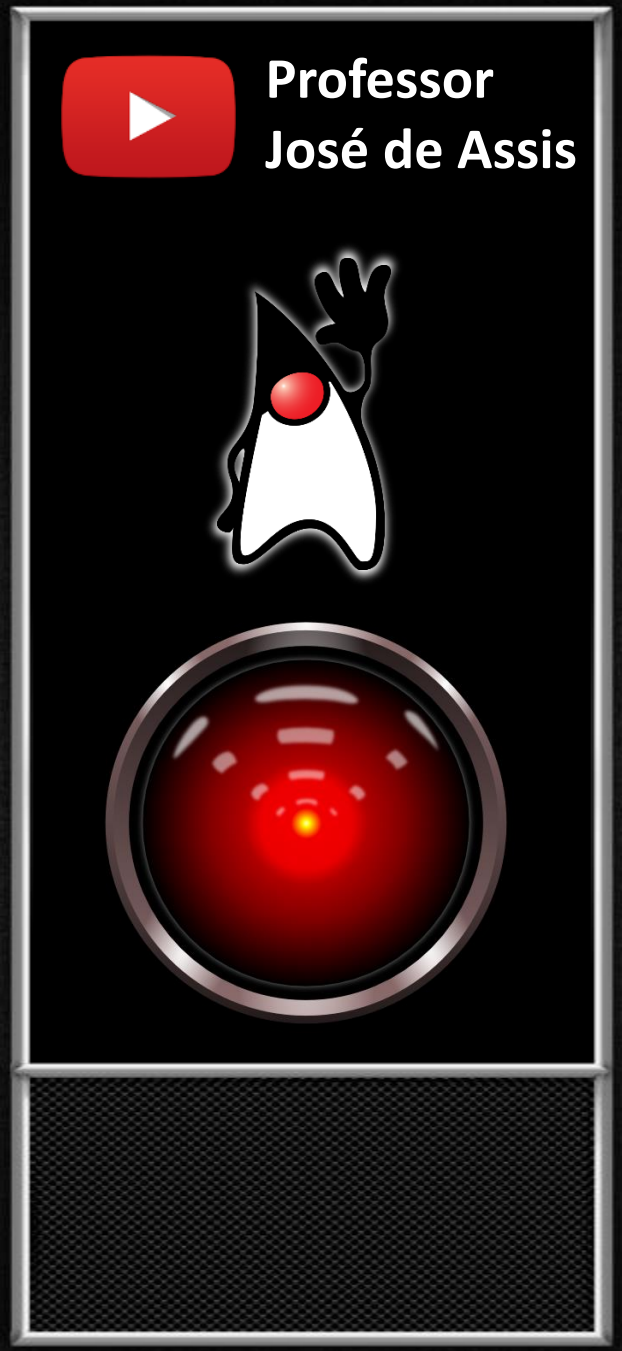
Para começar a desenvolver o Java, além do **JDK** é necessário também instalar uma **IDE**. A IDE é um ambiente integrado de desenvolvimento que contém um editor de código, um compilador e um depurador.

Fundamentos do Java



VARIÁVEIS

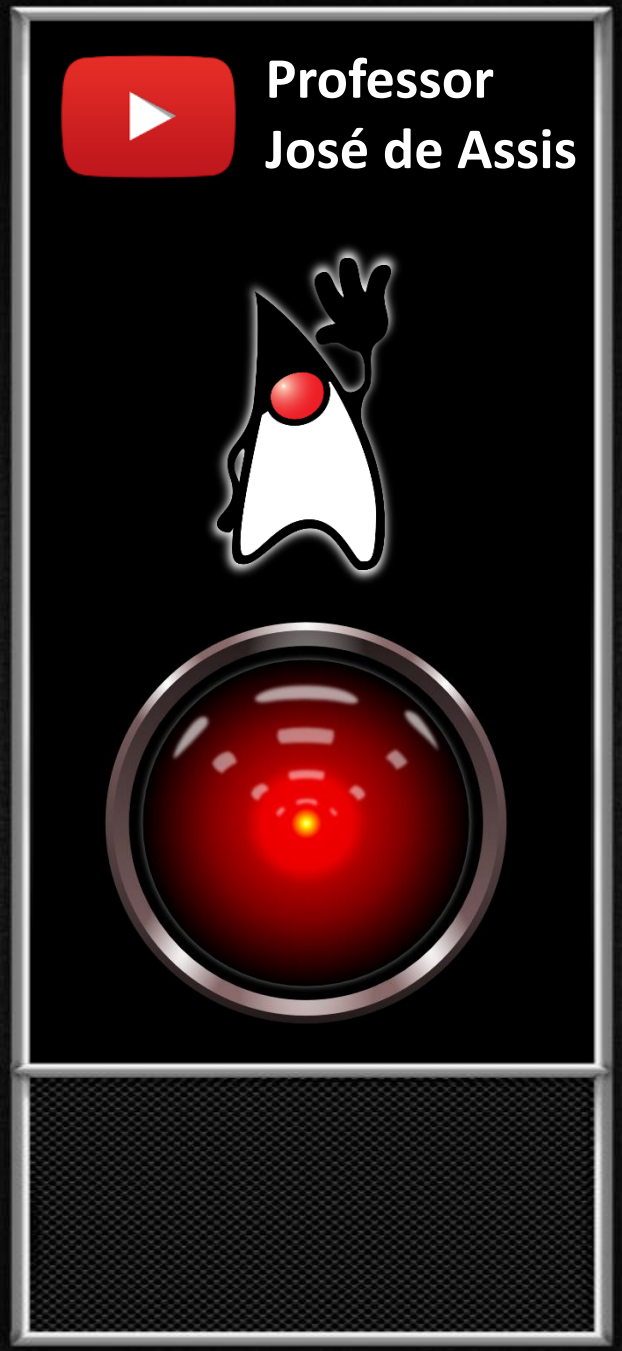
Na programação utilizamos as variáveis para armazenar dados na memória. Estes dados podem ser alterados de acordo com o tempo, como por exemplo o armazenamento do valor da idade.



Tipos de variáveis usadas no Java

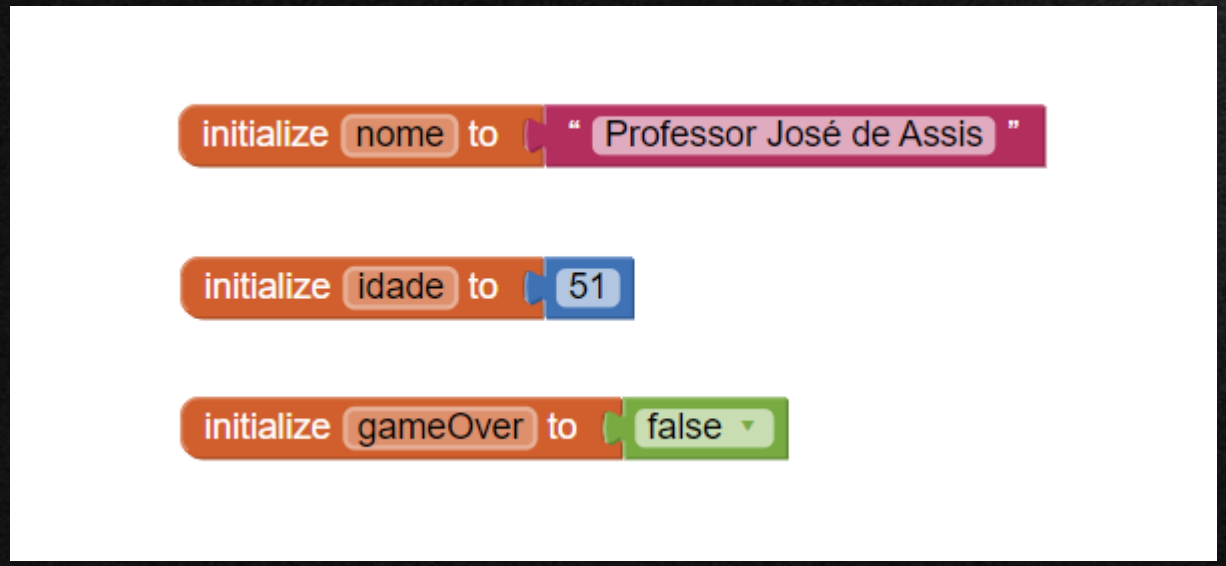
Grupos	Tipos primitivos	Tamanho	Valores válidos
Inteiros (números sem casas decimais)	**byte**	1 byte	-128 até 127
	short	2 bytes	-32768 até 32767
	int	4 bytes	-2147483648 até 2147483647
	long	8 bytes	-9223372036854775808L até 9223372036854775808L
Flutuantes (números com casas decimais)	**float**	4 bytes	±3.40282347E+38F
	double	8 bytes	±1.79769313486231570E+308
Booleanos (verdadeiro ou falso)	**boolean**	1 byte	true ou false
Caractere (aceita apenas um caractere)	**char**	2 bytes	1 caractere da tabela Unicode (Ex: 'a')

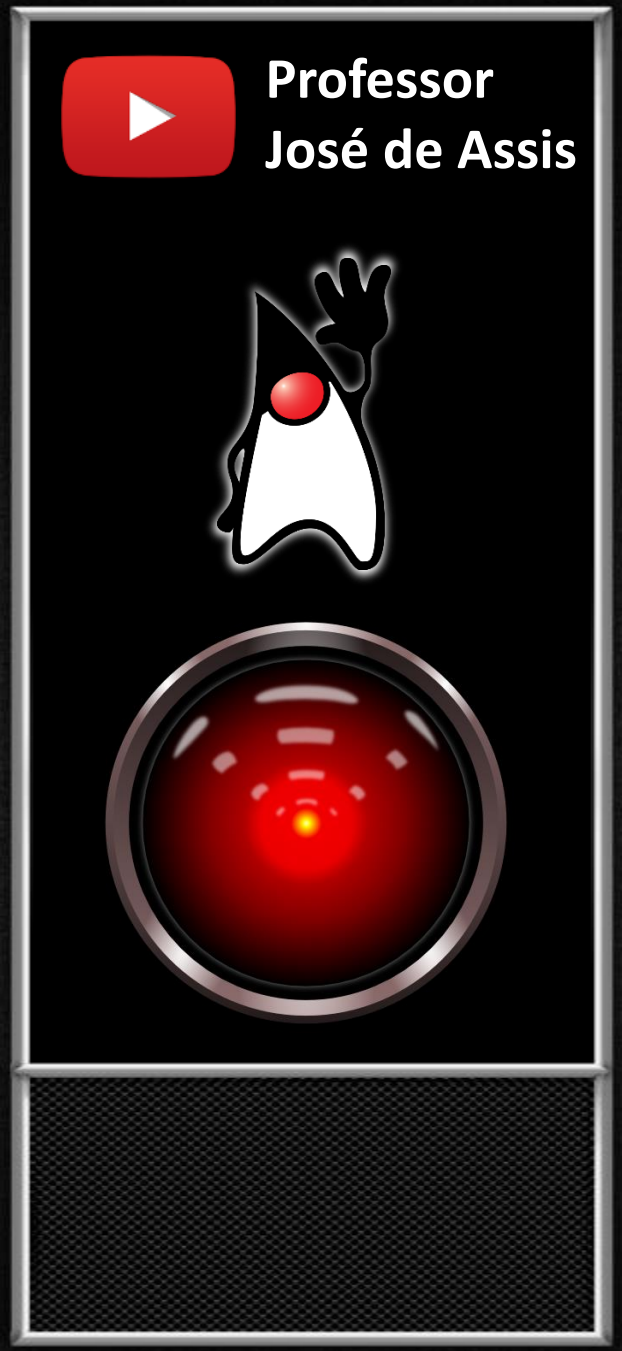
Tipos não primitivos	Descrição
String	Uma cadeia de caracteres Unicode
Object	Um objeto genérico



Criando variáveis no Java

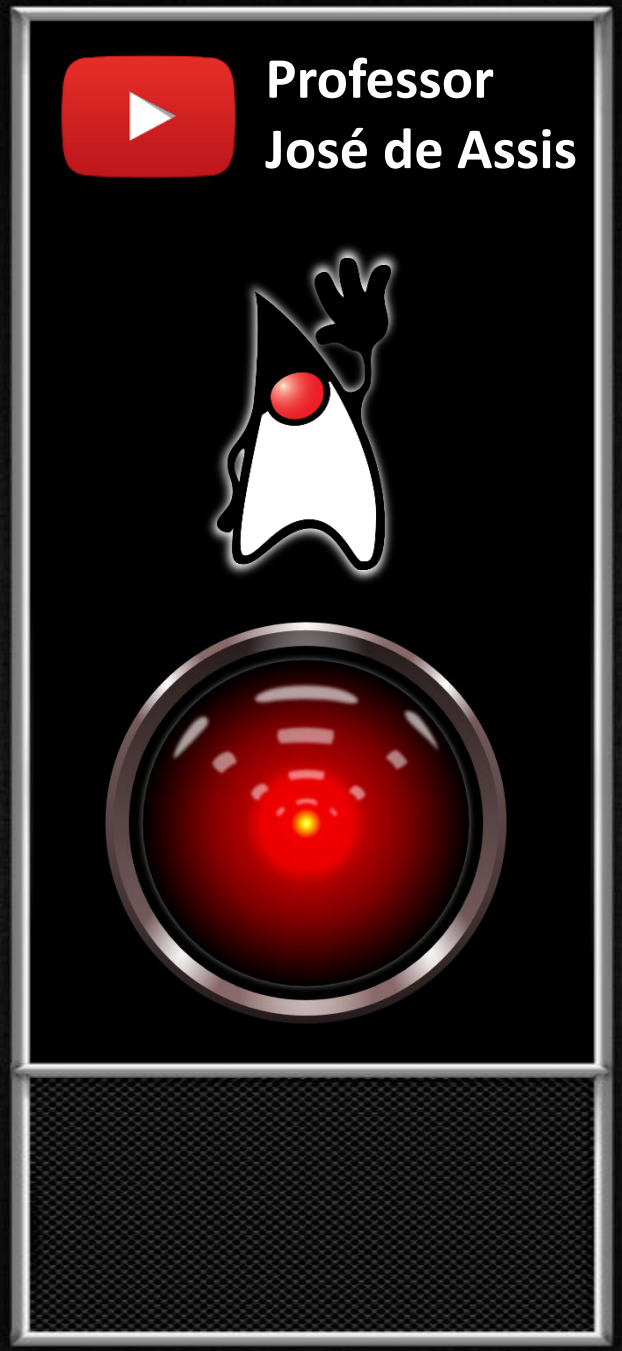
Na linguagem Java devemos atribuir o tipo de variável de acordo com os valores que serão armazenados. No Java uma variável também não pode ser usada se não for inicializada.



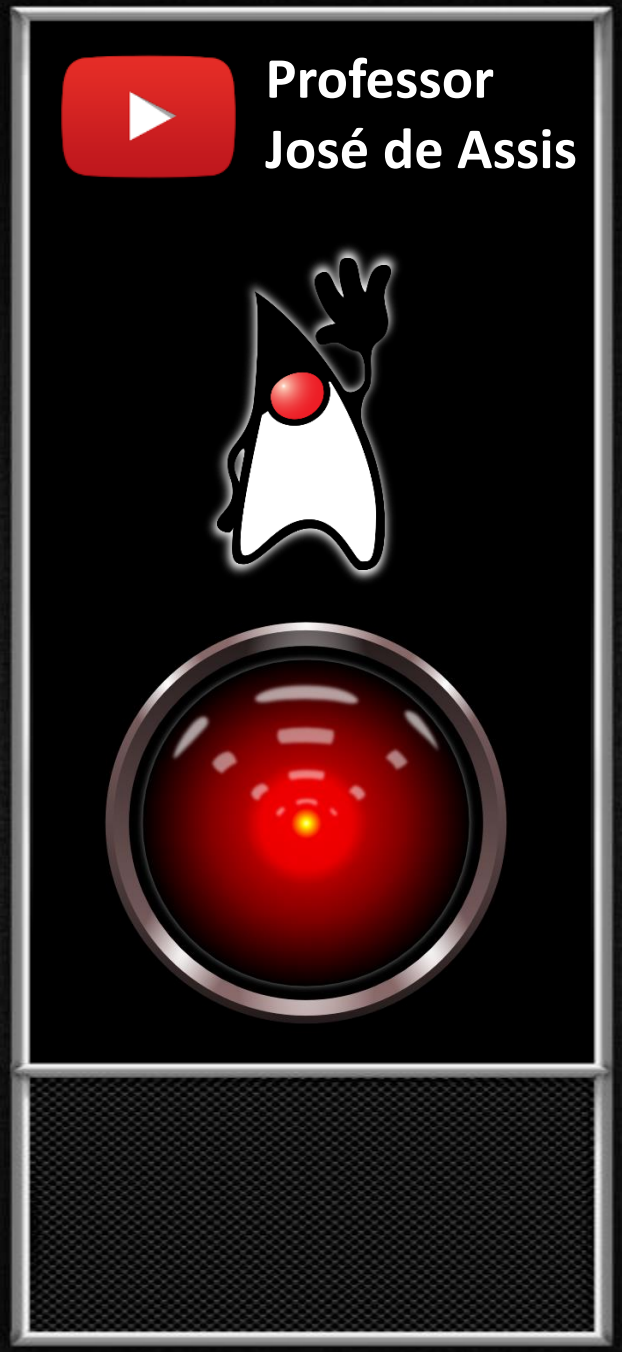


Nomeando variáveis

- O nome da variável deve começar com uma letra minúscula
- Não usar caracteres especiais nem deixar espaços
- Não usar palavras chaves ou reservadas da linguagem de programação
- Definir nomes que descrevam o tipo de informação que as variáveis irão armazenar (clean code)

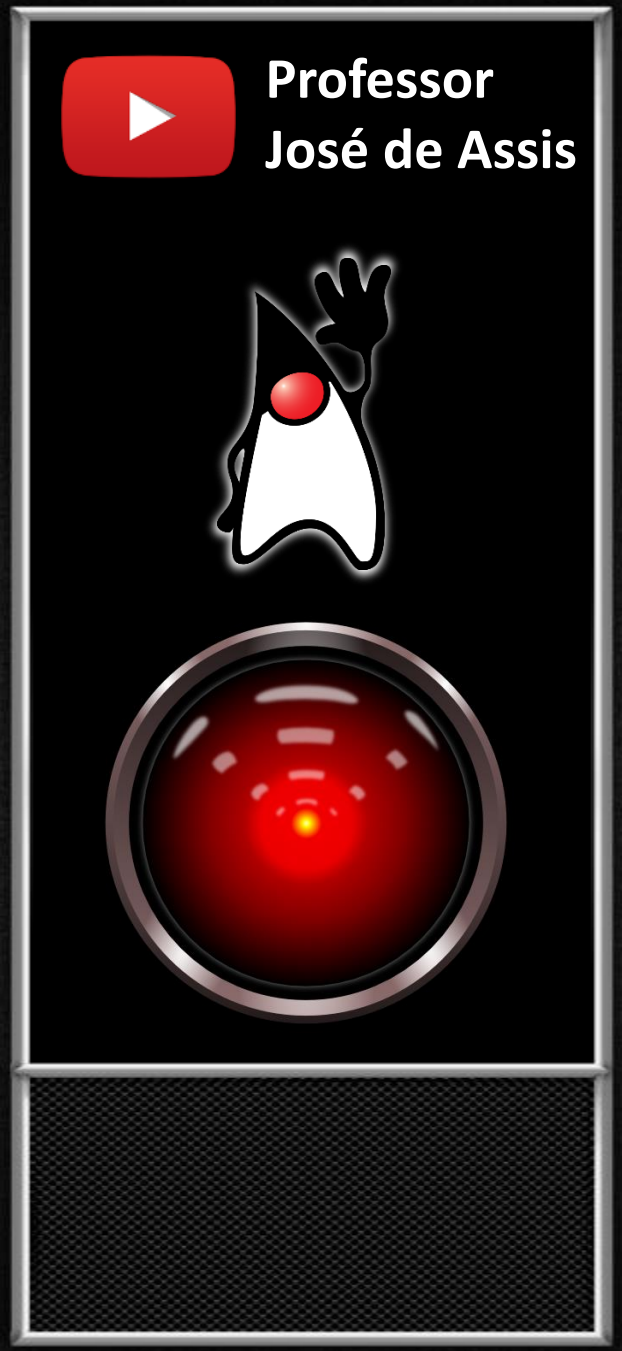


Operadores



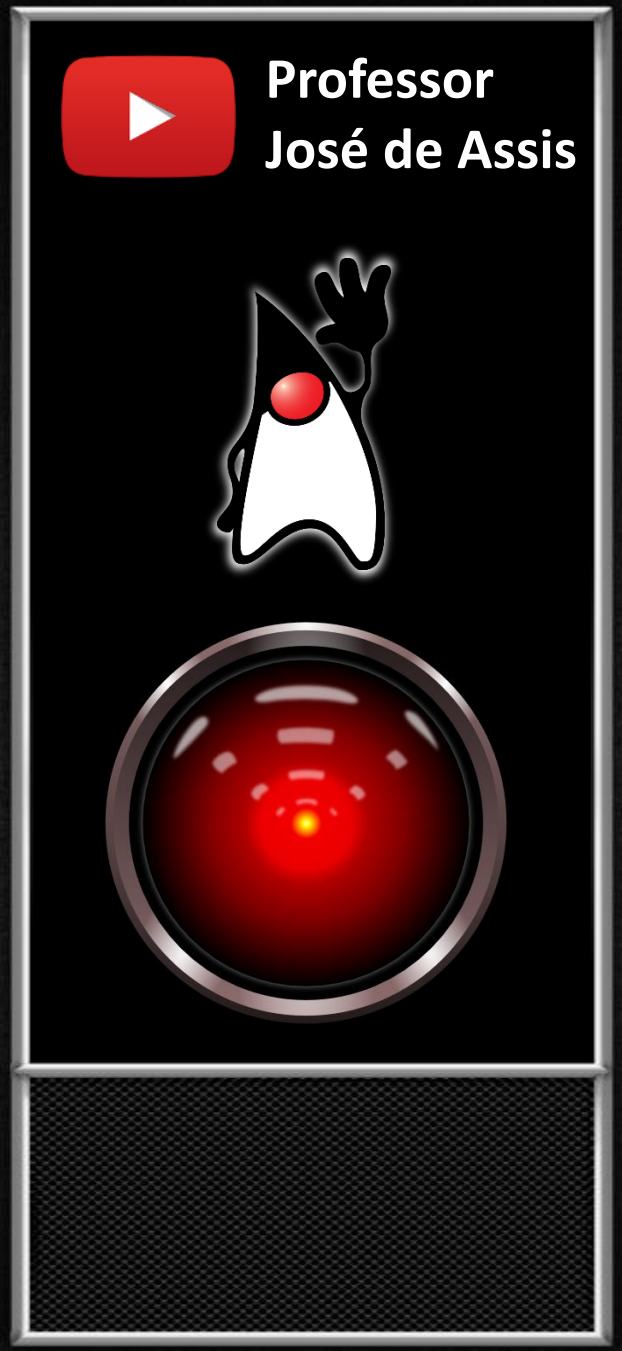
Operadores Aritméticos

Operador	Propósito	Exemplo	Resultado
=	“Setar” uma variável	i = 10	i = 10
+	Soma	i = 10 + 5	i = 15
-	Subtração	i = 10 - 5	i = 5
*	Multiplicação	i = 10 * 5	i = 50
/	Divisão	i = 10 / 5	i = 2
%	Resto da divisão	i = 10 % 5	i = 0



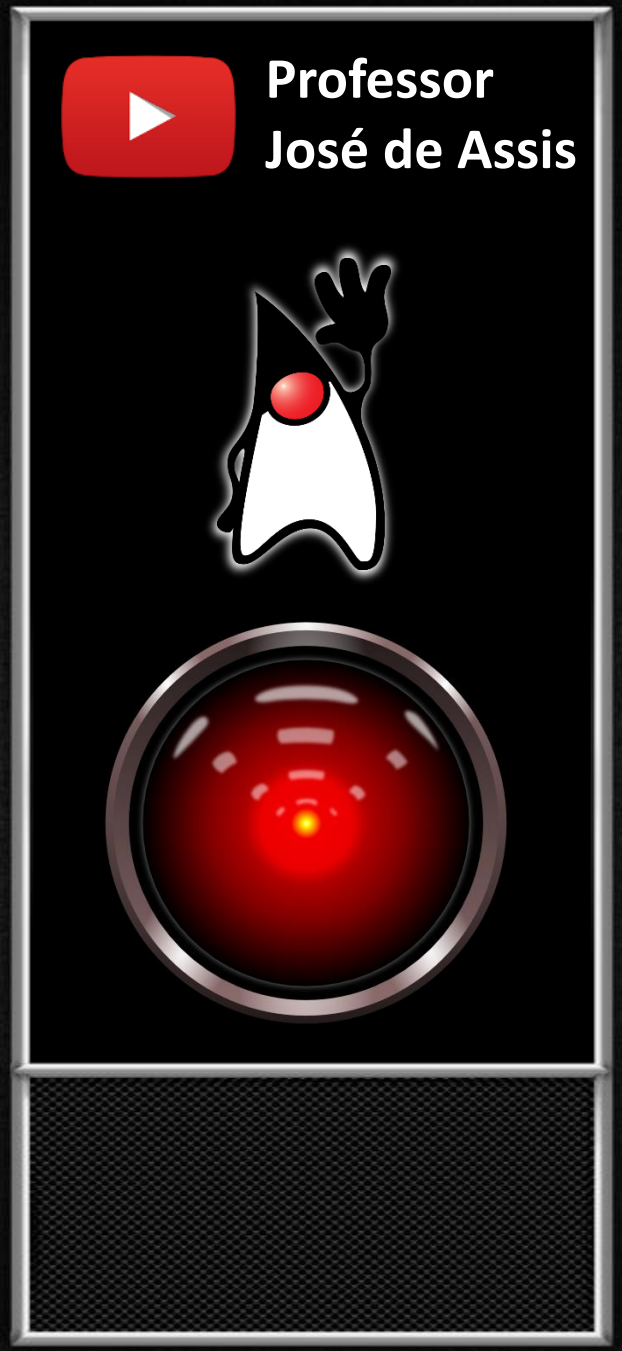
Operadores Aritméticos ATRIBUIÇÕES

Operador	Propósito	Exemplo	Resultado
		i = 10	
+=	Somar o valor	i += 5	i = 15
-=	Subtrair o valor	i -= 5	i = 10
*=	Multiplicar o valor	i *= 5	i = 50
/=	Dividir o valor	i /= 5	i = 10
++	Adicionar 1 ao valor	i++ (i = i + 1)	i = 11
--	Subtrair 1 do valor	i-- (i = i - 1)	I = 10



Operadores Comparativos

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



Operadores Lógicos

AND &&

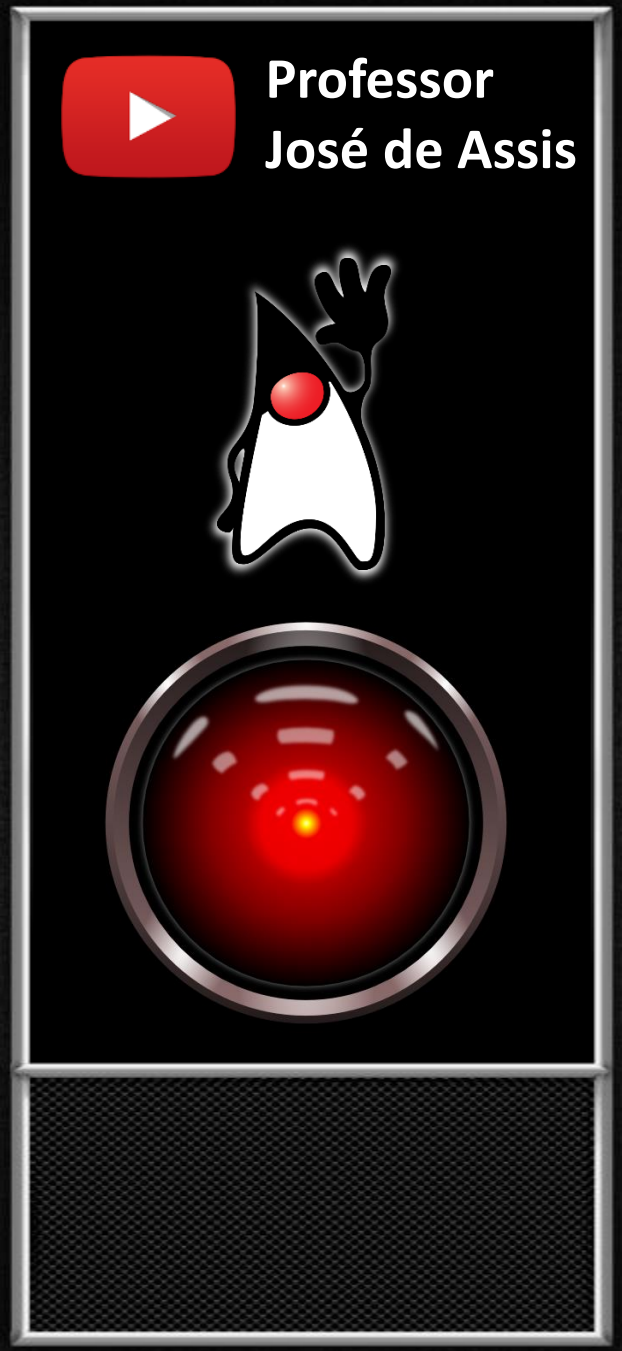
ENTRADA 1	ENTRADA 2	SAÍDA
FALSO	FALSO	**FALSO**
FALSO	VERDADEIRO	**FALSO**
VERDADEIRO	FALSO	**FALSO**
VERDADEIRO	VERDADEIRO	**VERDADEIRO**

OR ||

ENTRADA 1	ENTRADA 2	SAÍDA
FALSO	FALSO	**FALSO**
FALSO	VERDADEIRO	**VERDADEIRO**
VERDADEIRO	FALSO	**VERDADEIRO**
VERDADEIRO	VERDADEIRO	**VERDADEIRO**

NOT !

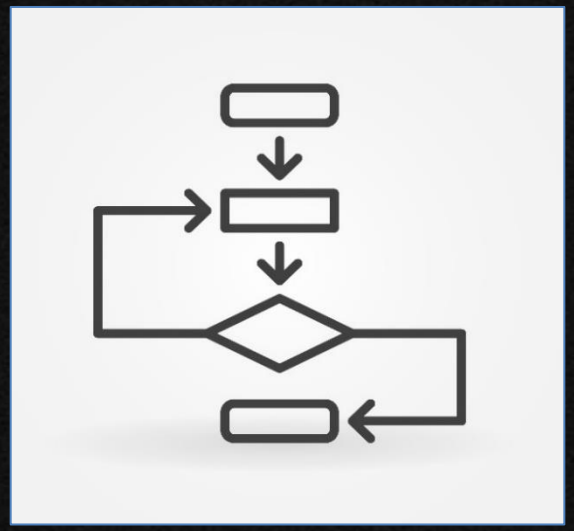
ENTRADA	SAÍDA
FALSO	**VERDADEIRO**
VERDADEIRO	**FALSO**

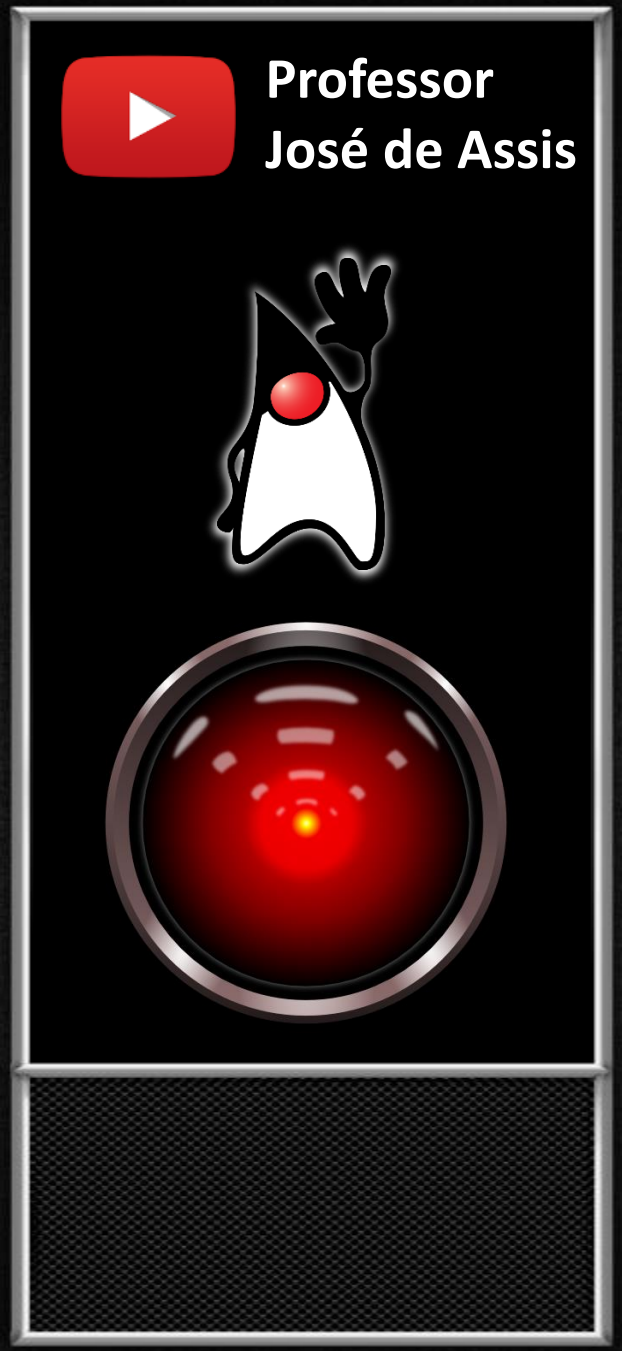


Estruturas de controle

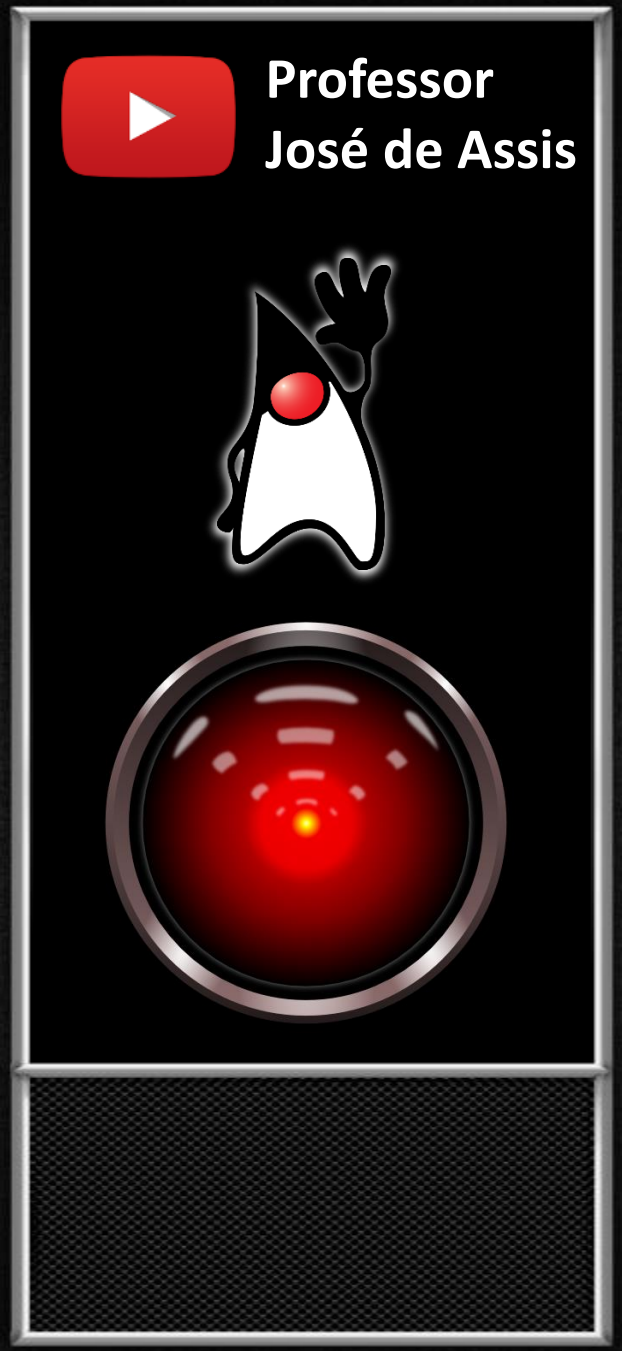
Estruturas de controle são blocos de código que analisam os tipos de dados armazenados nas variáveis. As estruturas de controle são processos básicos de tomada de decisões das máquinas. Existem dois tipos de estruturas de controle:

- Estruturas de Controle Condicional
- Estruturas de Repetições (laços)





Estruturas de controle condicional

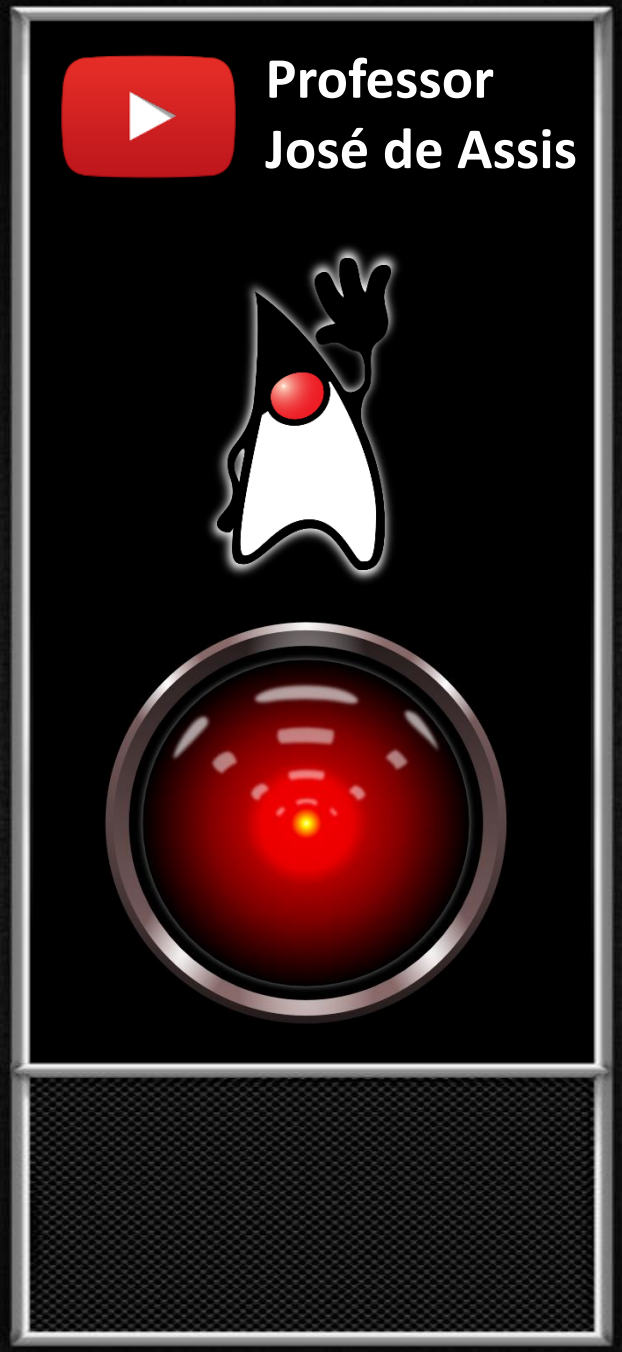


if – if else – else if

```java
if (condição) {
    //verdadeira
}
```

```java
if (condição) {
    //verdadeira
} else {
    //falsa
}
```

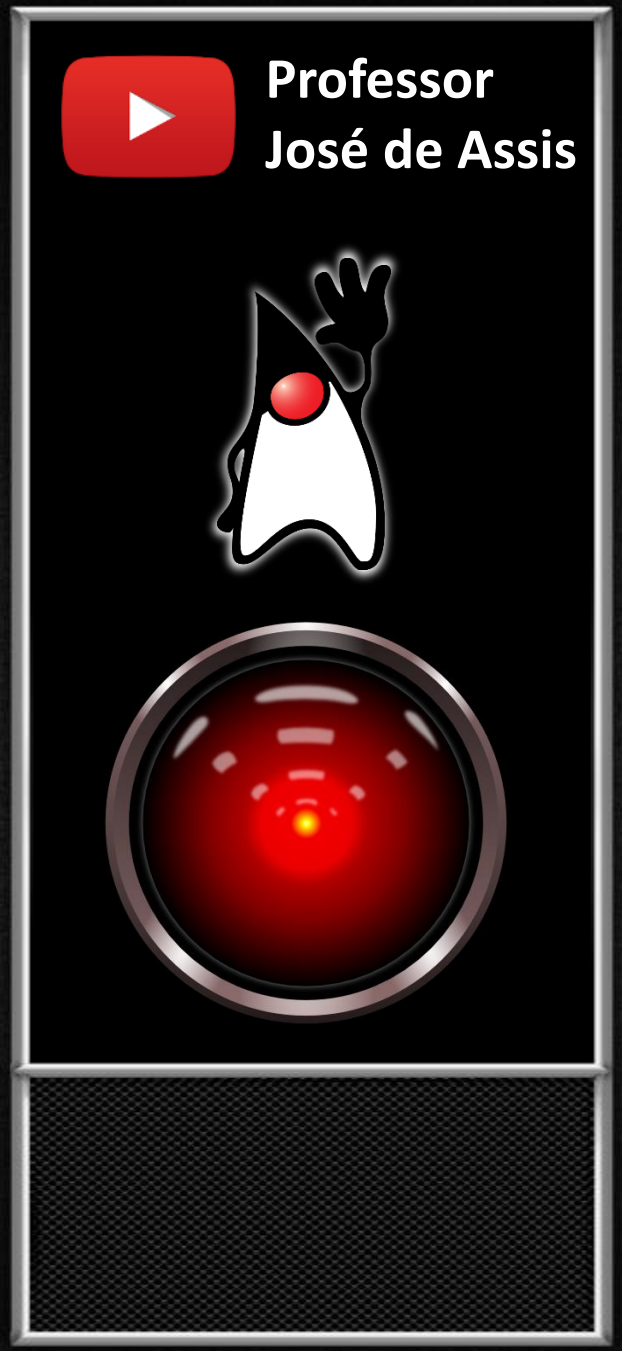
```java
if (condição 1) {
    //condição 1 verdadeira
} else if (condição 2) {
    //condição 2 verdadeira
} else {
    //nenhuma condição verdadeira
}
```



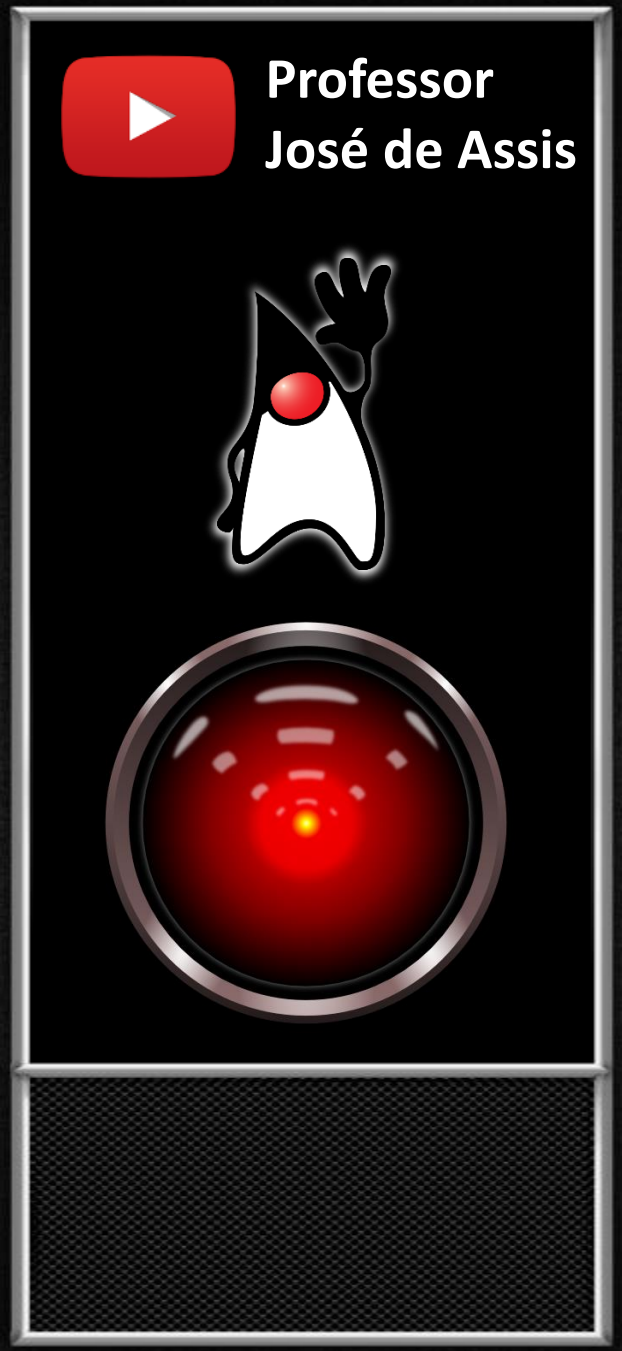
switch case

```
switch (variável) {
case 1:
    //código (variável igual a 1)
    break;
case 2:
    //código (variável igual a 2)
    break;
default:
    //código (variável diferente de 1 ou 2)
    break;
}
```

* Tipos de variáveis aceitas (int e char)



Estruturas de repetições

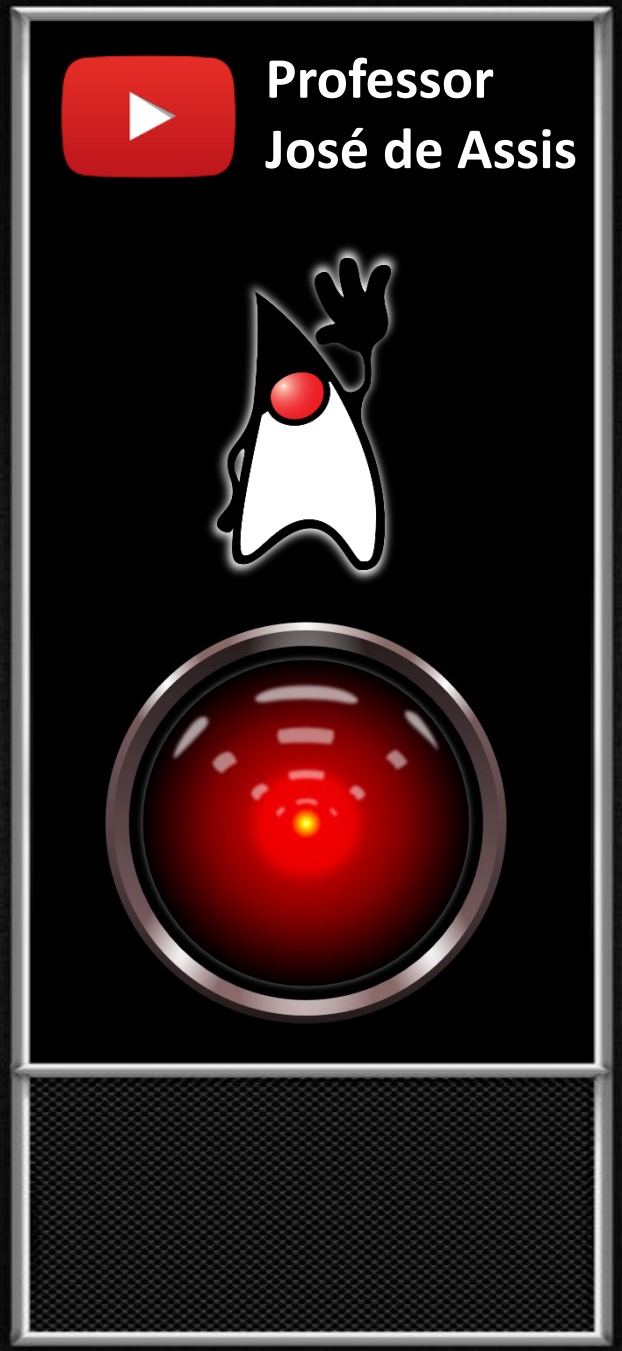


for

Estrutura de repetição usada para repetir um bloco de código envolvido por chaves. Um contador de incremento é utilizado para terminar o loop. A inicialização ocorre primeiro e apenas uma vez. A cada repetição do loop, a condição é testada e se for verdadeira, o bloco de comandos e o incremento são executados. Quando a condição se torna falsa o loop termina.

```
for (int i = 1; i < 10; i++) {
    // código
}
```

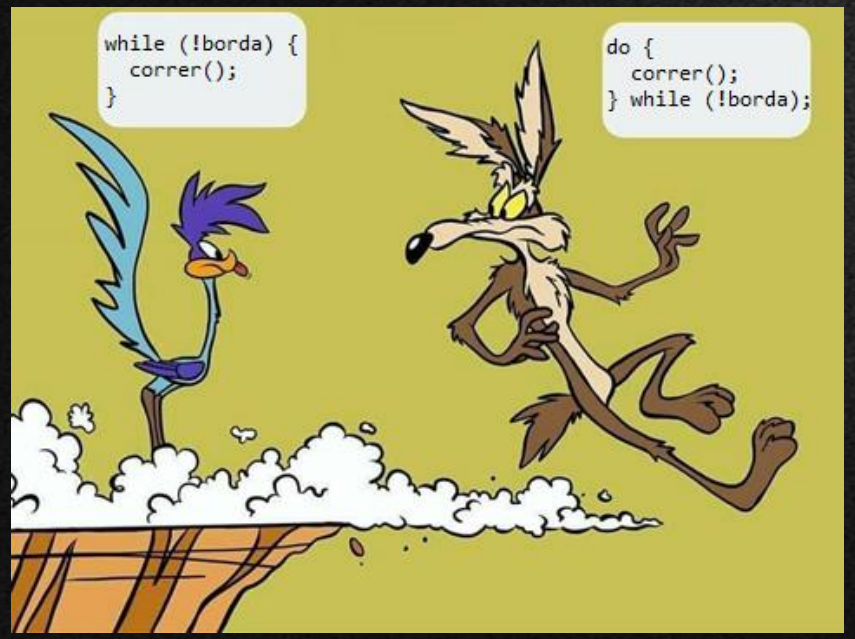
```
for (int j = 10; j > 0; j--) {
    // código
}
```

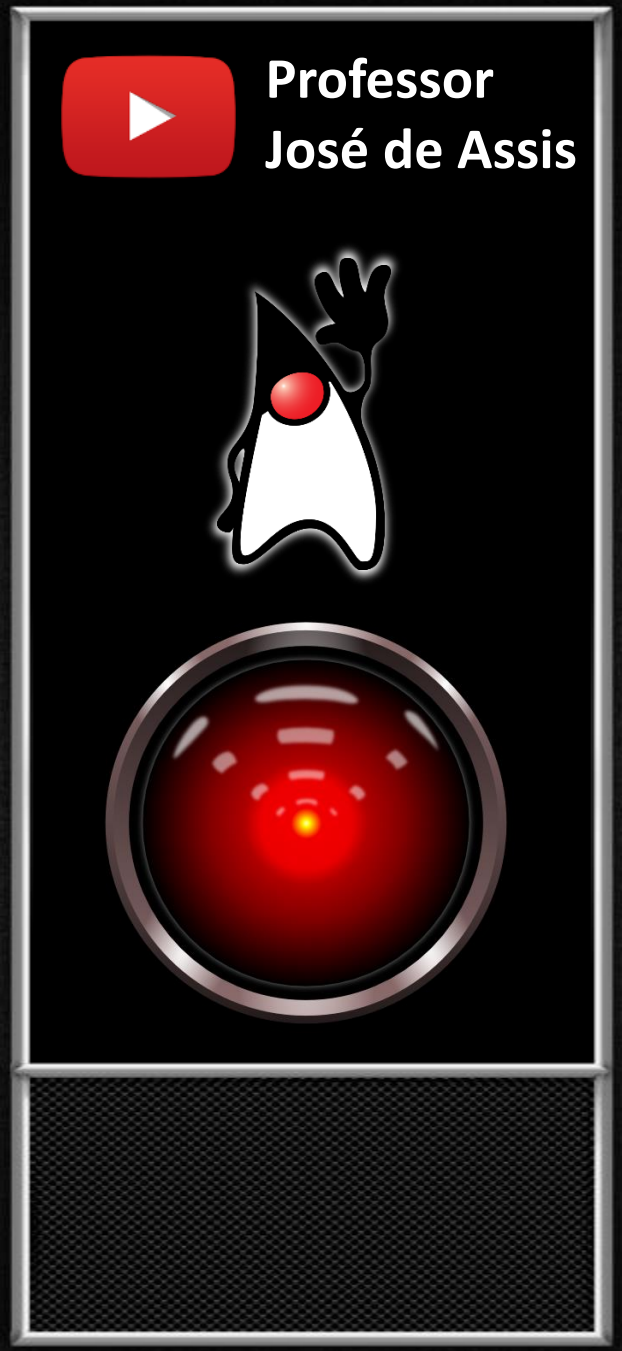


while – do while

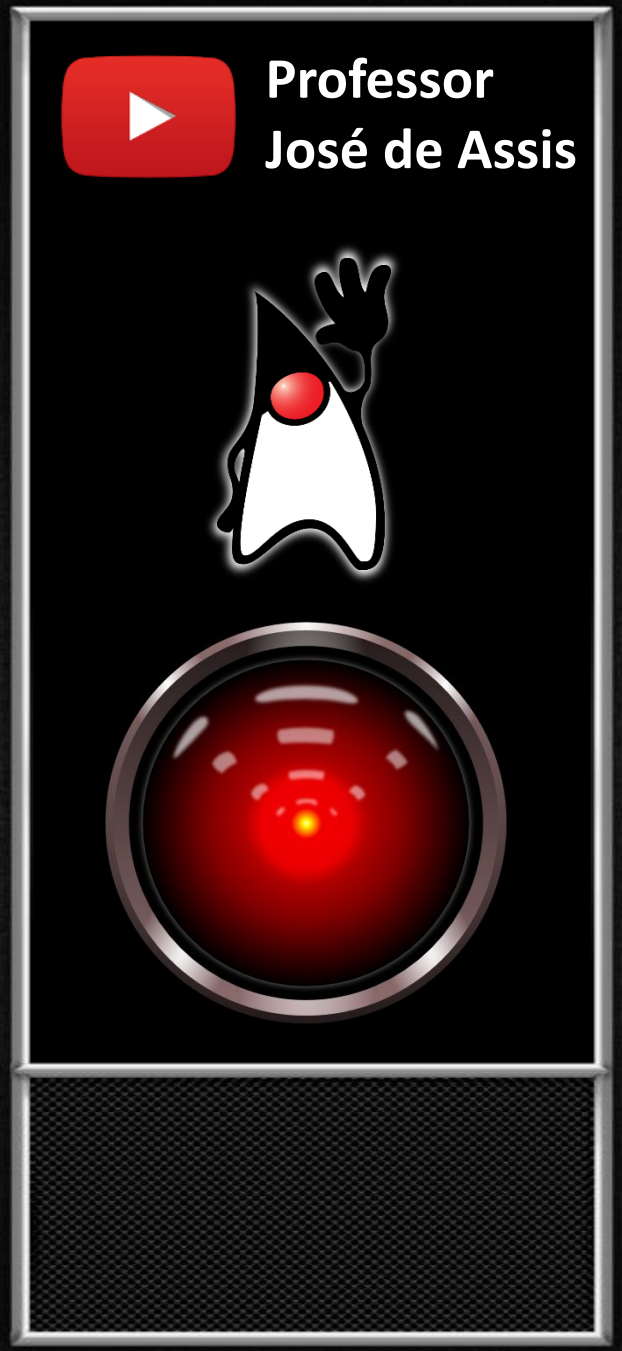
Um loop while se repetirá infinitamente enquanto a condição dentro dos parênteses () for verdadeira. Algo deve mudar a condição, ou o loop while nunca encerrará. O loop do...while funciona da mesma forma que o loop while, com a exceção da condição ser testada no final.

```
while (condição) {
    //verdadeira
}



do {
    //1º loop
    //(verdadeira ou falsa)
    //Demais loops
    //(verdadeira)
} while (condição);
```



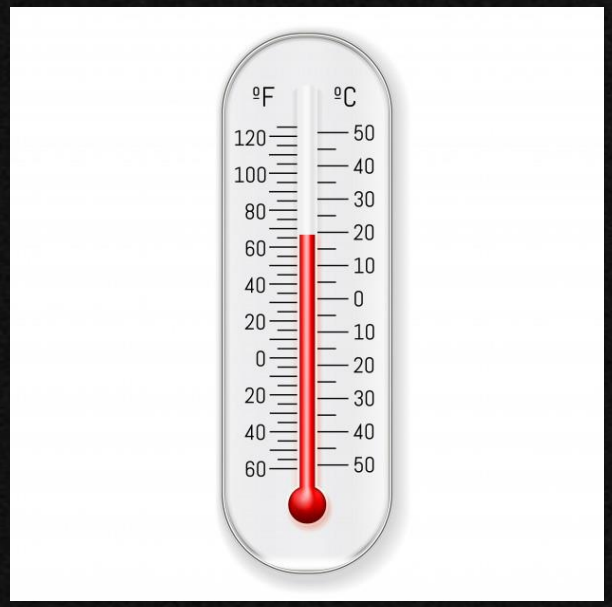


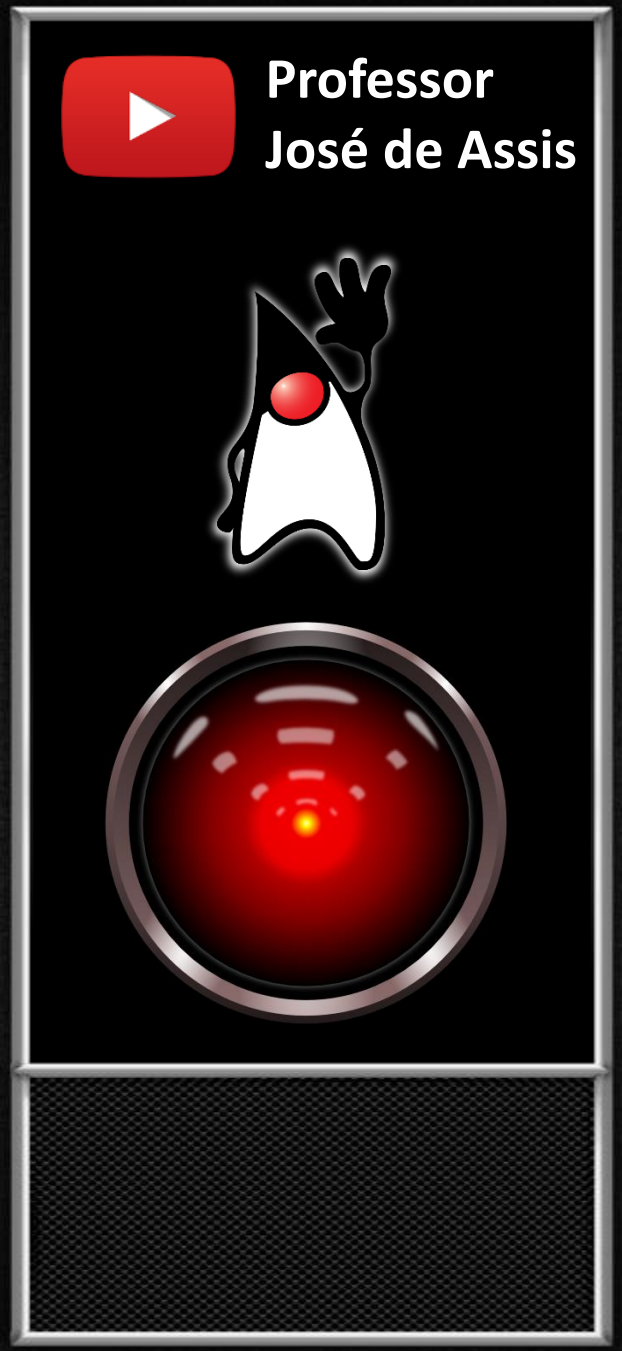
EXERCÍCIOS



Exercício 1

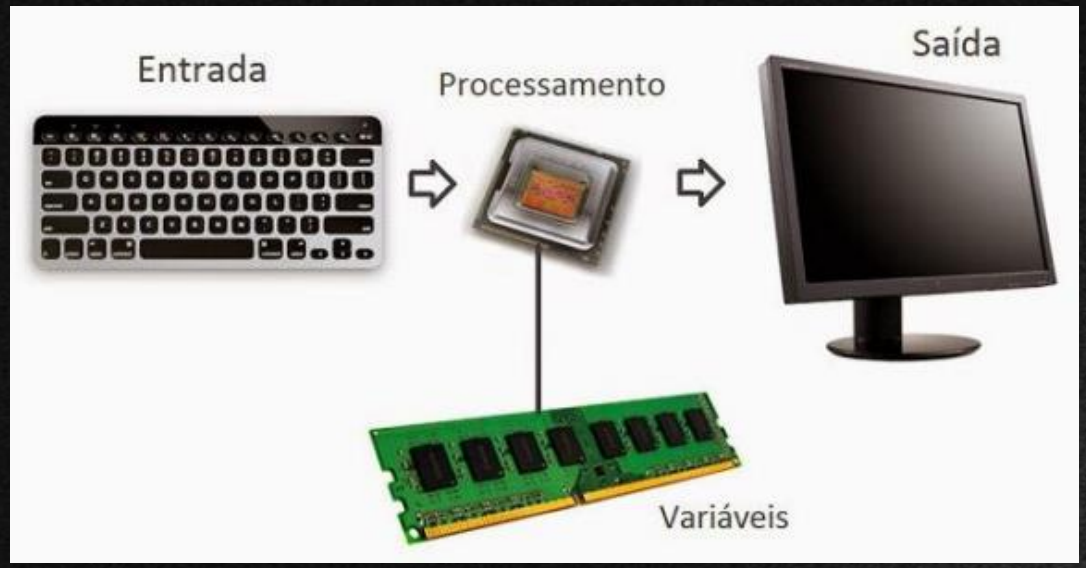
Nos Estados Unidos a temperatura em geral é medida em uma escala denominada fahrenheit. Desenvolva uma aplicação no console (terminal) que faça a conversão da temperatura em Fahrenheit para Celsius.



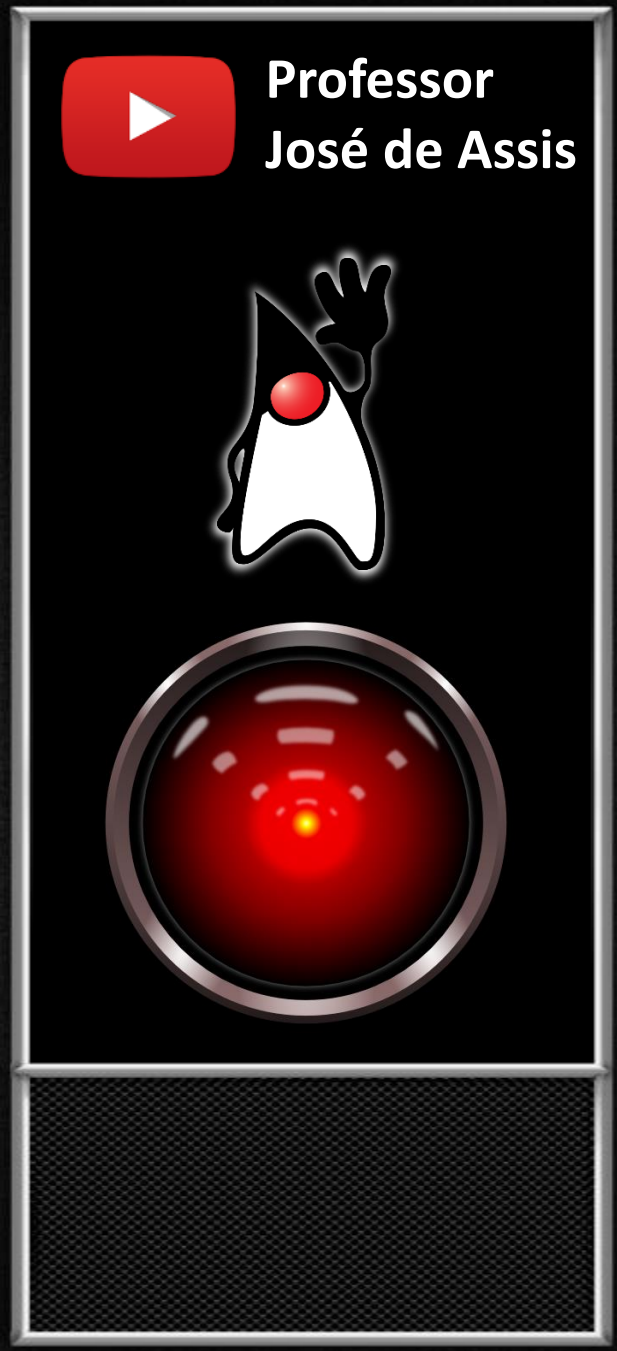


Fórmula:

$$\frac{°C}{5} = \frac{°F - 32}{9}$$



Variáveis: c,f (double)
Entrada: f
Processamento: c = (5 * (f – 32)) / 9
Saída: c

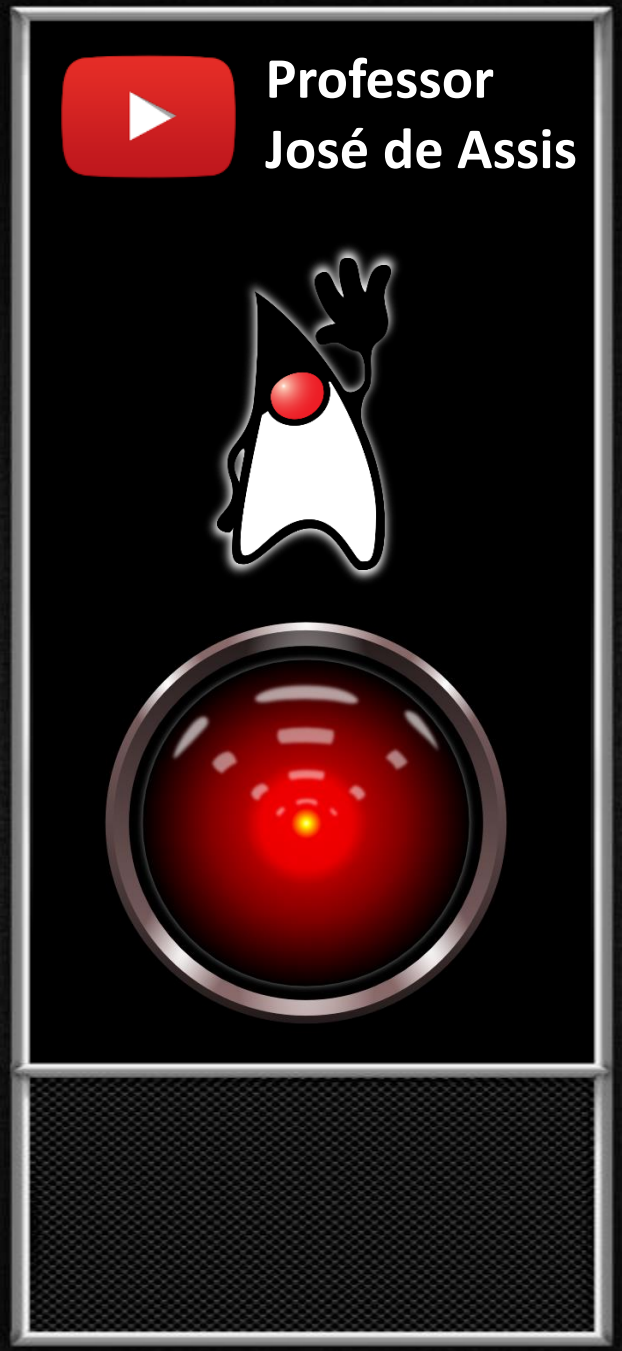


Exercício 2

Desenvolva uma aplicação no console (terminal) para calcular o valor da porcentagem usando **Regra de 3**, conforme modelo abaixo:

x% de y = valor

Exemplo: 10% de 200 = 20

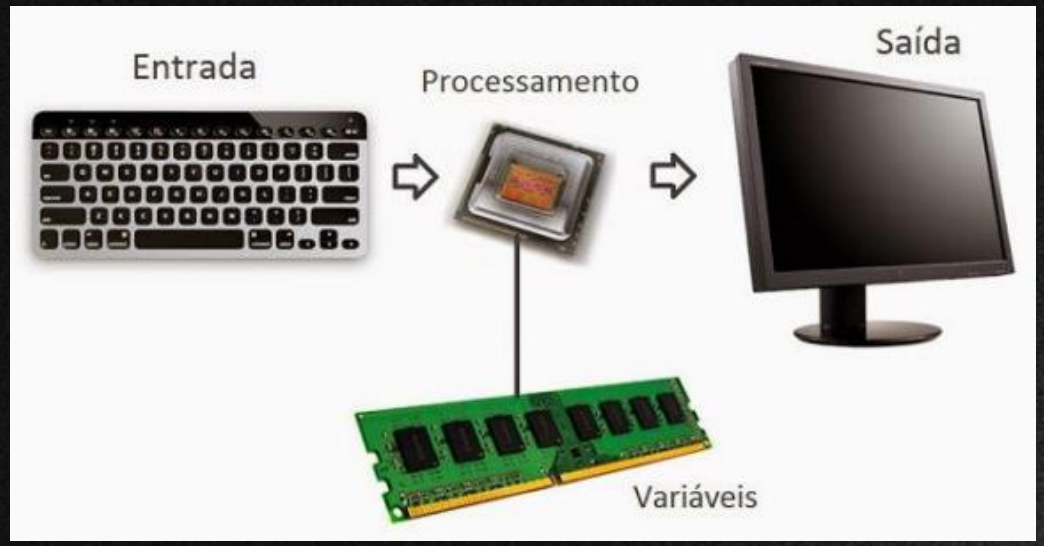


Fórmula:

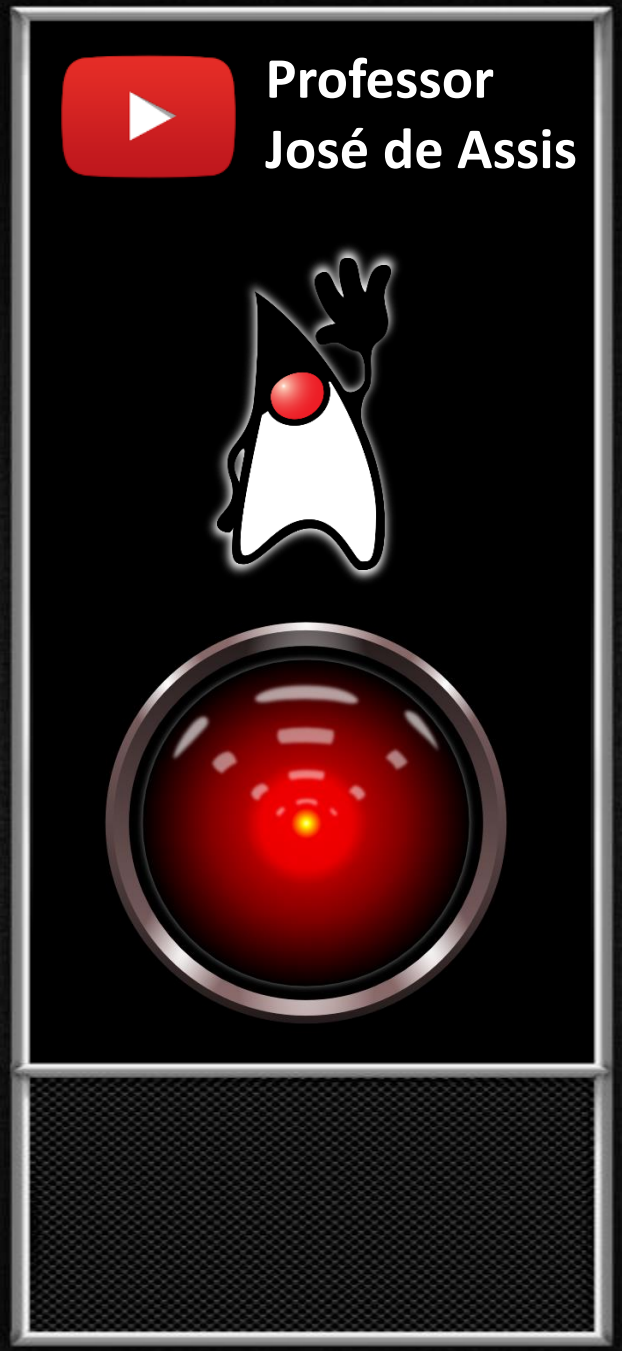
x% de y = valor

y	100%
valor	x%

valor = (x * y) / 100



Variáveis: x,y,valor (double)
Entrada: x,y
Processamento: valor = (x * y) / 100
Saída: valor

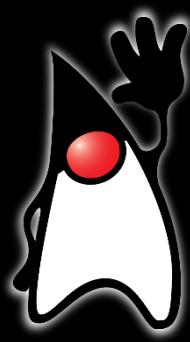


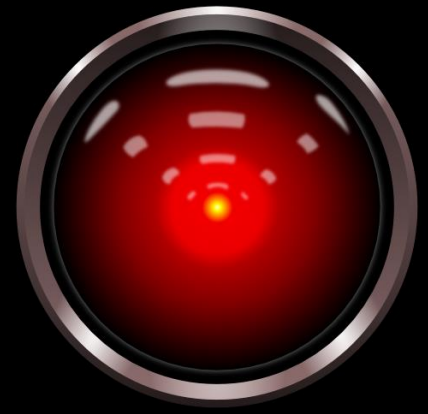
Exercício 3

Desenvolva uma aplicação no console (terminal) de ponto de vendas (PDV), conforme modelo abaixo:

- Valor total: **200,00**
- Desconto(%): **5**
- Total com desconto: **190,00** (valor calculado)
- Valor pago: **200,00**
- Troco: **10,00** (valor calculado)

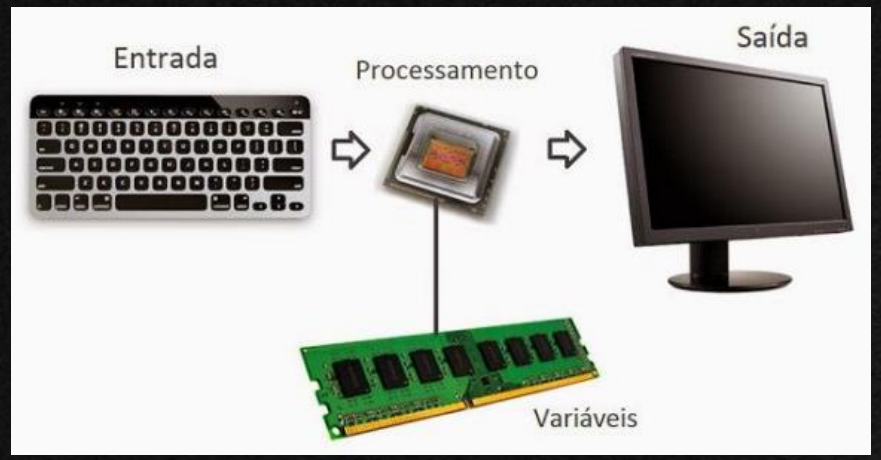
Professor
José de Assis





Fórmulas:

Total com desconto = total - desconto
Troco = valor pago - total com desconto



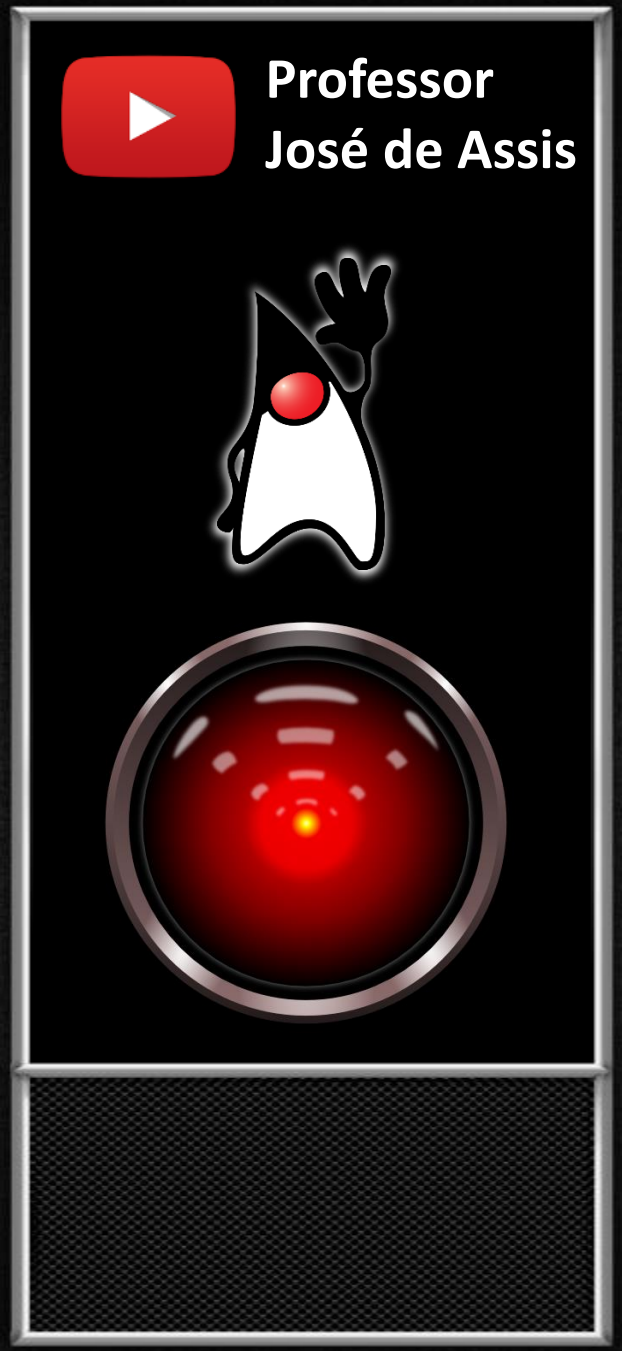
Variáveis: total, desconto, totalDesconto, valorPago, troco (double)
Entrada1: total, desconto
Processamento1: totalDesconto = total - (desconto * total) /100
Saída1: totalDesconto
Entrada2: valorPago
Processamento2: troco = valorPago - totalDesconto
Saída2: troco

Exercício 4

Desenvolva uma aplicação no console (terminal) para calcular o valor da hora de um serviço, de acordo com a fórmula abaixo:

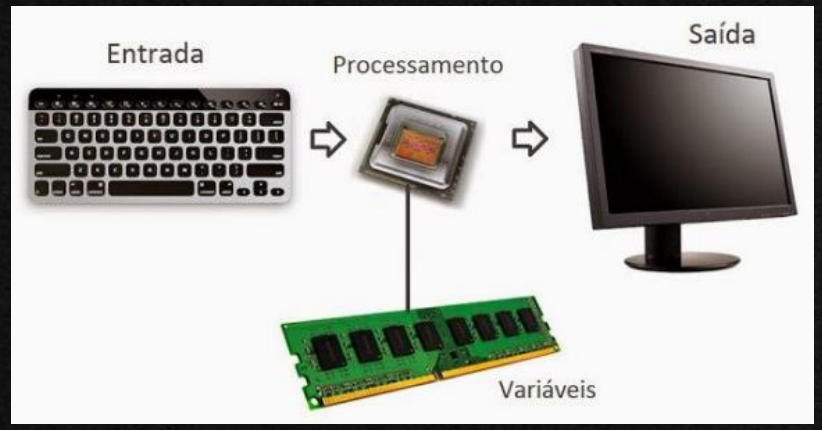
$$\text{hora} = \frac{\text{remuneração mensal} + \text{impostos} + \text{custo operacional} + \text{investimentos}}{\text{Carga horária mensal de trabalho}}$$

Impostos: 30% da remuneração mensal
Investimento: 20% da remuneração mensal

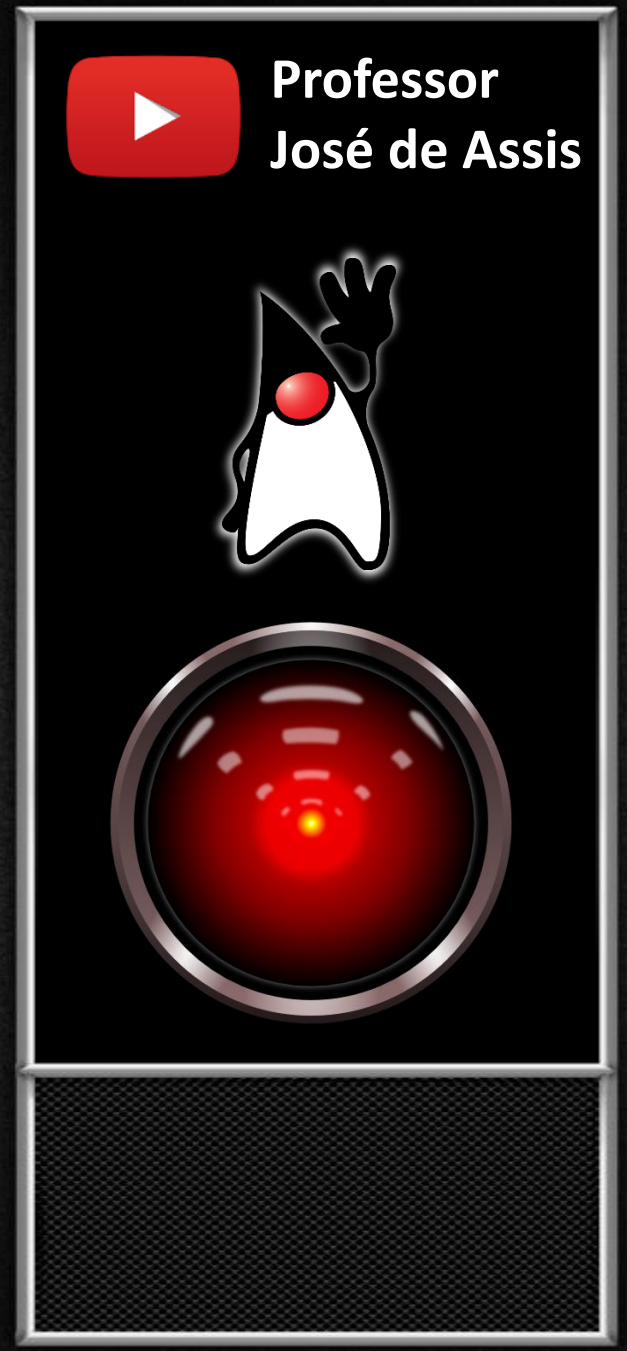


Fórmula

$$hora = \frac{remuneração\ mensal + impostos + custo\ operacional + investimentos}{Carga\ horária\ mensal\ de\ trabalho}$$

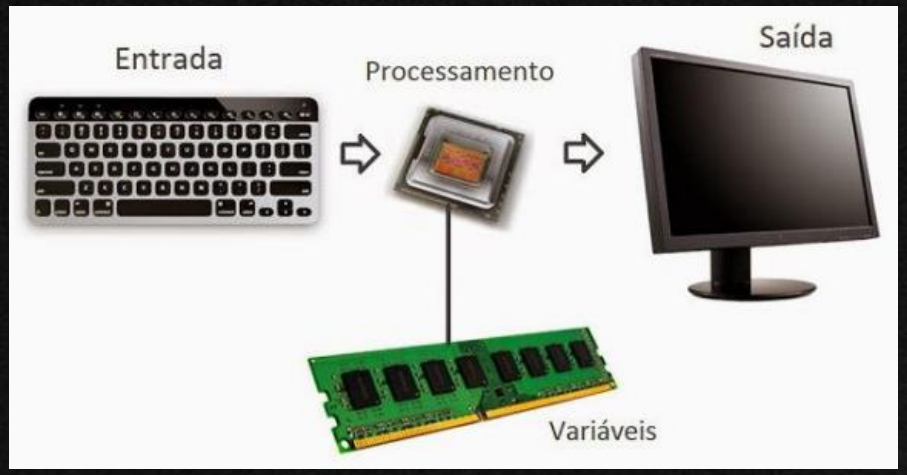


Variáveis: hora, remuneracao, custo, cargaHoraria (double)
Entrada: remuneracao, custo, cargaHoraria
Processamento:
hora = (remuneracao + (remuneracao * 0.3) + custo + (remuneracao * 0.2)) / cargaHoraria
Saída: hora



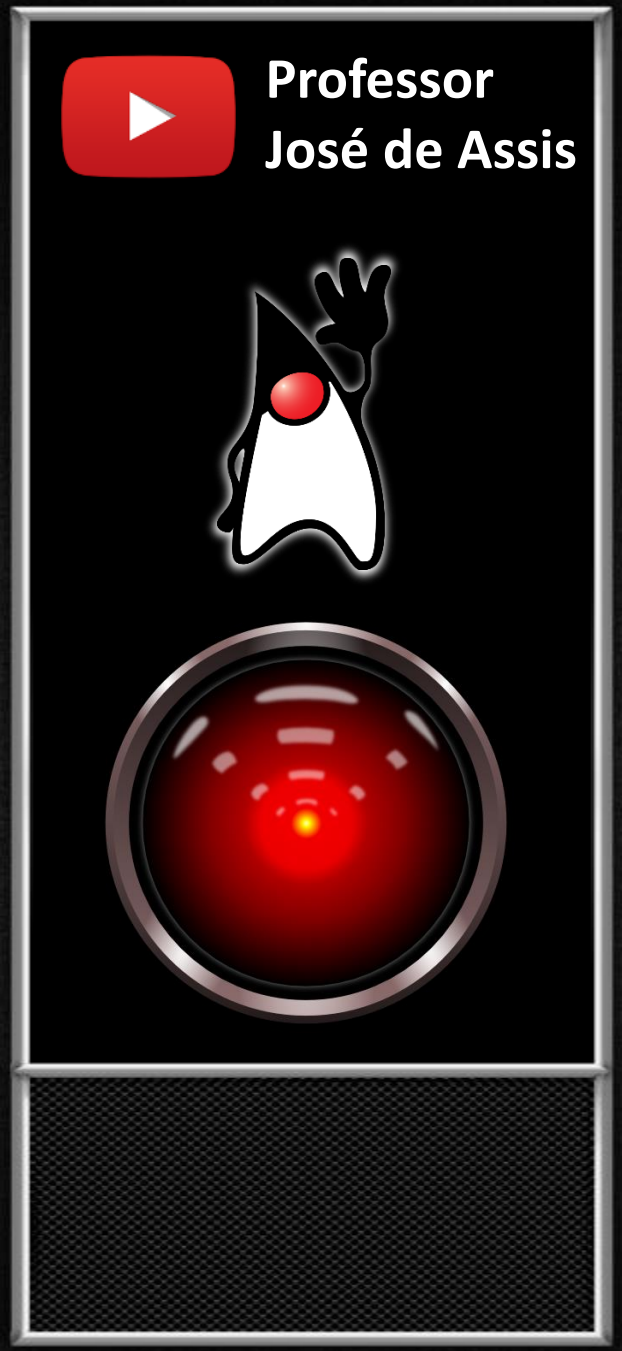
Exercício 5

Para carros flex é preciso ter cautela ao escolher o combustível na hora de abastecer. A principal diferença de preços e vantagens entre os dois combustíveis está na proporção preço X desempenho. Para o álcool ser mais vantajoso do que a gasolina, o preço do litro tem que custar até 70% do litro da gasolina. Baseado nestas informações desenvolva um aplicativo no console (terminal) para determinar qual é o combustível mais vantajoso para abastecer.



Variáveis: alcool, gasolina (double)
Entrada: alcool, gasolina
Processamento / Saída:

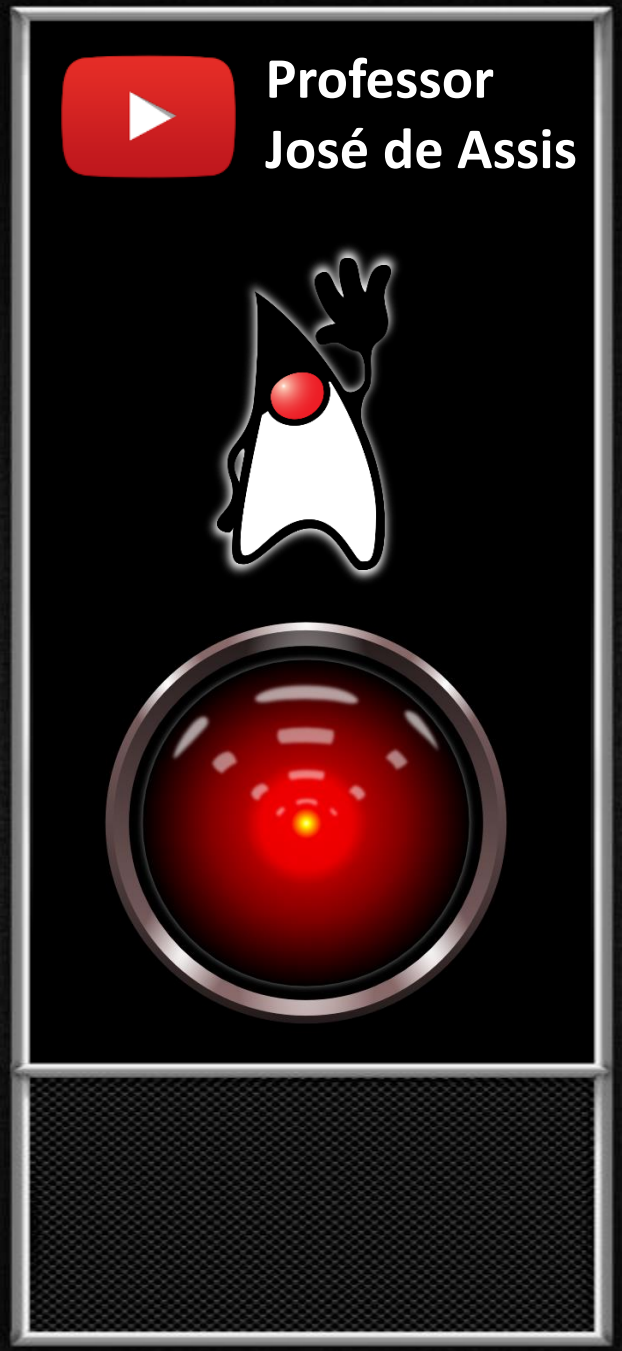
```
if (alcool < 0.7 * gasolina) {
    //abastecer com álcool
} else {
    //abastecer com gasolina
}
```



Exercício 6

Desenvolva um aplicativo no console (terminal) para calcular o valor do IMC. O aplicativo deverá exibir além do valor do IMC a classificação de acordo com a tabela abaixo:

IMC	Classificação
abaixo de 18,5	abaixo do peso
entre 18,6 e 24,9	Peso ideal
entre 25,0 e 29,9	Levemente acima do peso
entre 30,0 e 34,9	Obesidade grau I
entre 35,0 e 39,9	Obesidade grau II (severa)
acima de 40	Obesidade III (mórbida)



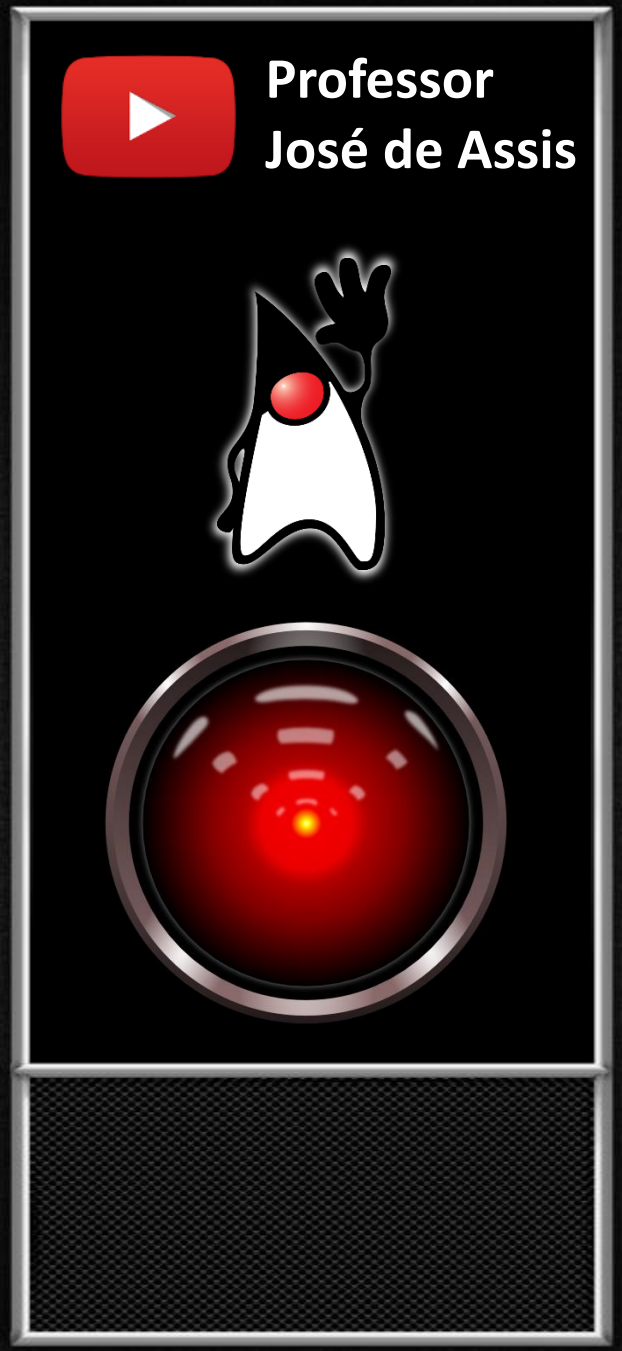
Fórmula:

$$IMC = \frac{PESO}{(ALTURA)^2}$$

Variáveis: peso, altura, imc (double)
Entrada: peso, altura
Processamento: imc = peso / (altura * altura)
Saída: imc

if
else if

IMC	Classificação
abaixo de 18,5	abaixo do peso
entre 18,6 e 24,9	Peso ideal
entre 25,0 e 29,9	Levemente acima do peso
entre 30,0 e 34,9	Obesidade grau I
entre 35,0 e 39,9	Obesidade grau II (severa)
acima de 40	Obesidade III (mórbida)



Exercício 7

Desafio:
Desenvolva o jogo “Pedra – Papel – Tesoura”, conforme modelo abaixo:

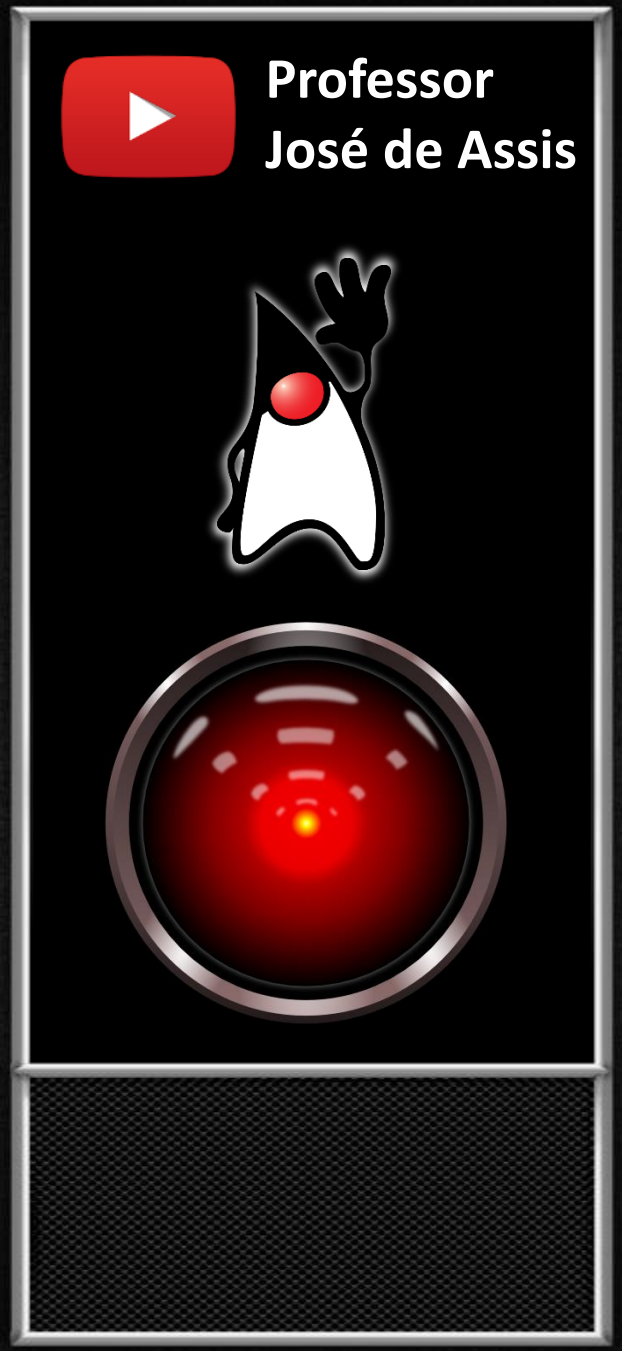
_____ JoKenPow _____

1. Pedra
2. Papel
3. Tesoura

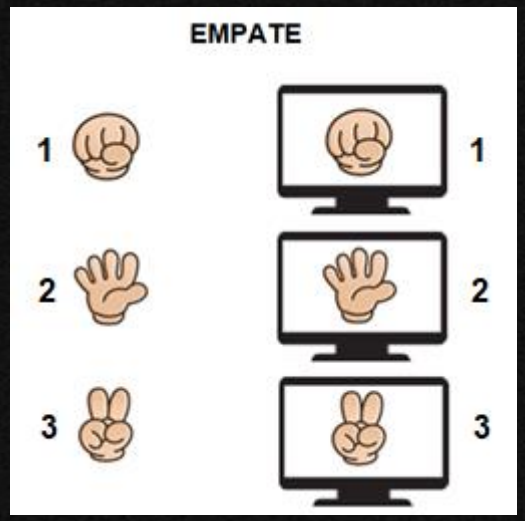
Digite a opção desejada:

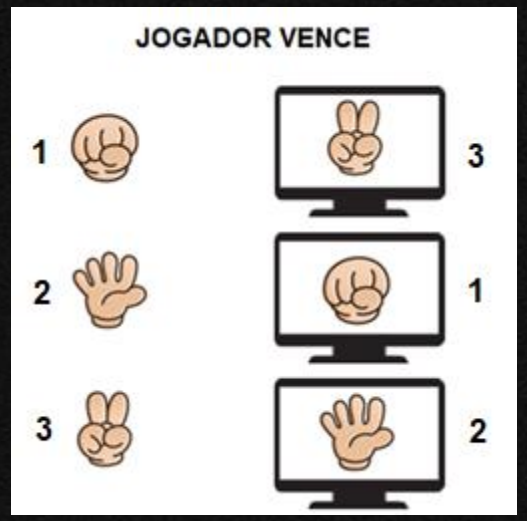
Jogador escolheu:
Computador escolheu:

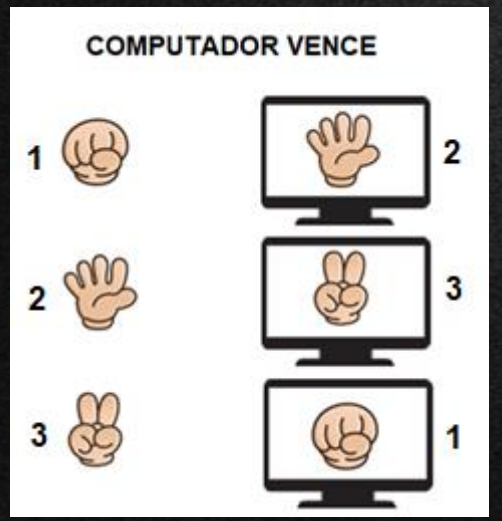
VENCEDOR:

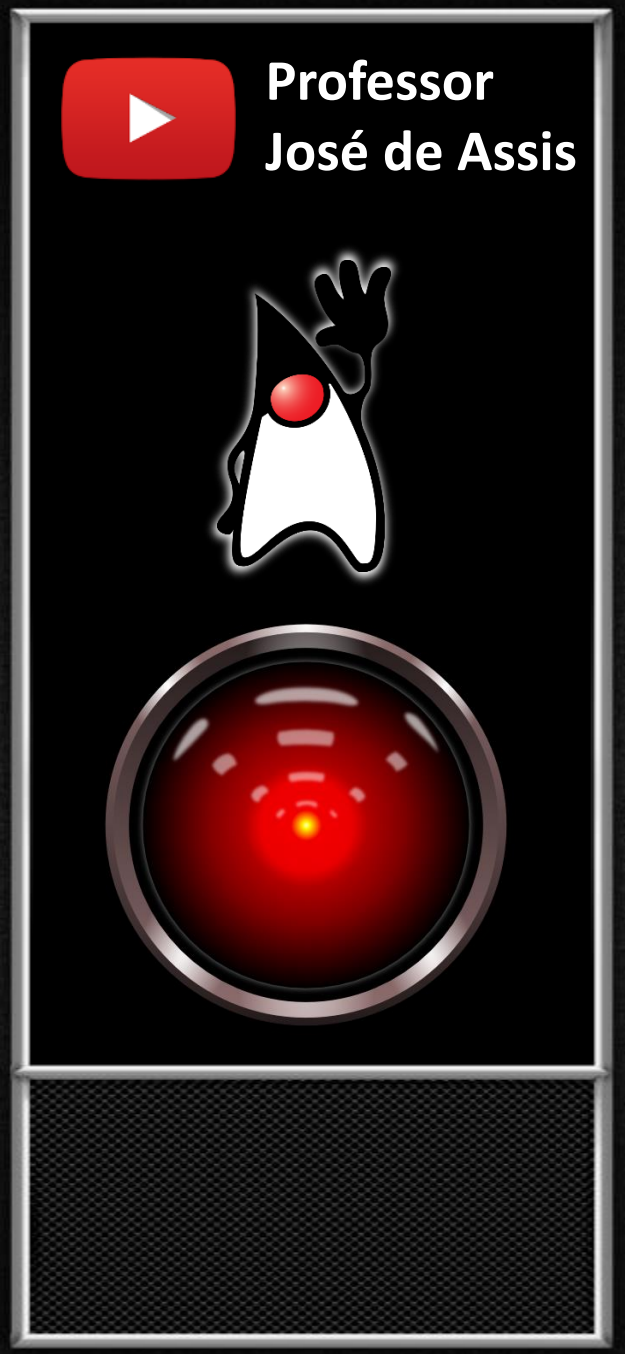


Variáveis: jogador, computador (int)
Entrada: jogador, computador
Processamento / Saída:



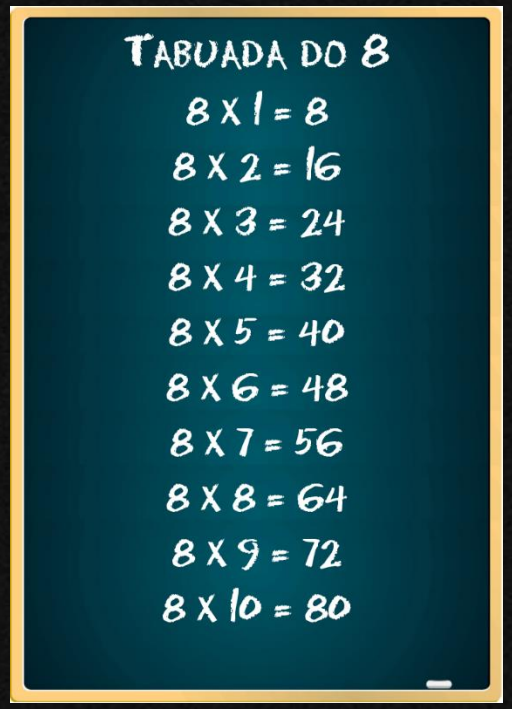


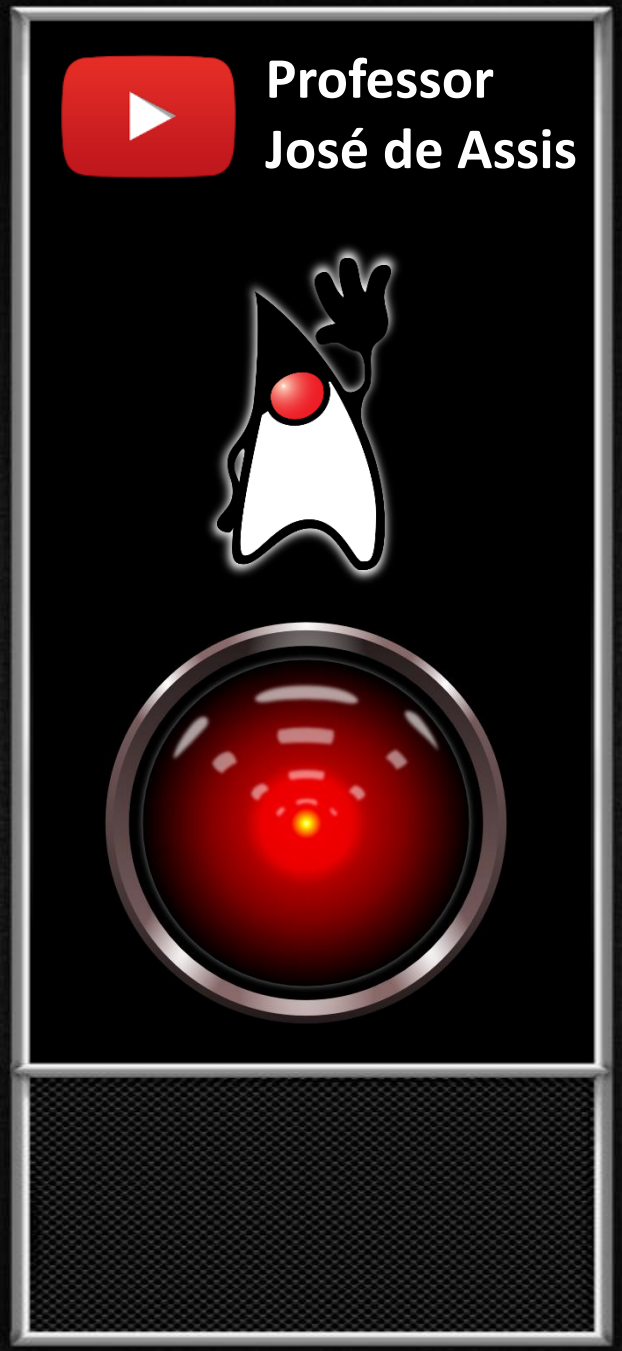




Exercício 8

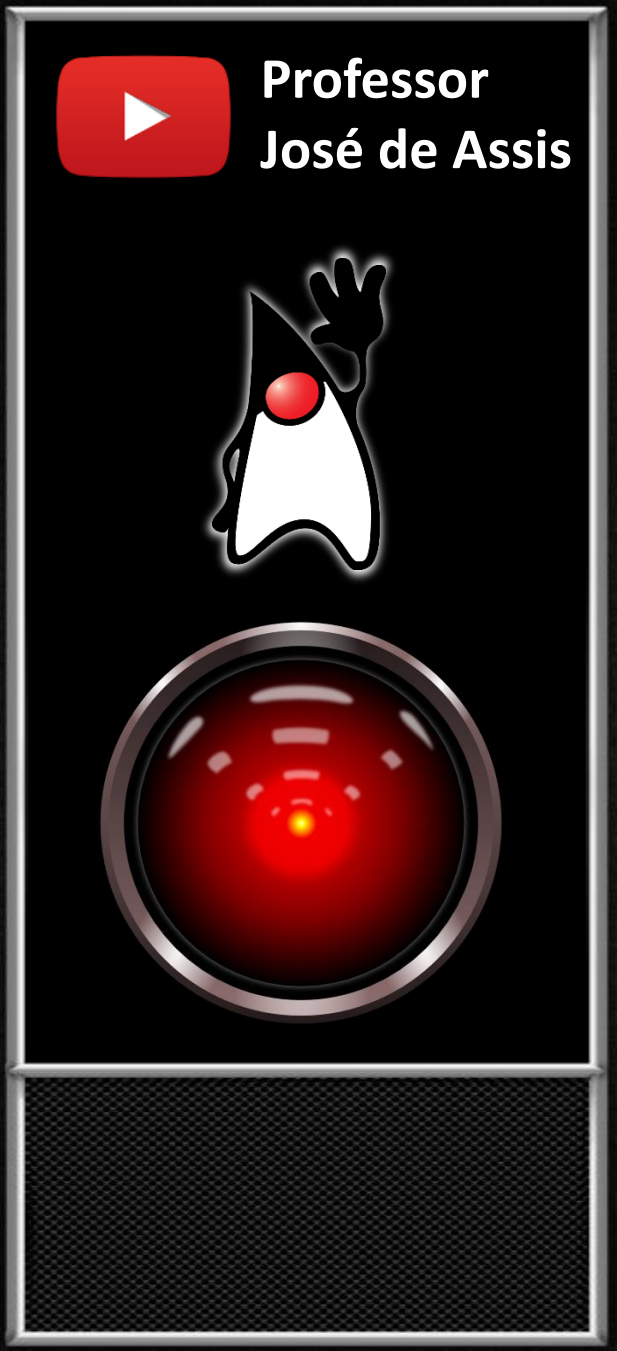
Desenvolva um aplicativo no console (terminal) para calcular o valor da tabuada. O aplicativo deverá perguntar qual é o valor da tabuada a ser calculada.





Exercício 9

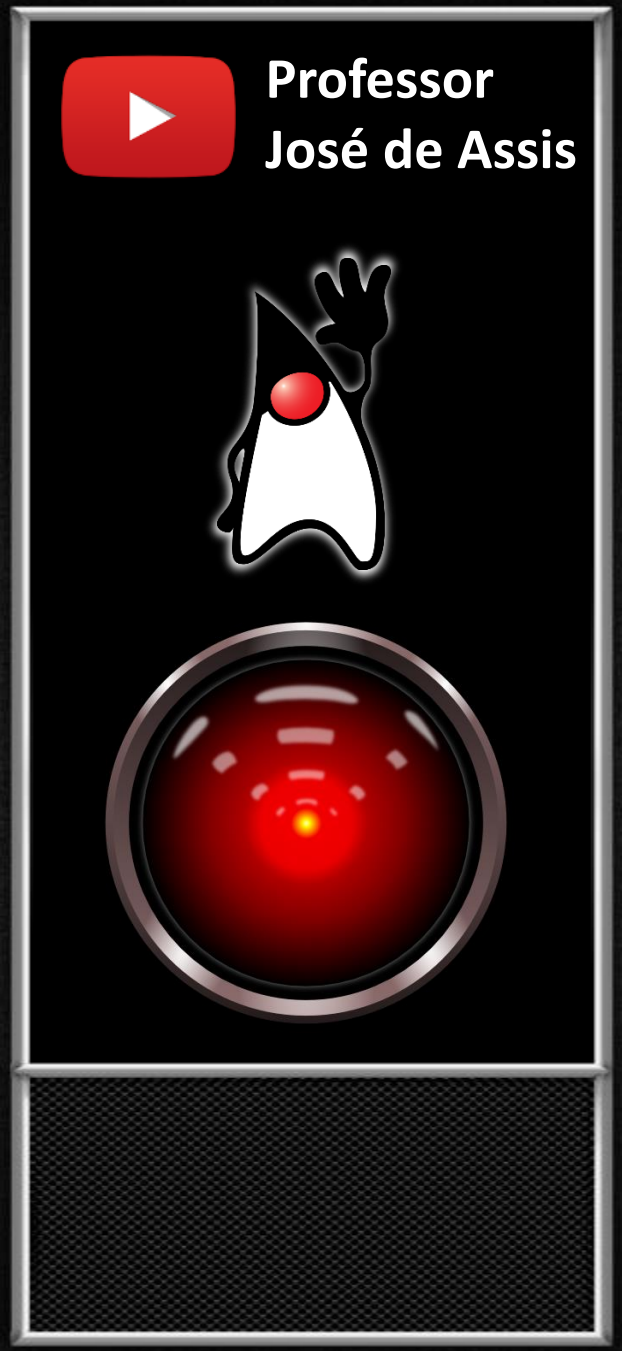
Desenvolva um aplicativo no console (terminal) para fazer o lançamento de um dado de 6 faces. O aplicativo deverá executar um sorteio usando a função **Math.random()**, exibindo a face do dado sorteada.
Após a exibição da face sorteada o aplicativo deverá perguntar se o jogador deseja lançar o dado novamente. Se o jogador responder que sim um novo lançamento do dado será executado.



Métodos

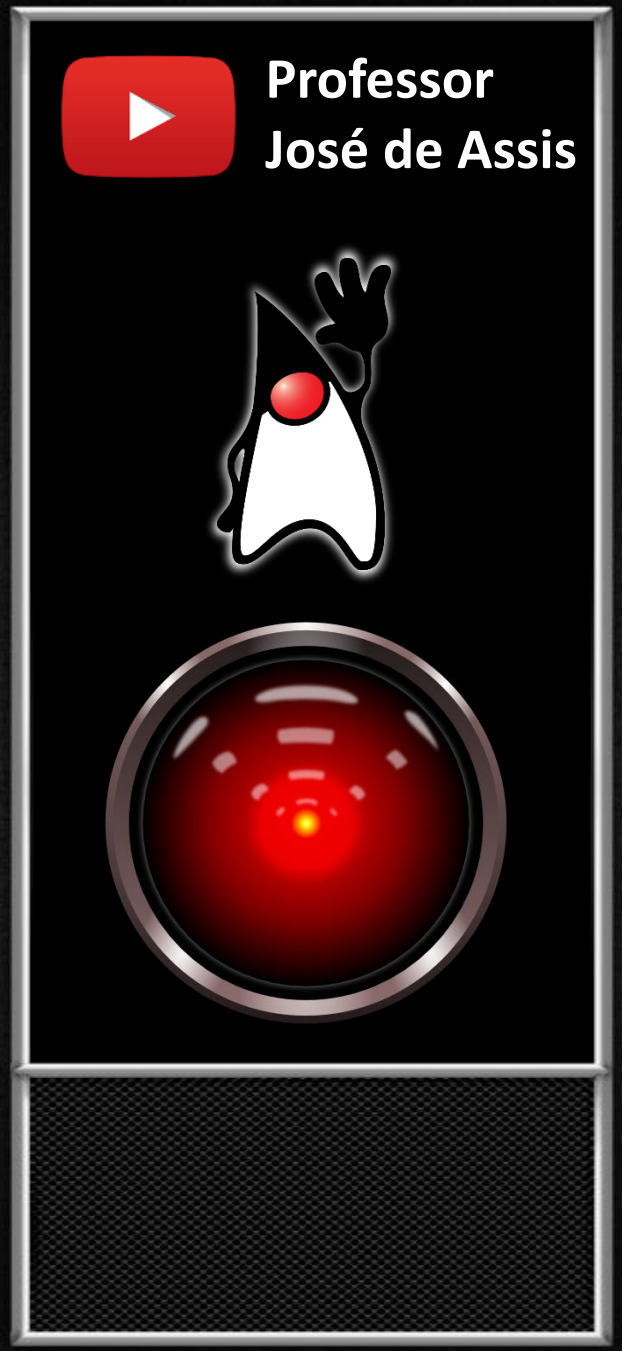
No Java os métodos são equivalentes as funções de outras linguagens de programação e precisam sempre ser definidos dentro de uma classe.

- main
- static
- void



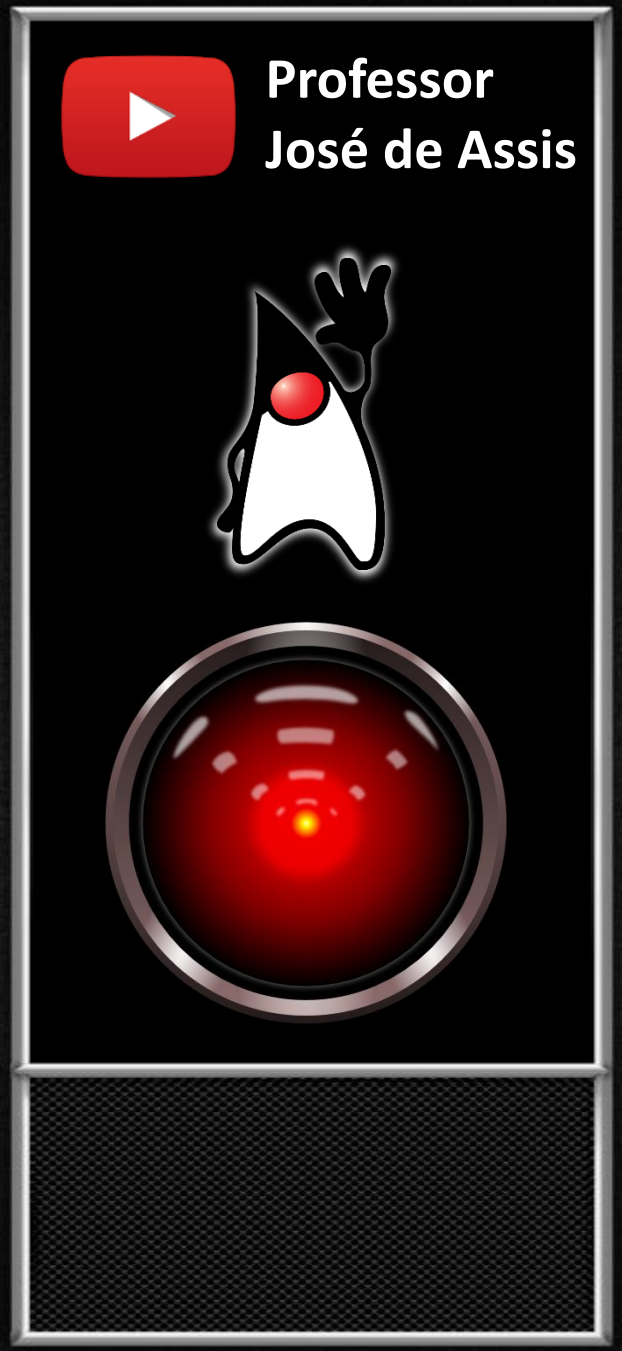
POO

O Java e a maioria das linguagens de programação são orientadas a objetos. Para um computador entender o que é um objeto é necessário descrever as características do objeto e também as ações que podem ser executadas.



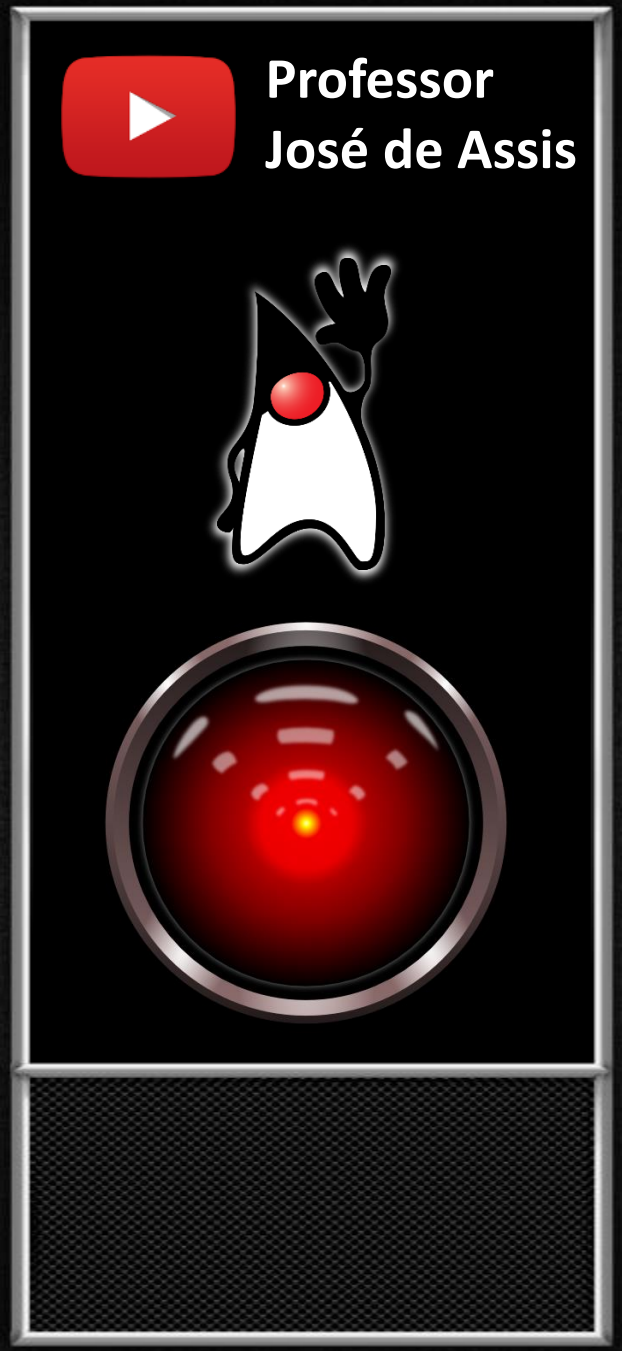
Pilares

- Abstração
- Herança
- Polimorfismo
- Encapsulamento



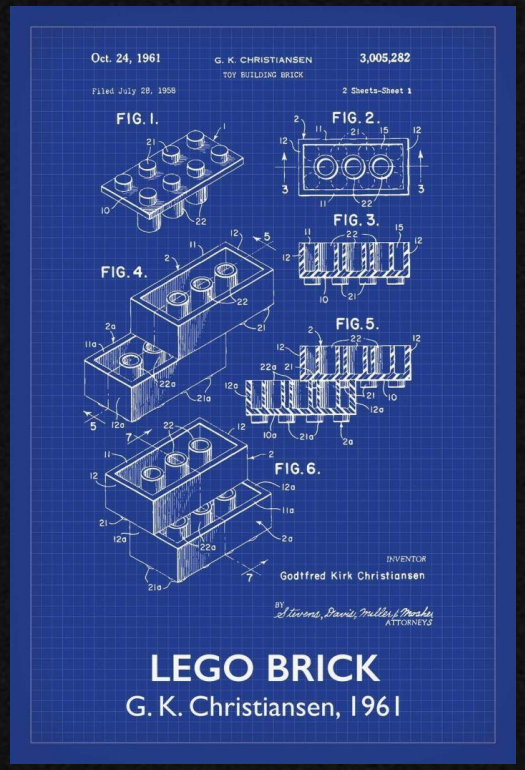
Abstração

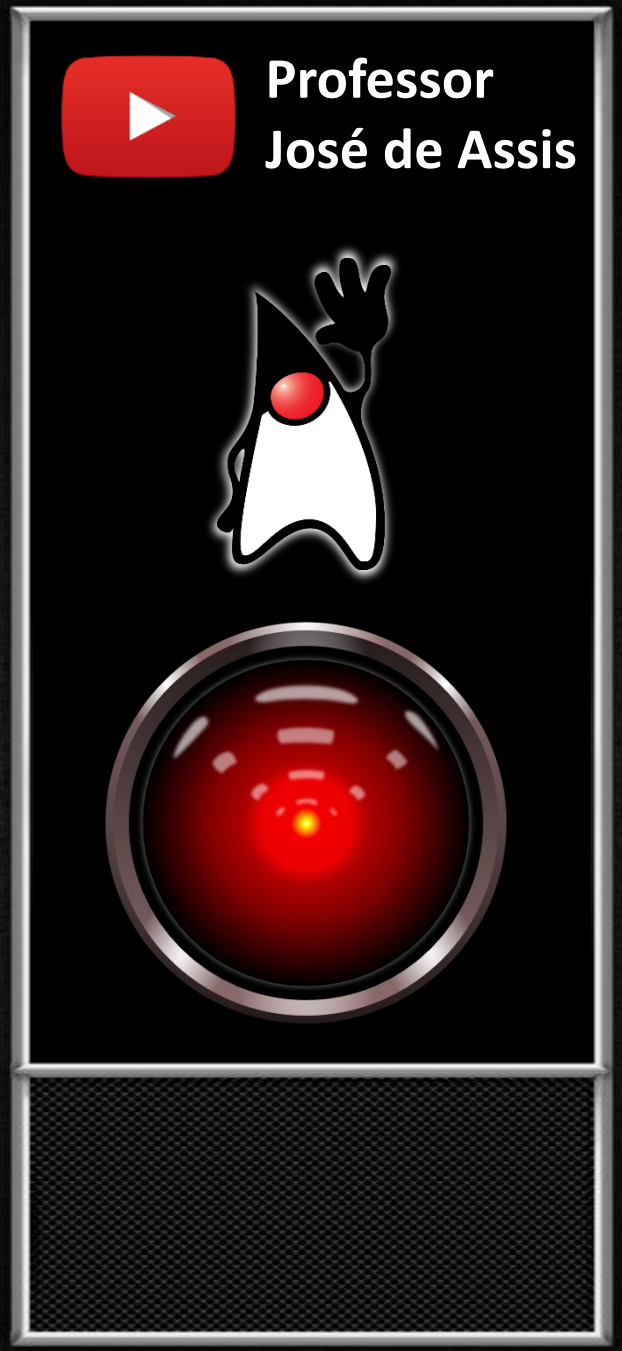
A abstração é usada para definir um modelo para a criação dos objetos. Na linguagem Java usamos uma classe para criar um modelo.



Classe Modelo

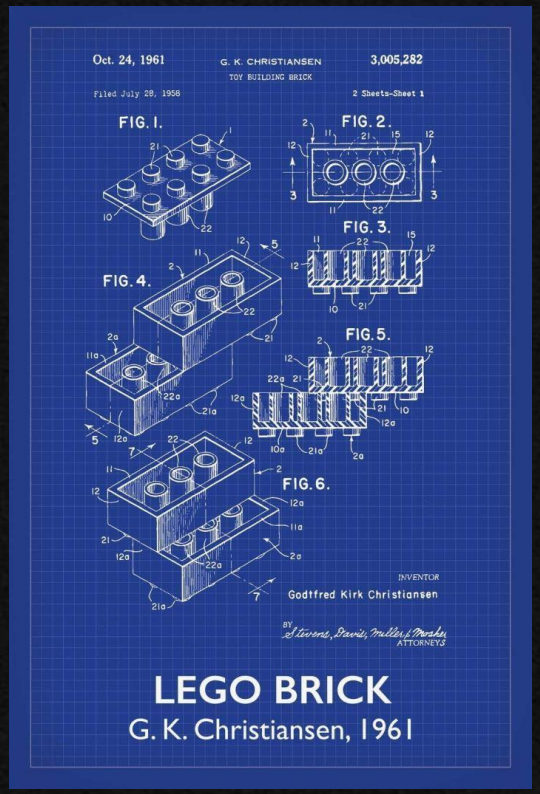
Para explicar uma classe modelo vamos fazer uma analogia com a patente da Lego que descreve as características de um bloco além de mostrar que é possível encaixar um bloco no outro. Este modelo será usado para criar os objetos (blocos).

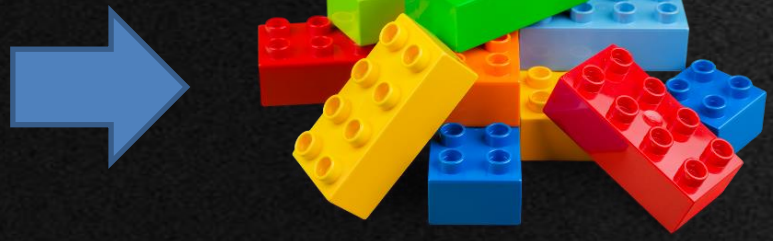


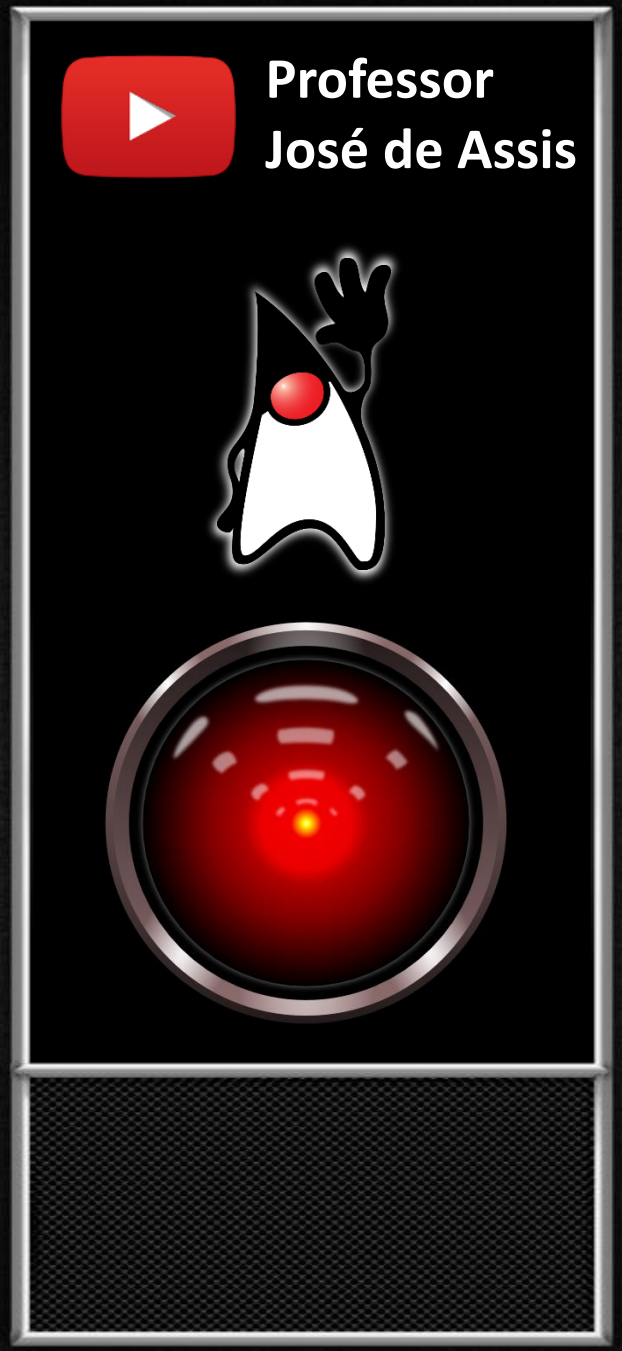


Objetos

Objetos são criados à partir da classe modelo. Percebam na imagem que temos blocos de diferentes cores e tamanhos, porém todos foram criados a partir de um modelo.

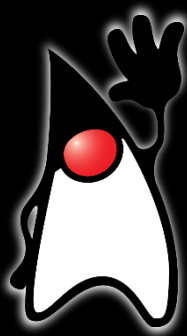


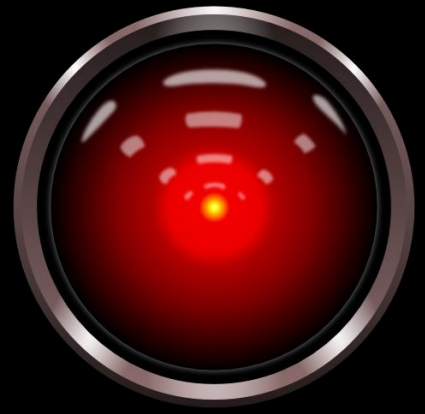




Exemplo

Professor
José de Assis





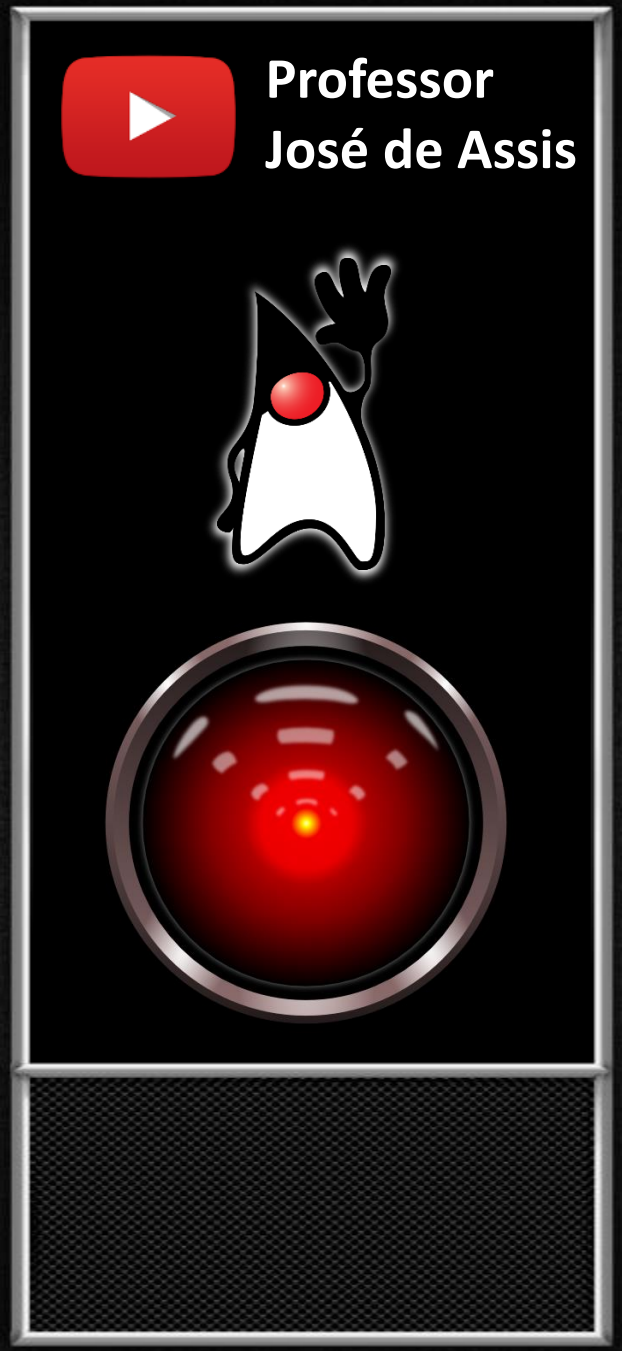
Classe Modelo

Bloco
resistencia : int textura : String
construir() : void minerar() : void craftar() : void

Tipo
Classe

atributos
variáveis

ações
métodos

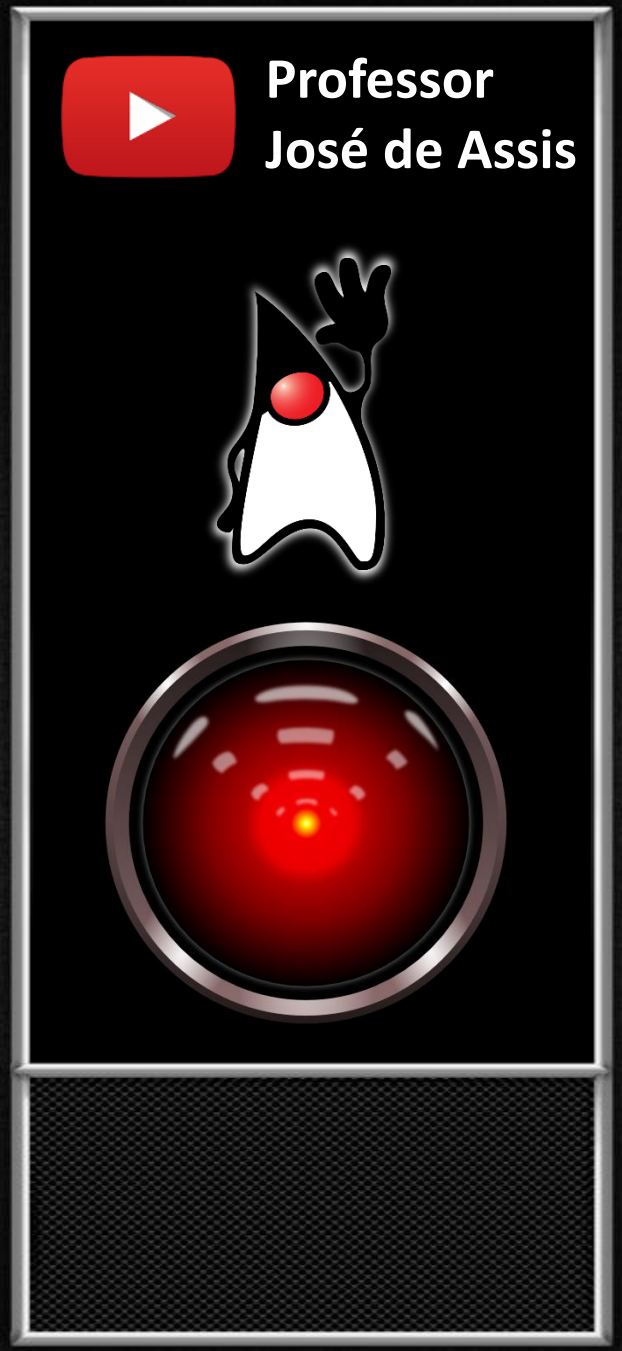


Bloco
resistencia : int textura : String
construir() : void minerar() : void craftar() : void

new

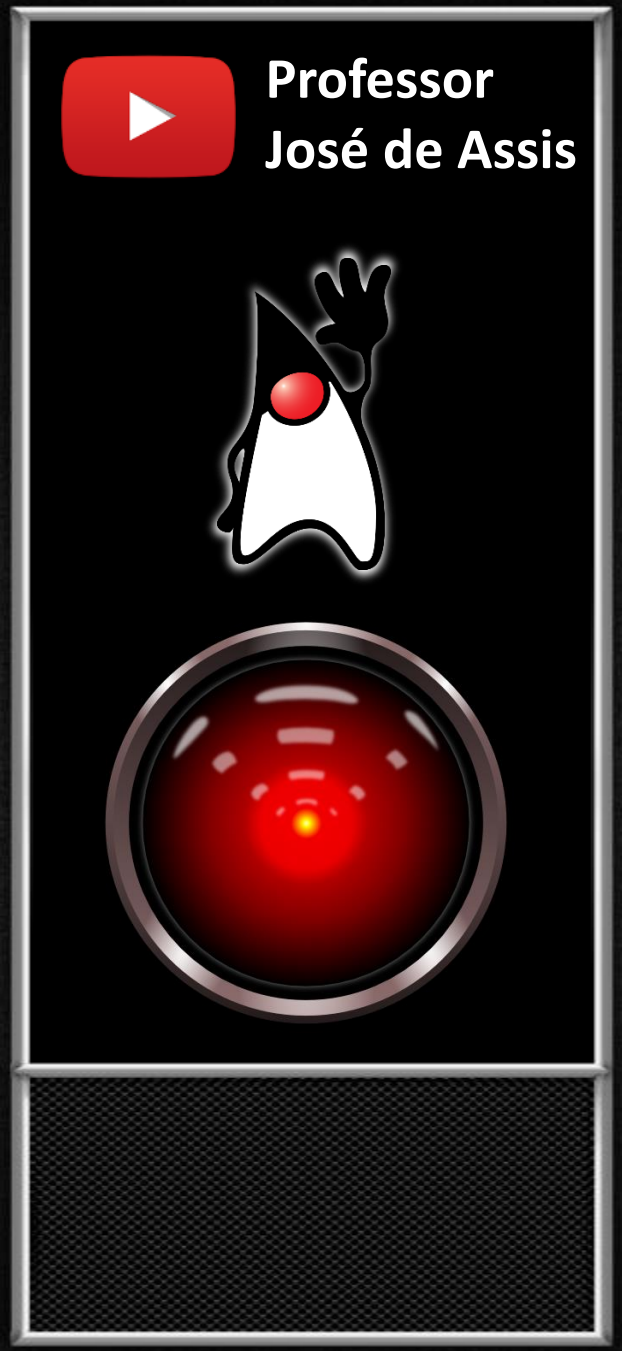
new

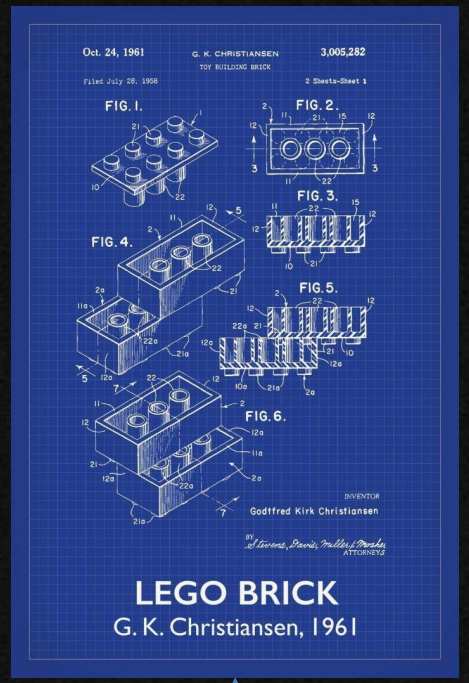
Objetos



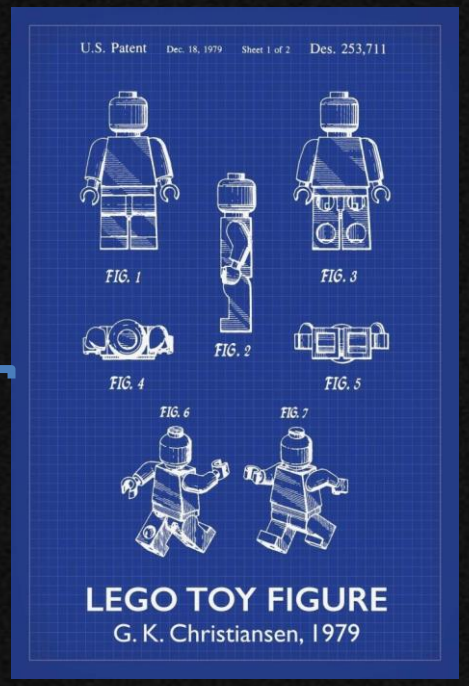
Herança

Na herança criamos uma subclasse da classe modelo estendendo seus atributos e métodos com o objetivo de criar outros tipos de objetos reutilizando atributos e métodos da classe modelo.



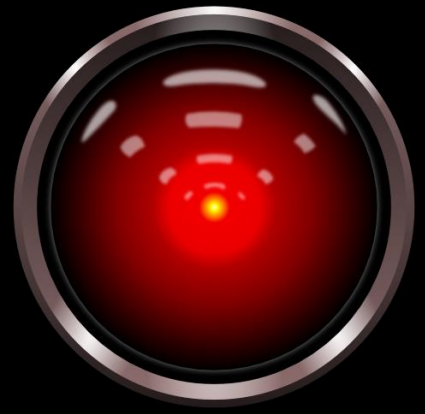


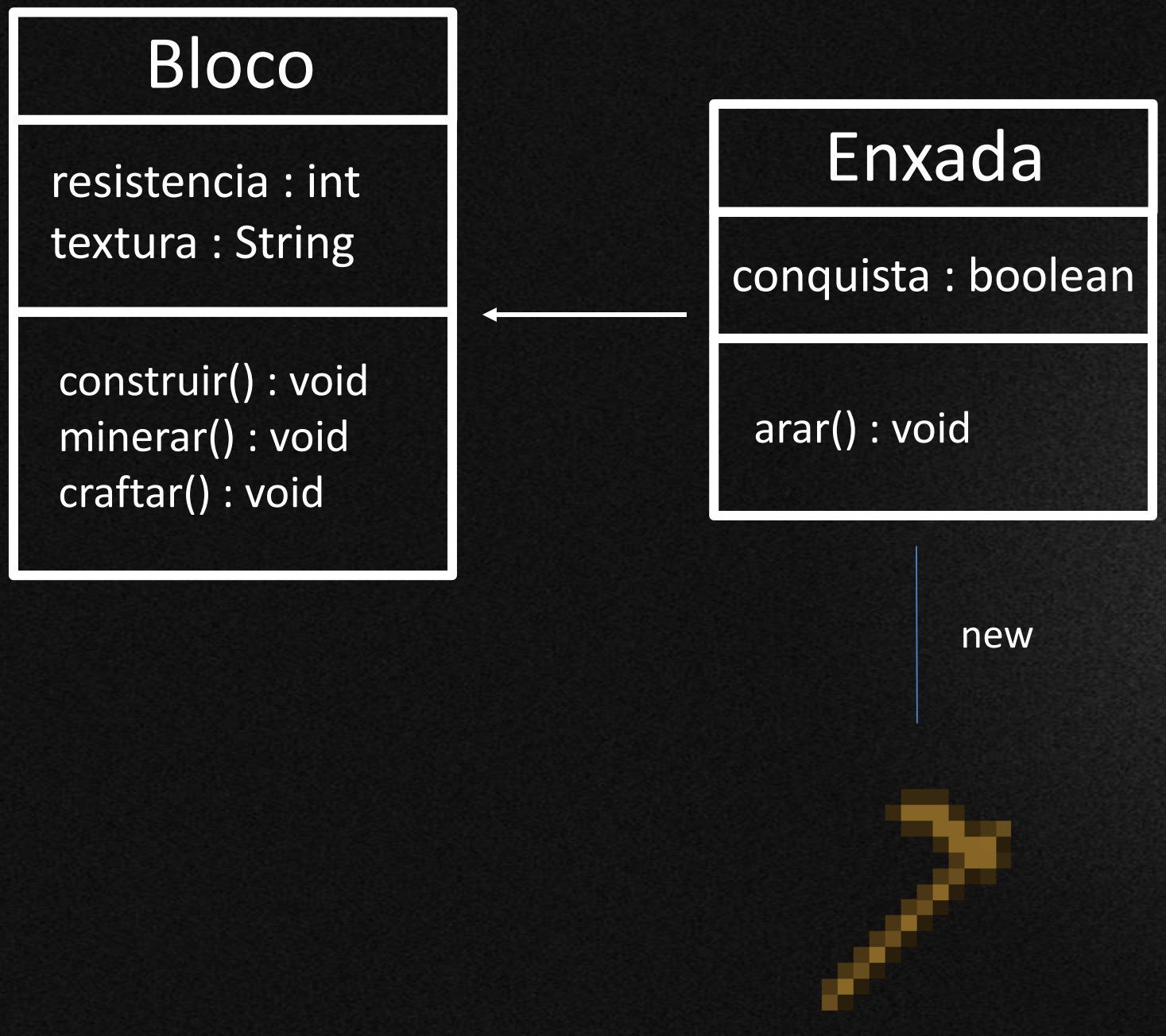
LEGO BRICK
G. K. Christiansen, 1961

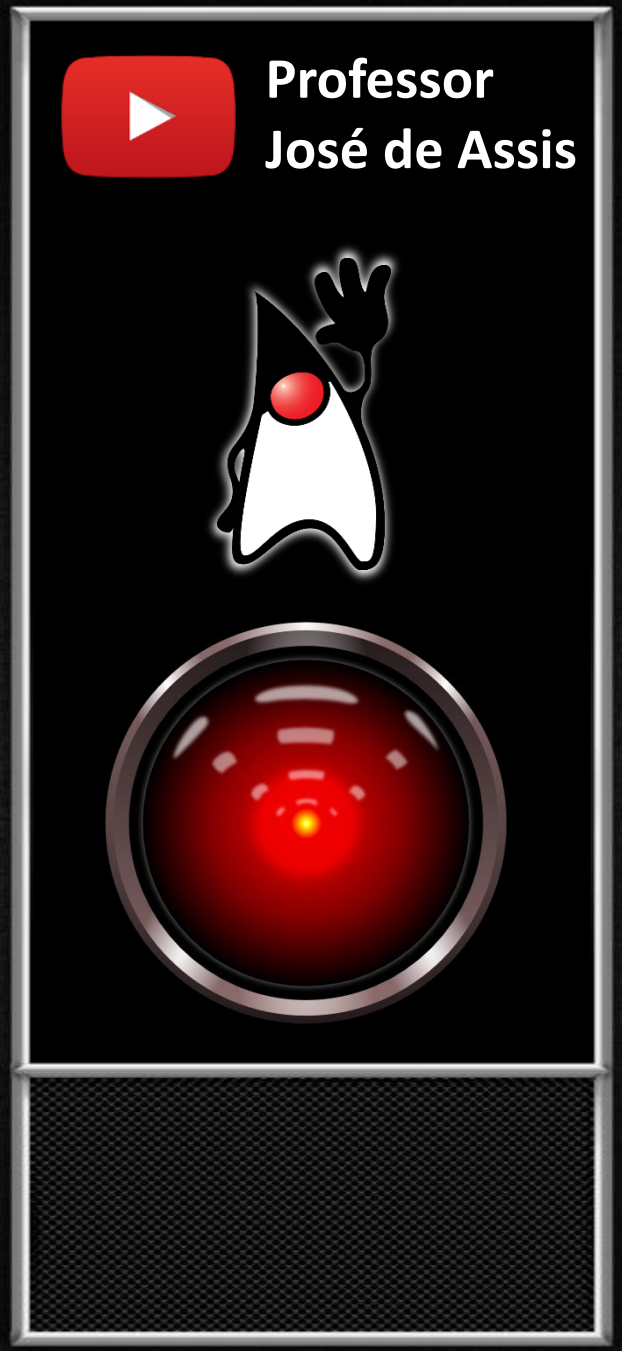


LEGO TOY FIGURE
G. K. Christiansen, 1979

Professor
José de Assis

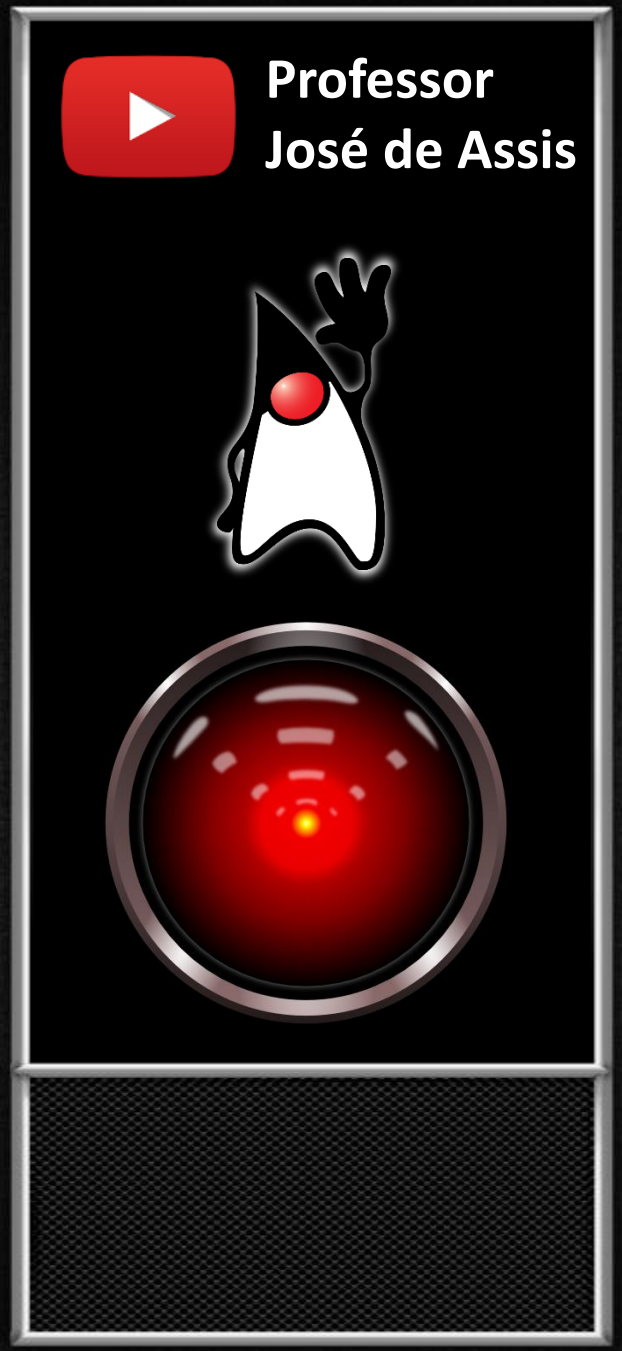



Bloco
resistencia : int
textura : String
construir() : void
minerar() : void
craftar() : void
Enxada
conquista : boolean
arar() : void
new


Professor
José de Assis

Polimorfismo

Polimorfismo é a modificação do comportamento de um método existente na classe modelo.



Bloco

resistencia : int
textura : String

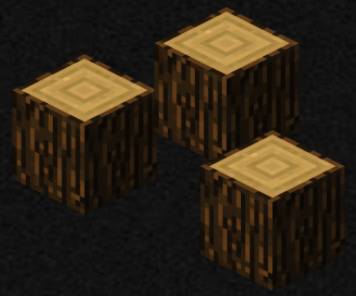
construir() : void
minerar() : void
craftar() : void

Enxada

conquista : boolean

arar() : void

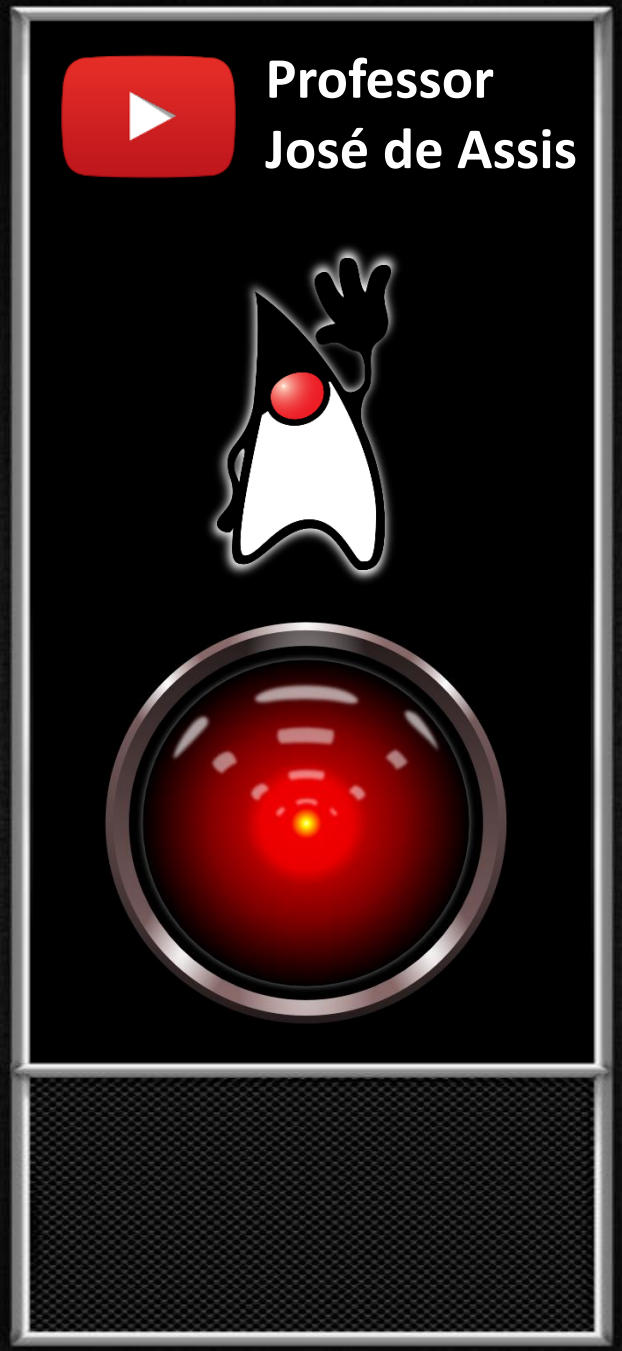
minerar()



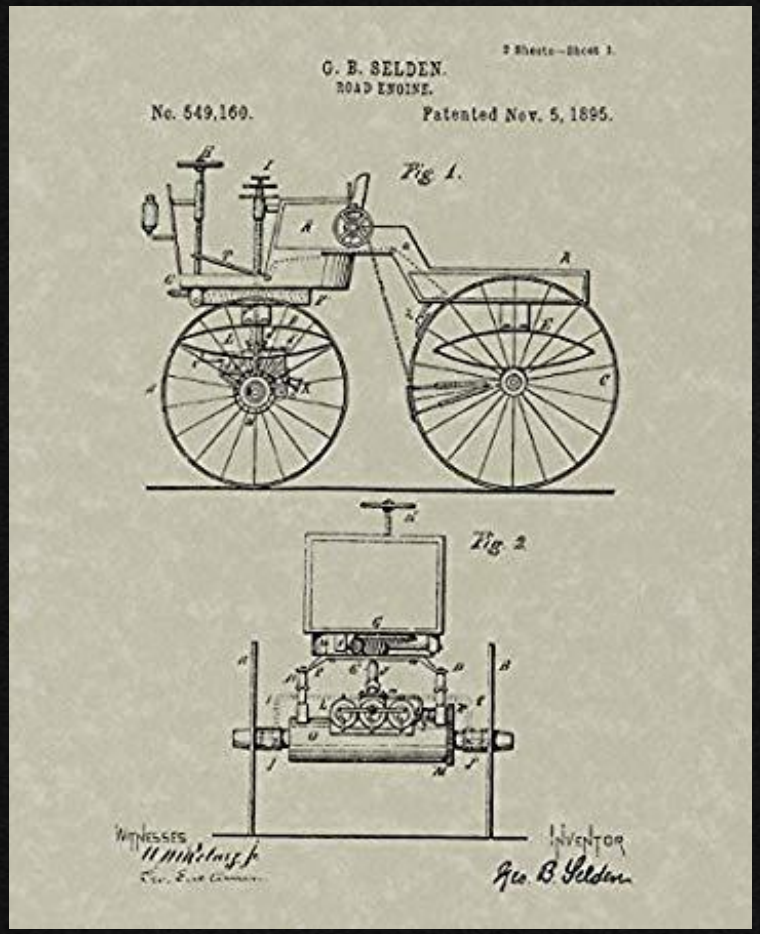
_ - _
Recursos

minerar()

))
Dano



POO – Exercício 1



G. B. SELDEN.
ROAD ENGINE.

No. 549,160. Patented Nov. 5, 1895.

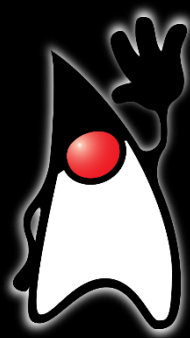
Fig. 1.

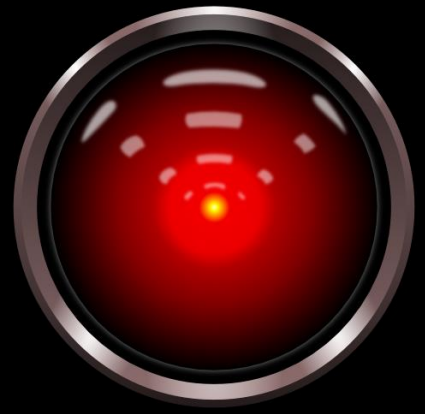
Fig. 2

WITNESSES

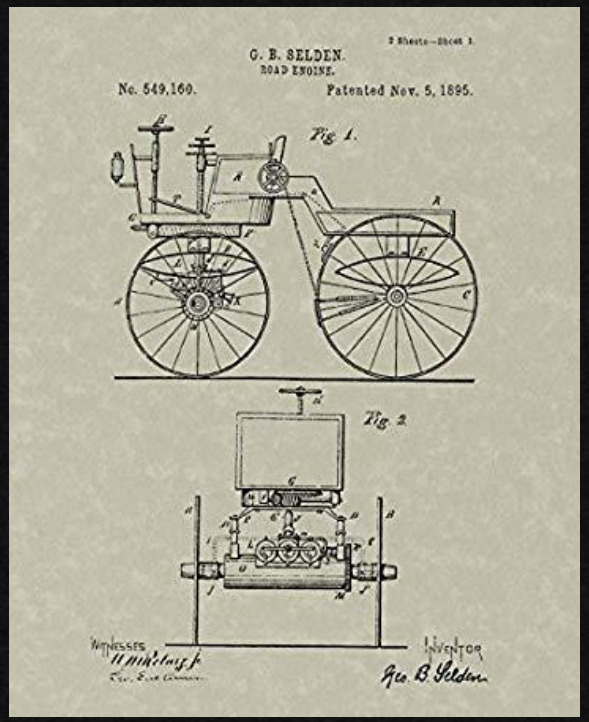
INVENTOR
Geo. B. Selden

Professor
José de Assis





1) Criar a classe modelo

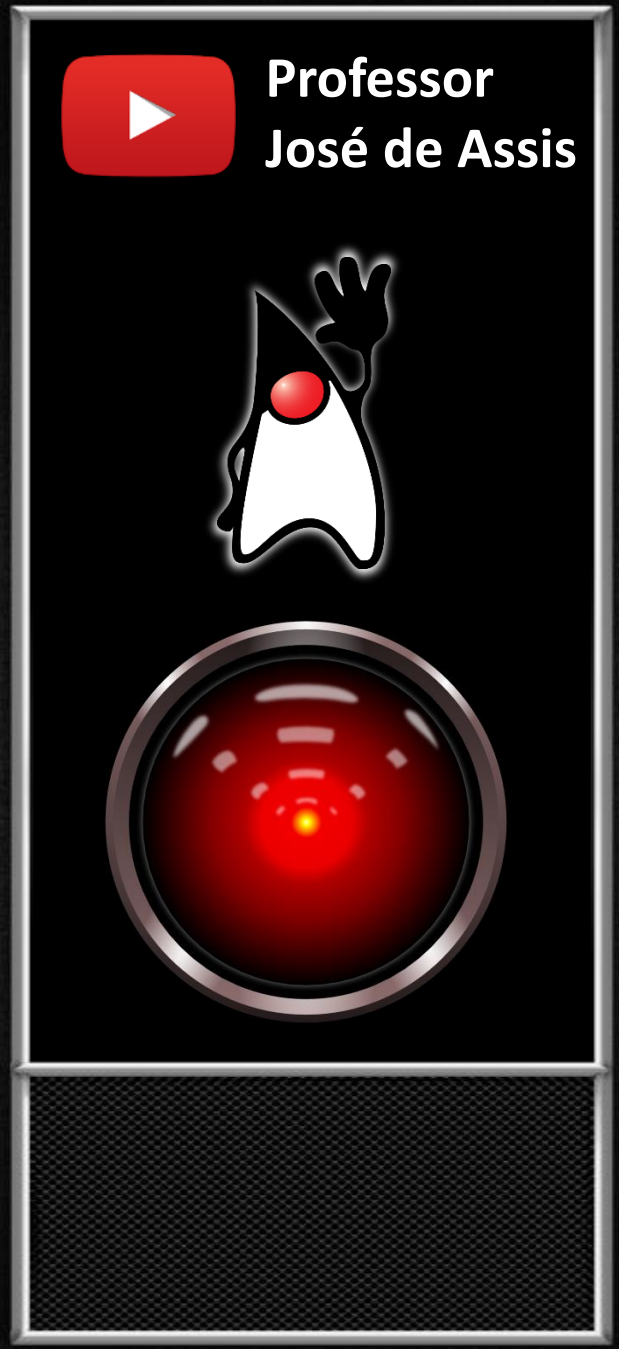


Carro
ano : int cor : String
ligar() : void desligar() : void acelerar() : void

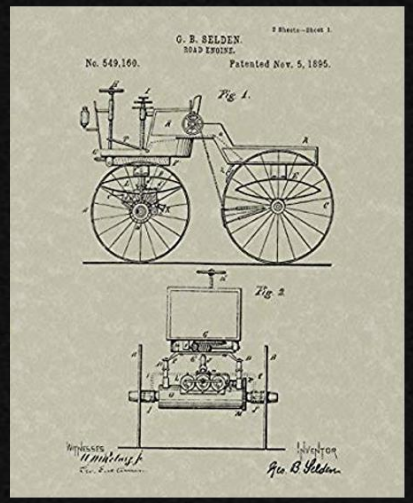
Tipo
Classe

atributos
variáveis

ações
métodos



2) Criar dois objetos(carros)

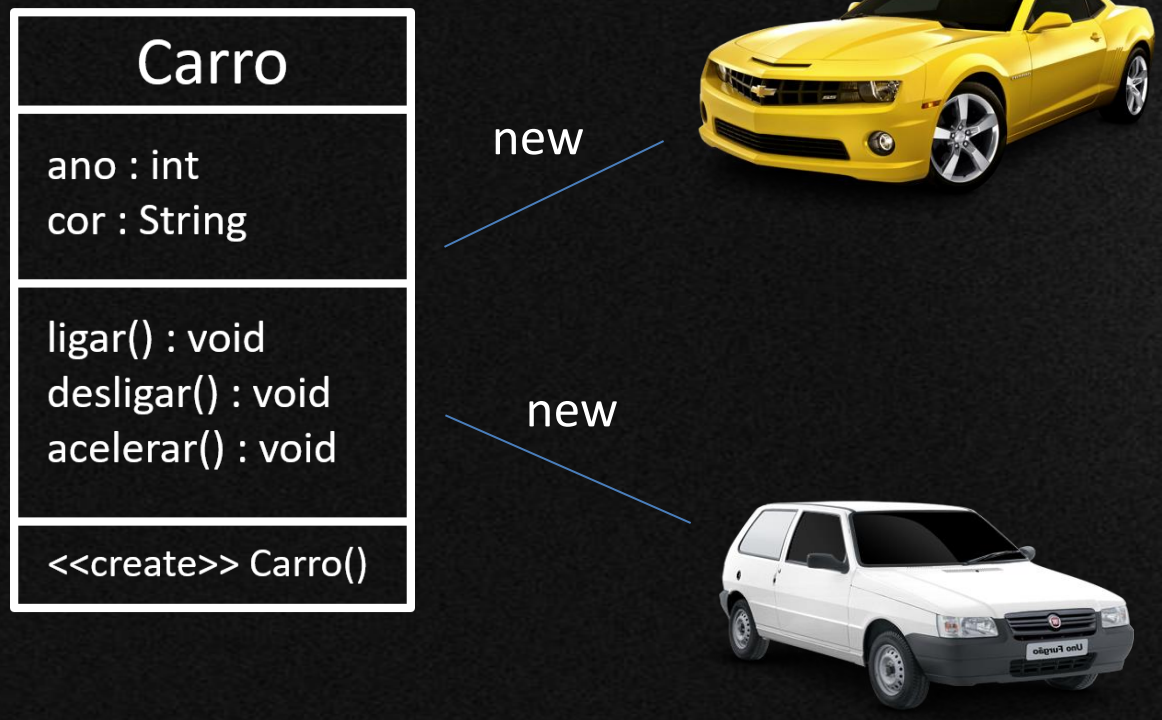


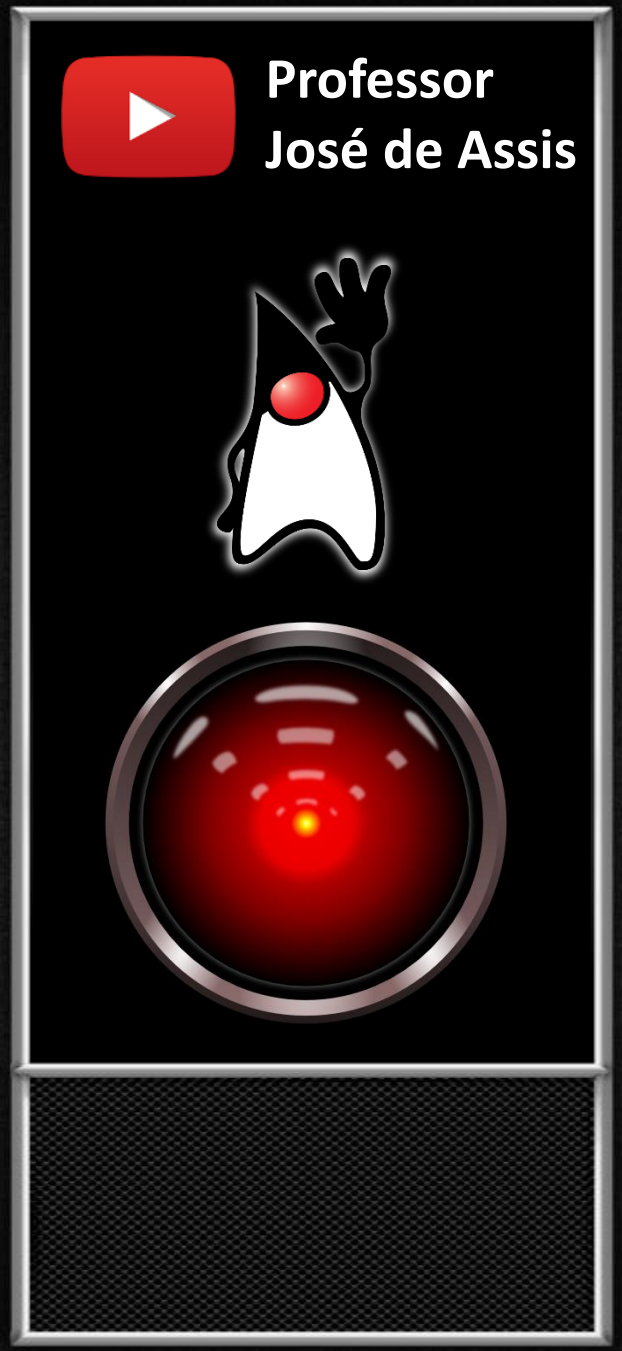
Carro
ano : int cor : String
ligar() : void desligar() : void acelerar() : void

new

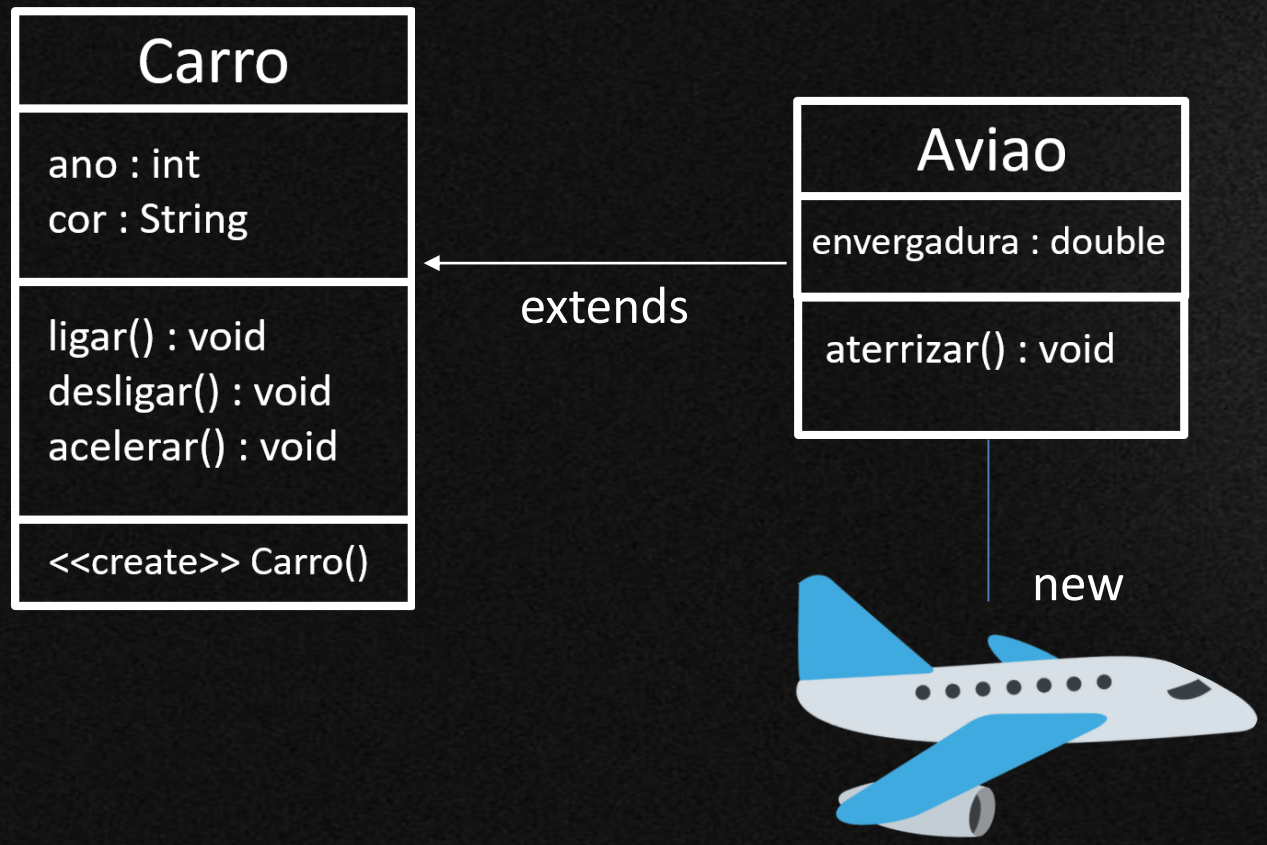
new

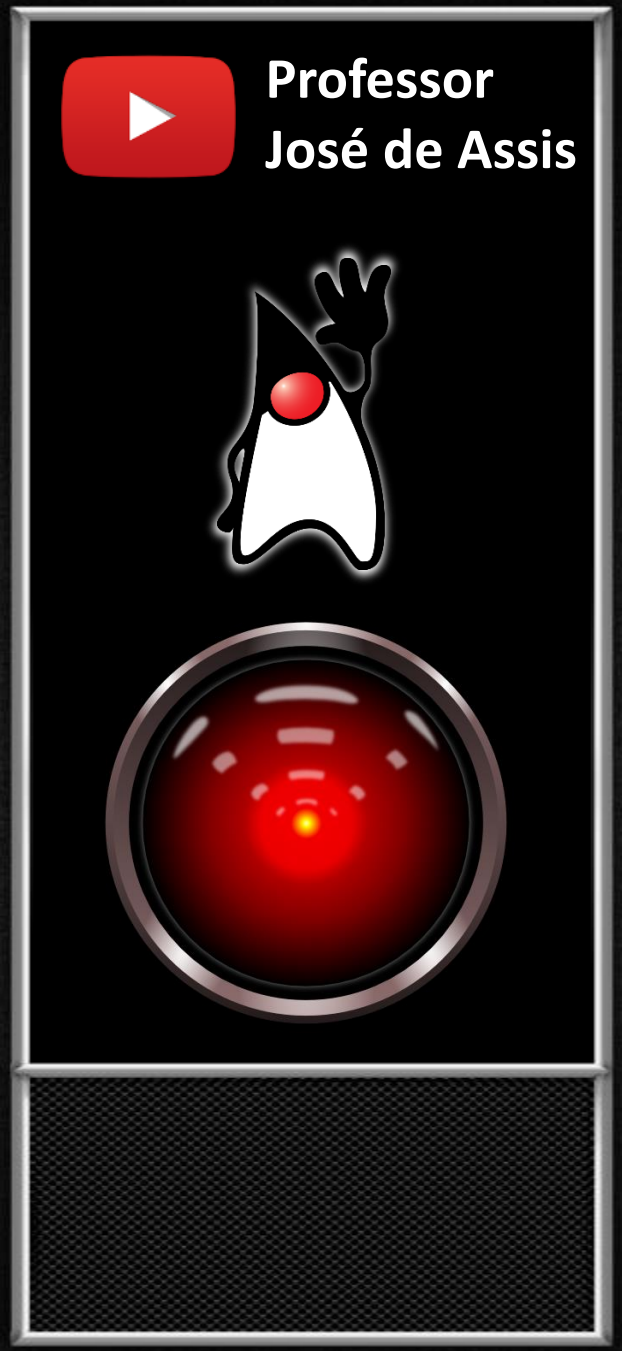
3) Adicionar um construtor que gere aleatoriamente um número de chassi e criar mais dois objetos (carros).



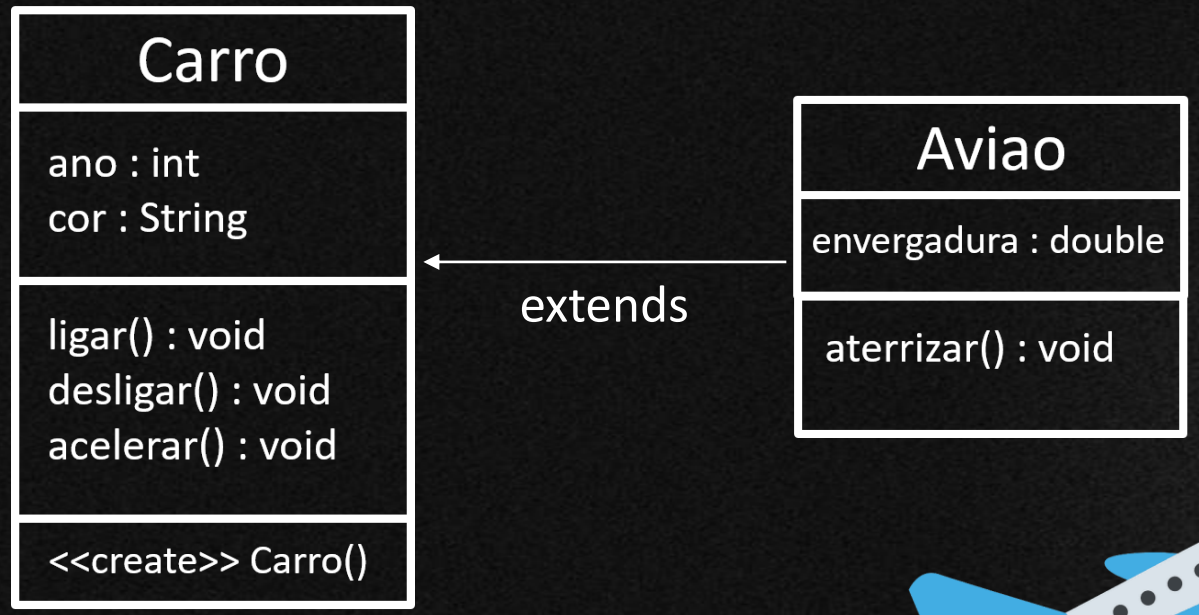


4) Criar uma nova classe modelo, estendendo a classe Carro (herança). Criar um novo objeto (avião).

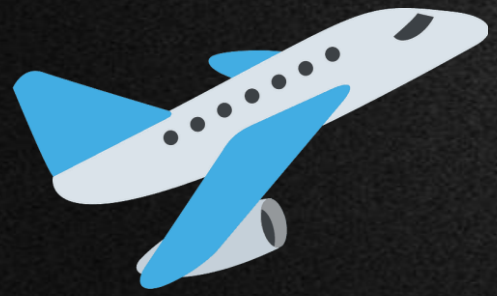




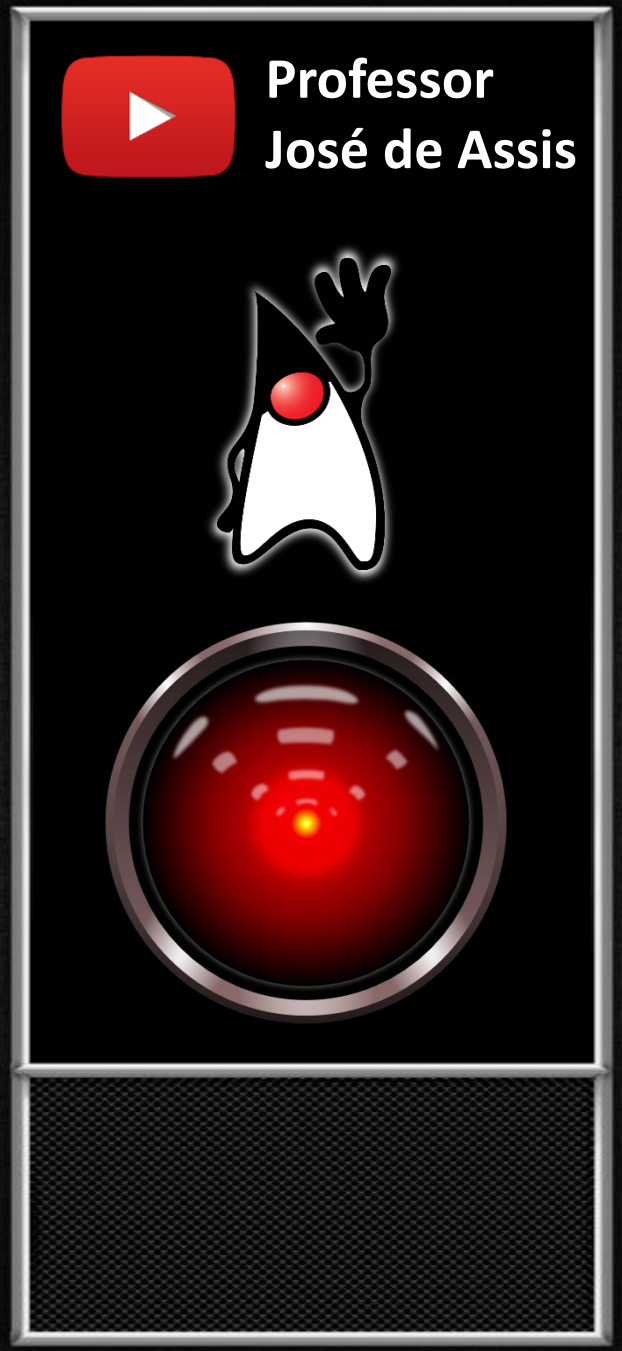
5) Modificar o método acelerar() na classe Aviao (polimorfismo).



acelerar() : void

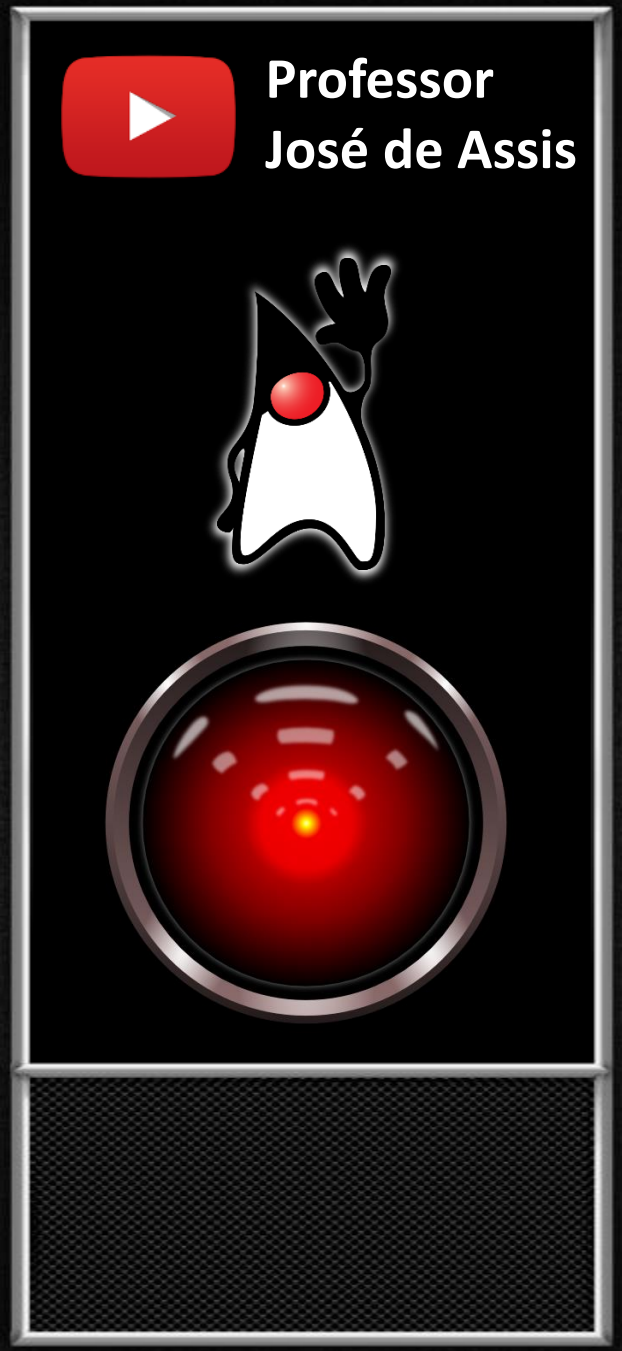


acelerar() : void



Encapsulamento

O principal objetivo do encapsulamento é refinar a segurança do sistema limitando o acesso as variáveis e também impedindo que valores inapropriados sejam atribuídos.



Modificadores de Acesso

private

Acesso permitido somente na própria classe.

“nenhum modificador definido”

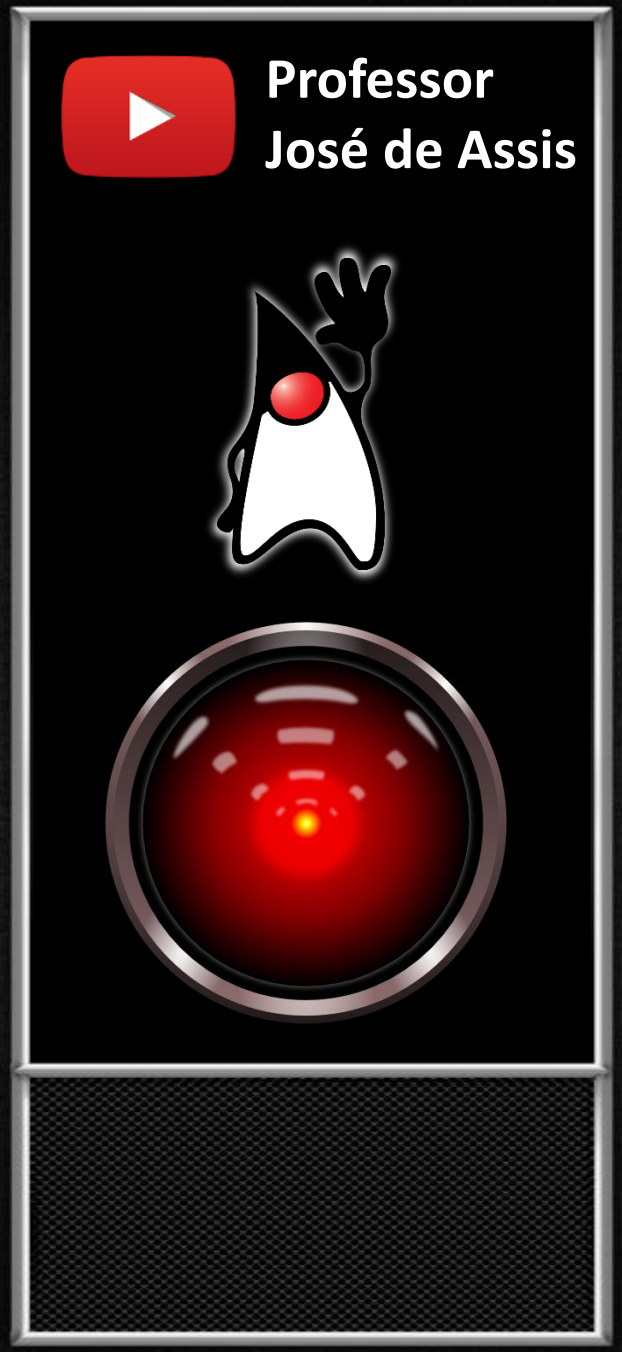
Acesso permitido a todas as classes que pertençam ao mesmo pacote.

protected

Acesso permitido a todas as classes que pertençam ao mesmo pacote.
Acesso permitido a classes de outros pacotes em caso de herança.

public

Acesso permitido a todas as classes de qualquer pacote.

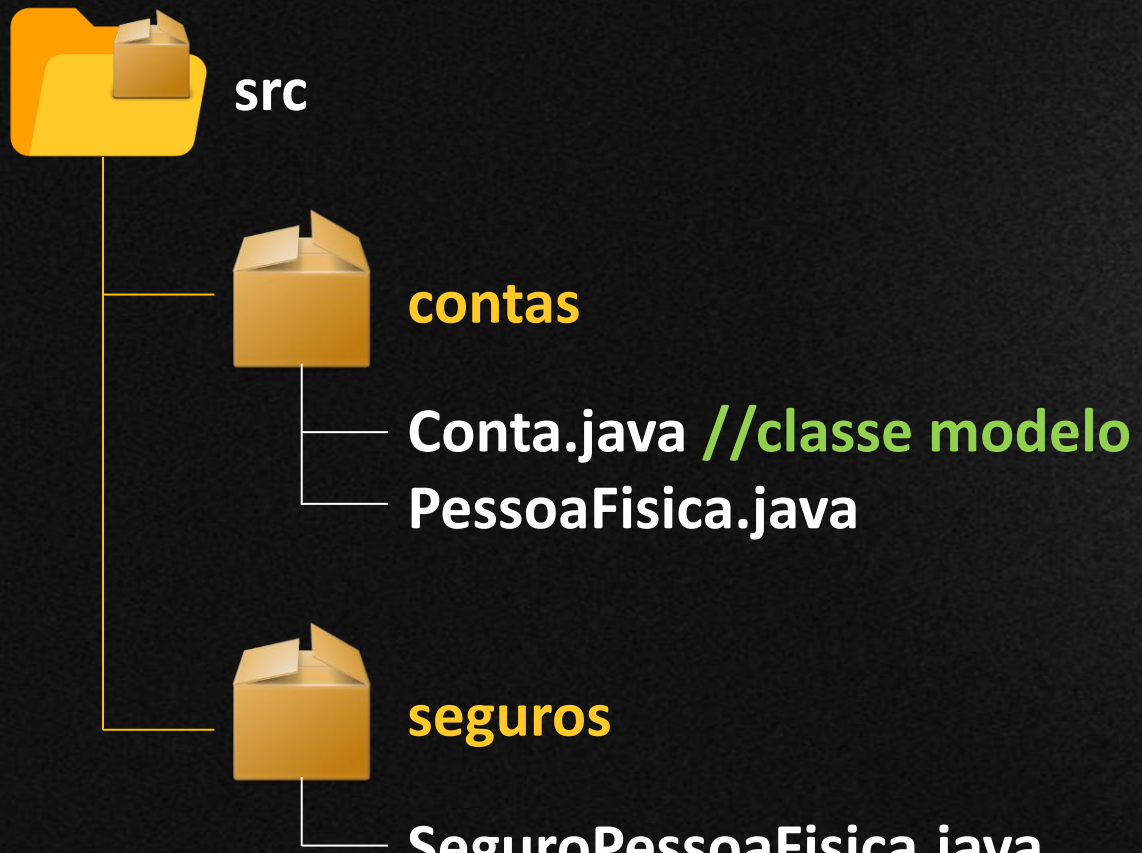


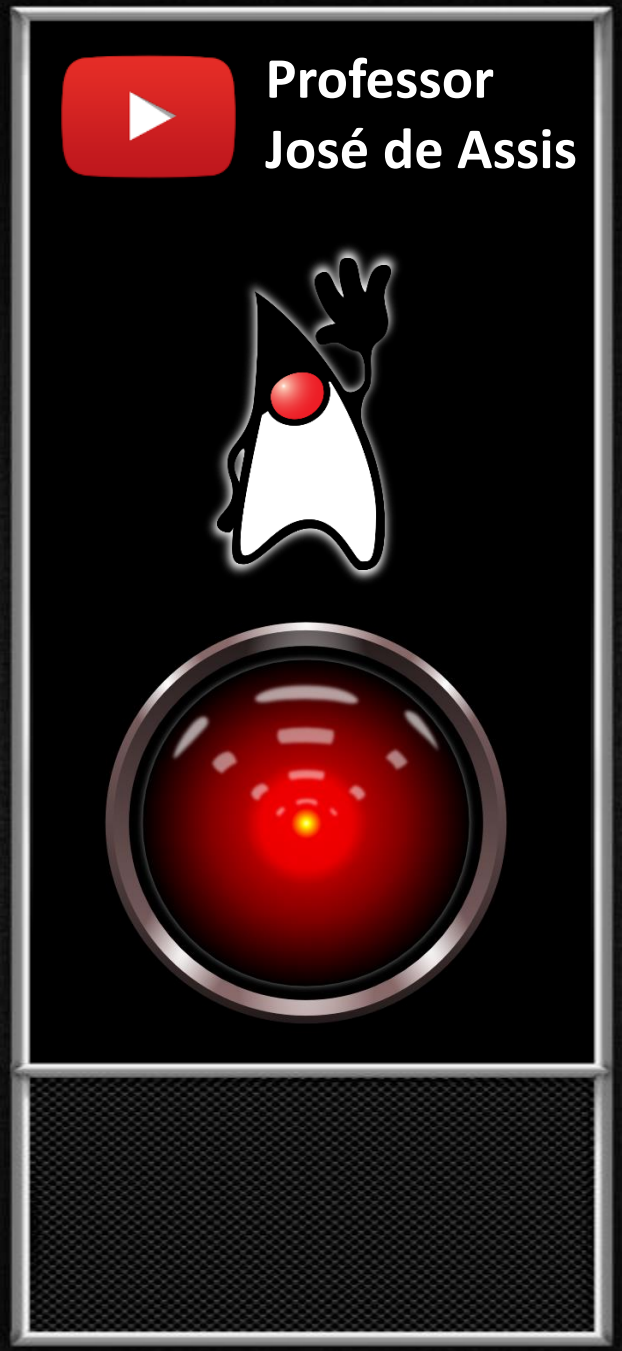
POO – Exercício 2

1) Criar um novo projeto

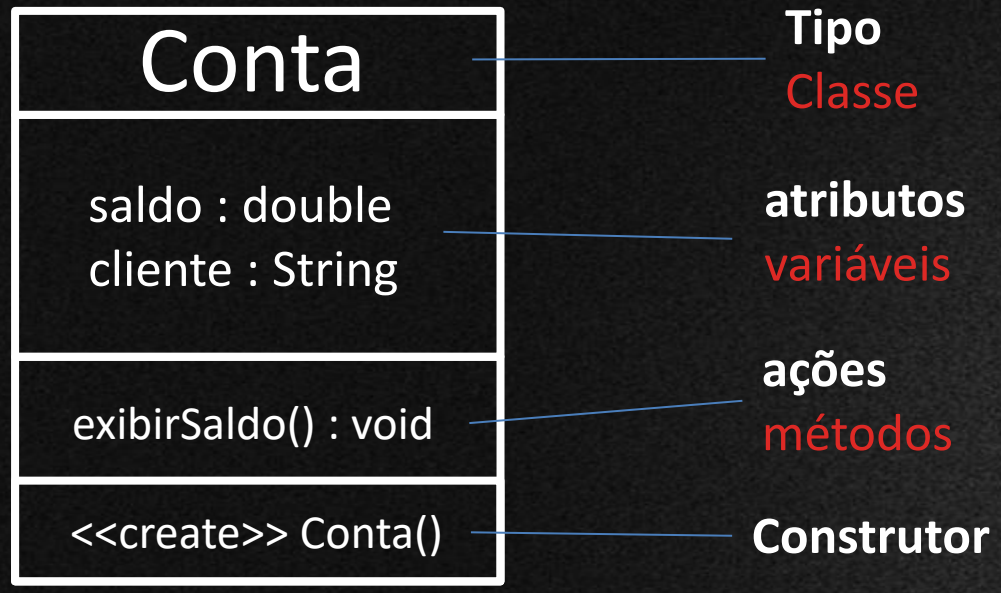
Nome do projeto: Agencia bancaria

Estrutura de pacotes:



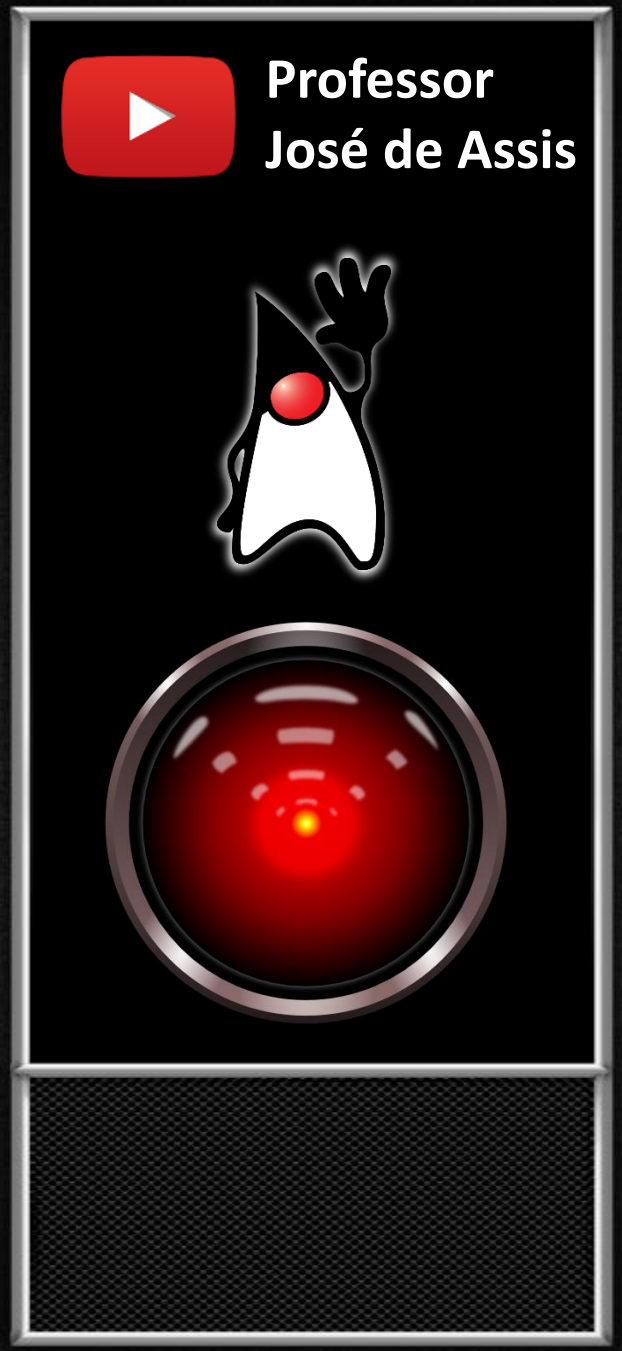


2) Criar a classe modelo

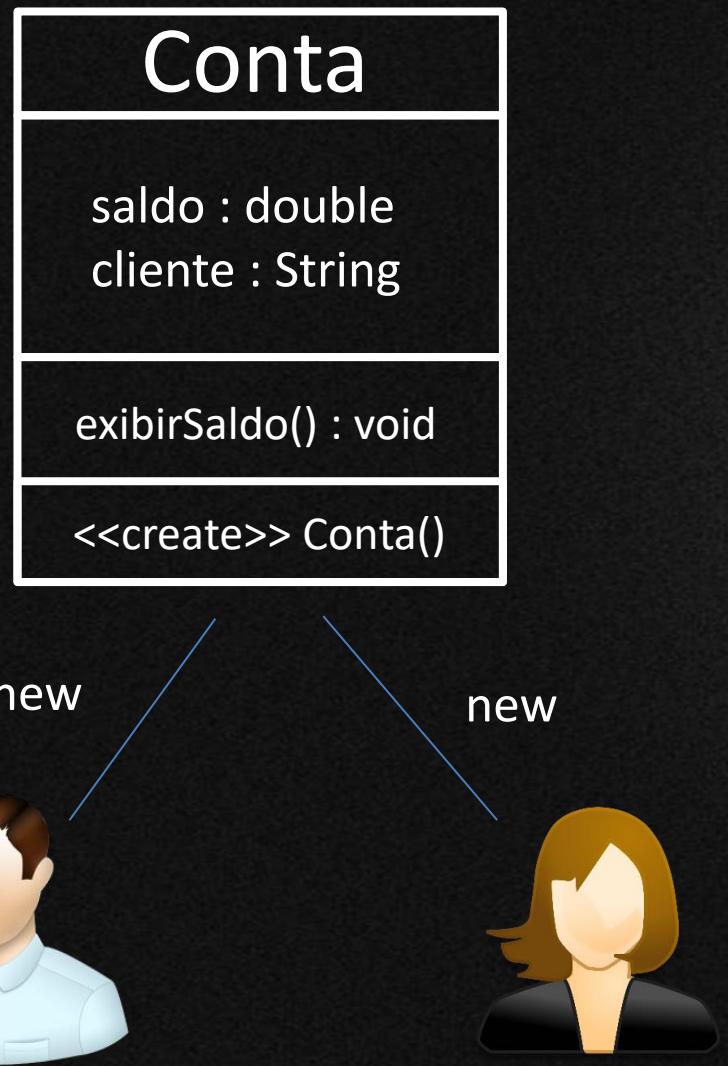


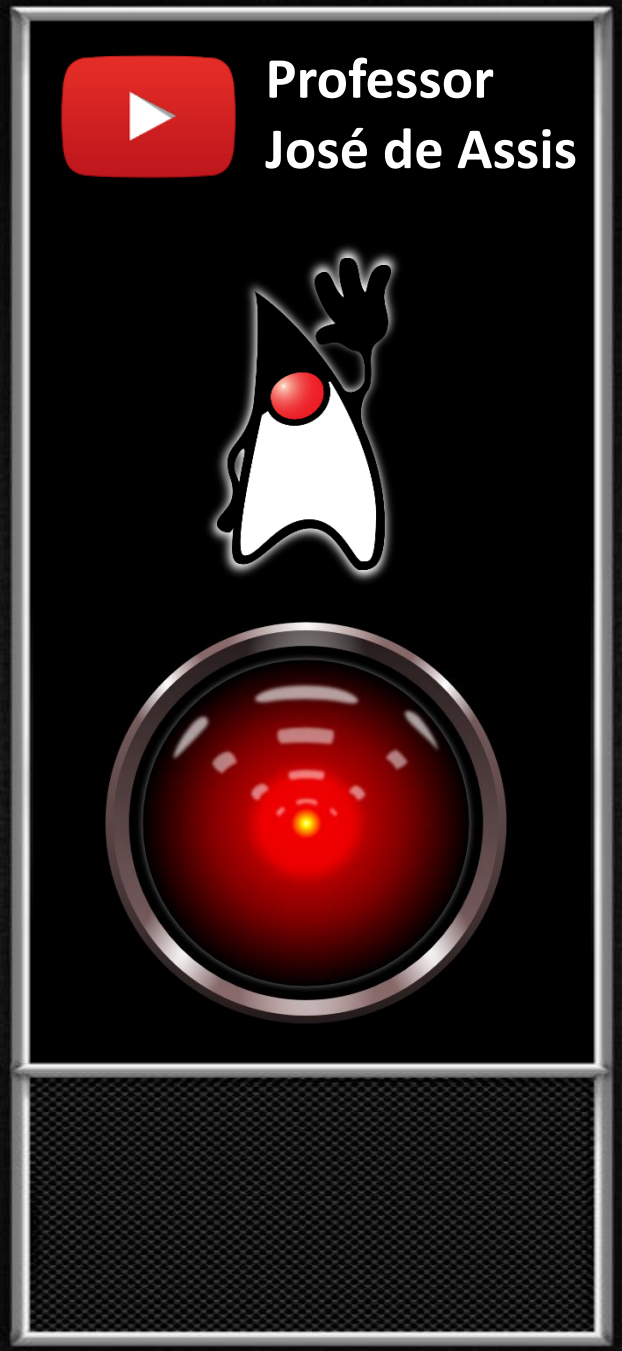
Construtor:
Gerar automaticamente o mesmo número de agência para cada conta criada.

Método exibirSaldo():
Quando executado deverá exibir automaticamente o saldo da conta corrente do cliente.

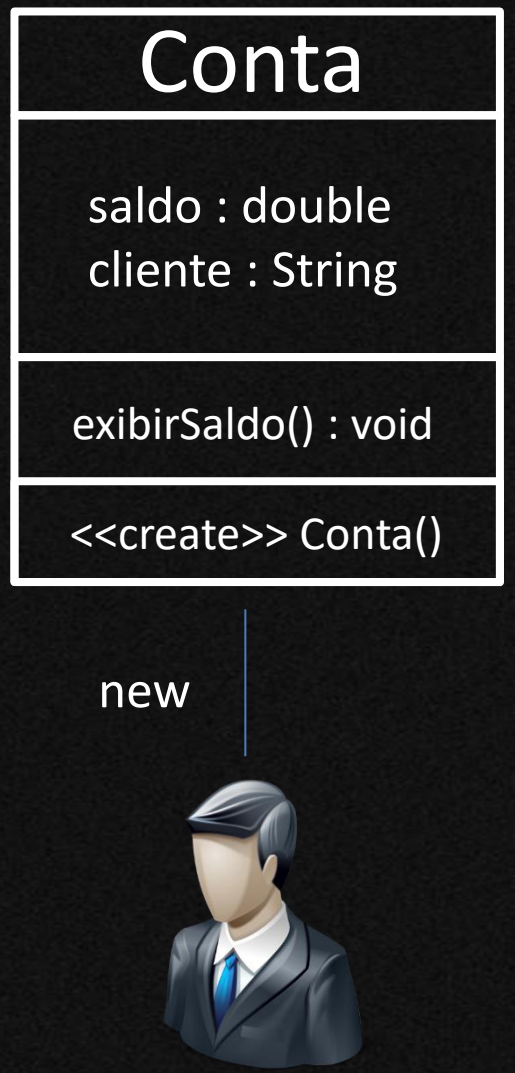


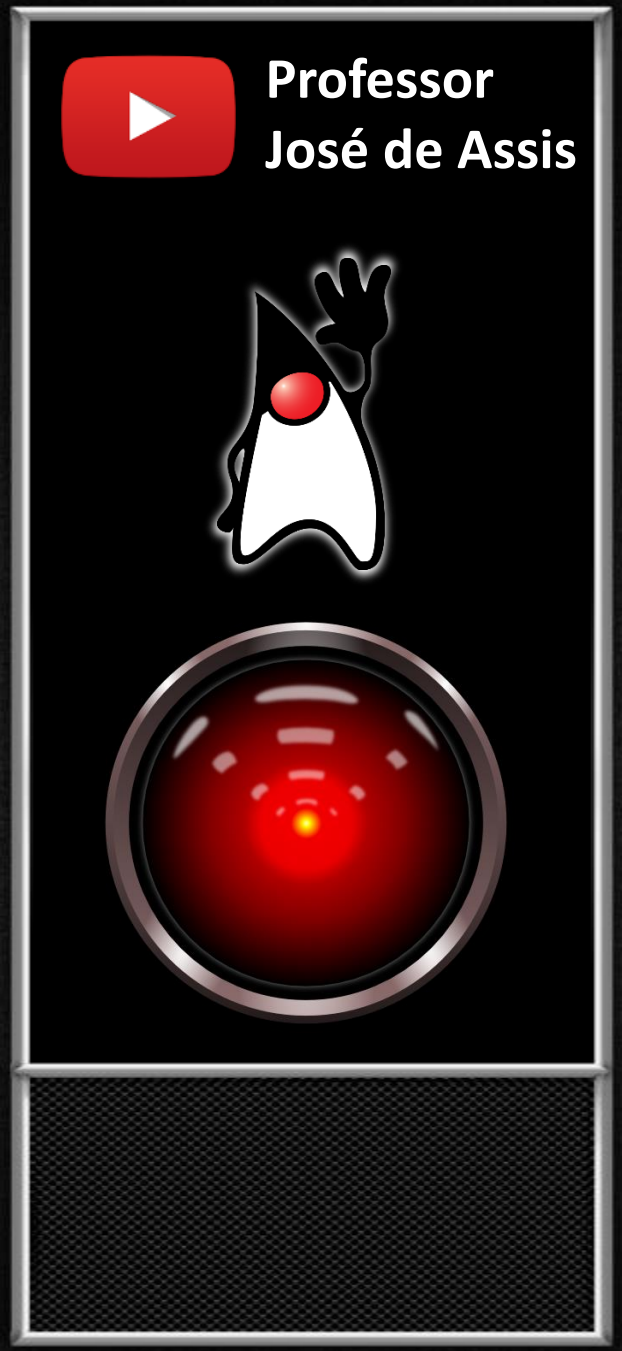
3) Criar 2 objetos (contas) na classe PessoaFisica





4) Criar 1 objeto (conta) na classe SeguroPessoaFisica

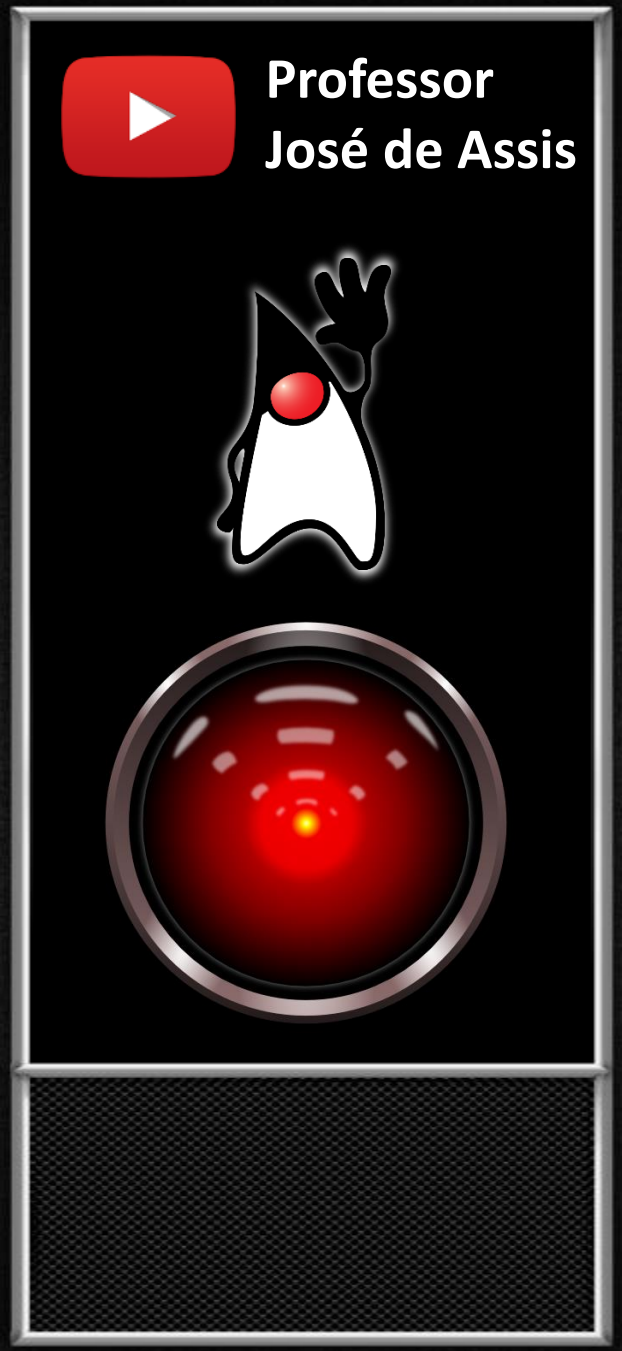




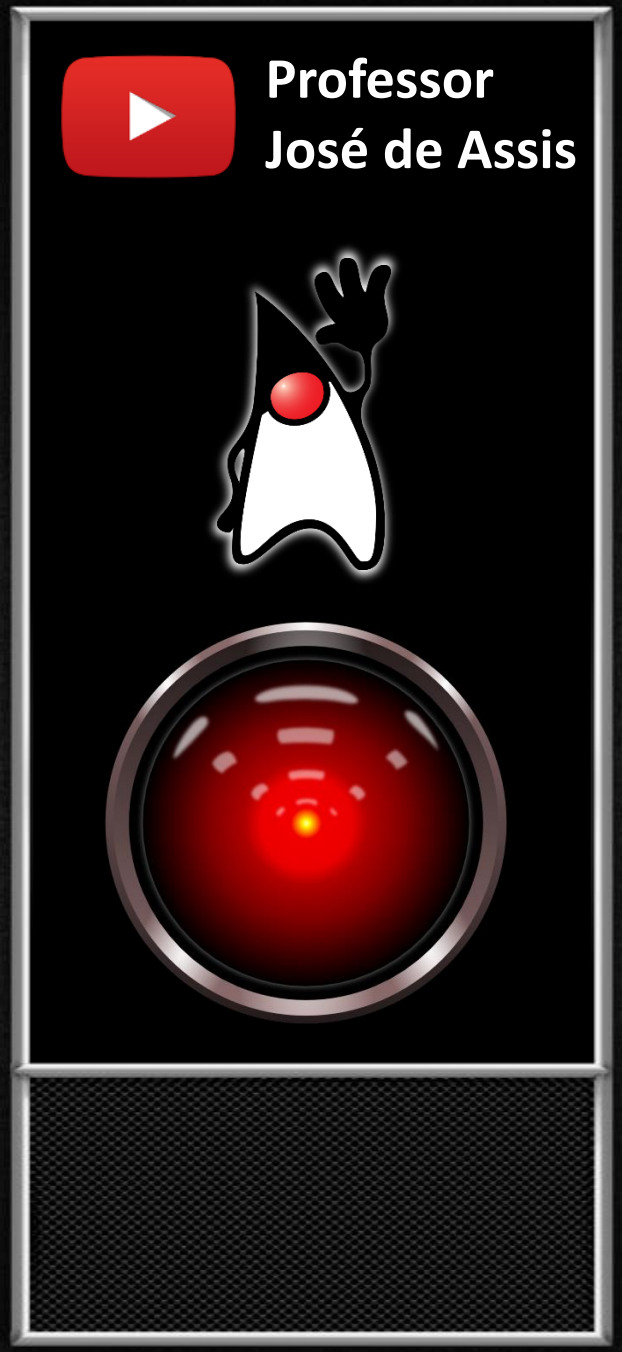
5) Criar métodos para depósito, saque e transferência de valores entre contas.

6) Criar um relatório gerencial para saber o total nas contas de pessoa física.

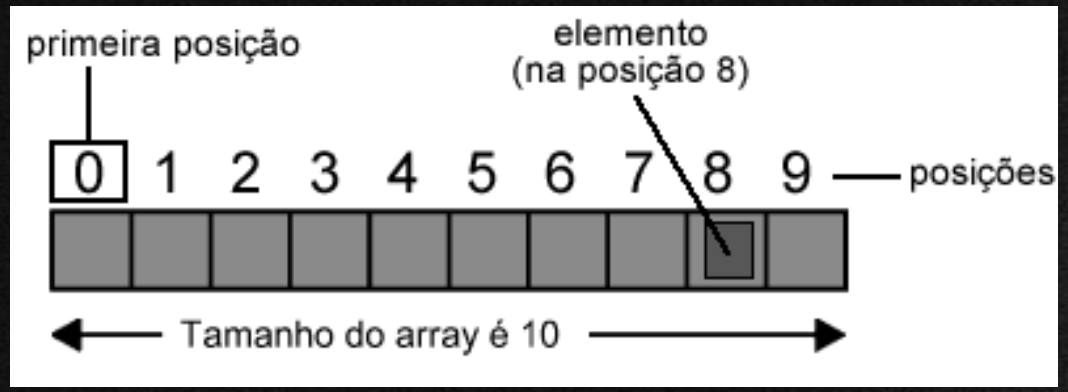
7) Reforçar a segurança encapsulando as variáveis da classe conta.



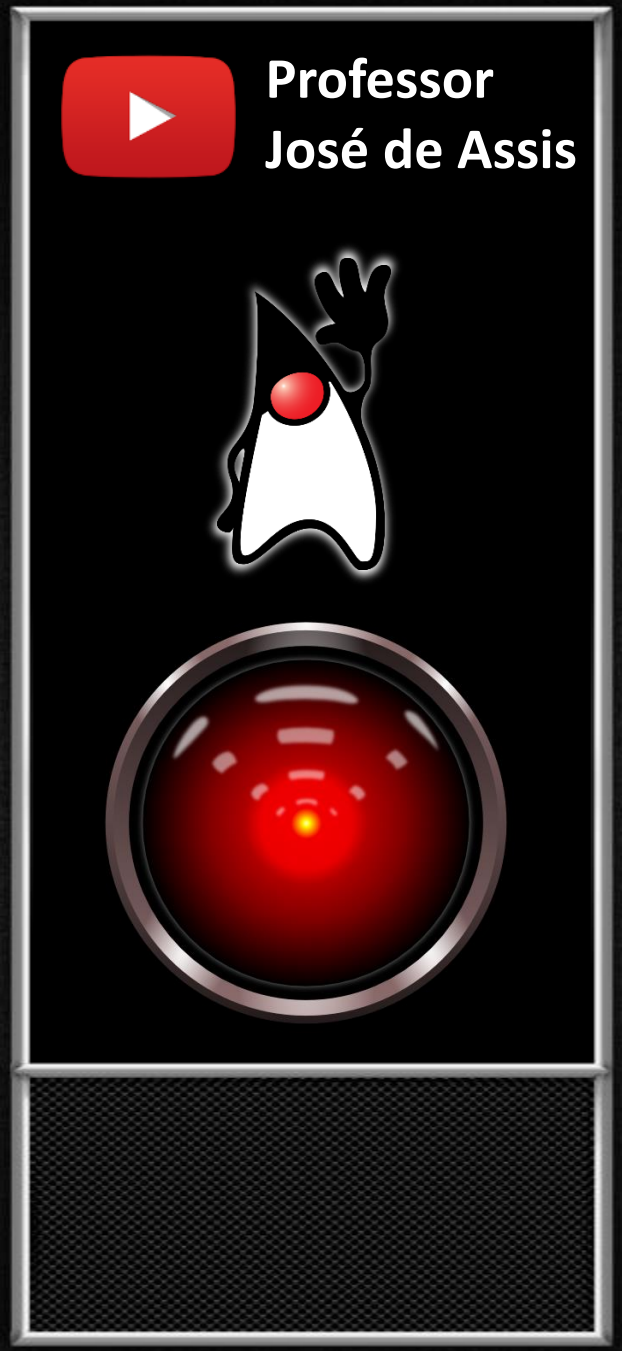
ARRAY



ARRAY (Vetor) é uma estrutura de dados indexada que permite armazenar múltiplos valores.

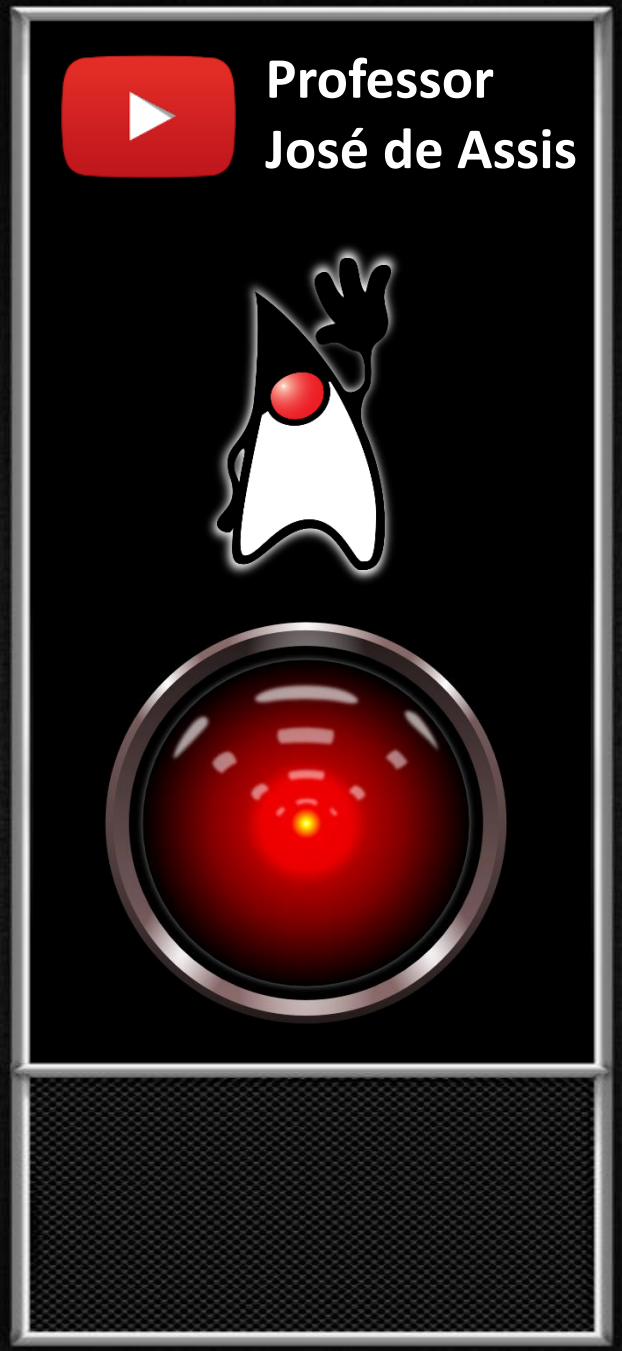


ARRAY - Vetor (estático)
ARRAY LIST - Vetor (dinâmico)
ARRAY MULTIDIMENSIONAL - Matriz



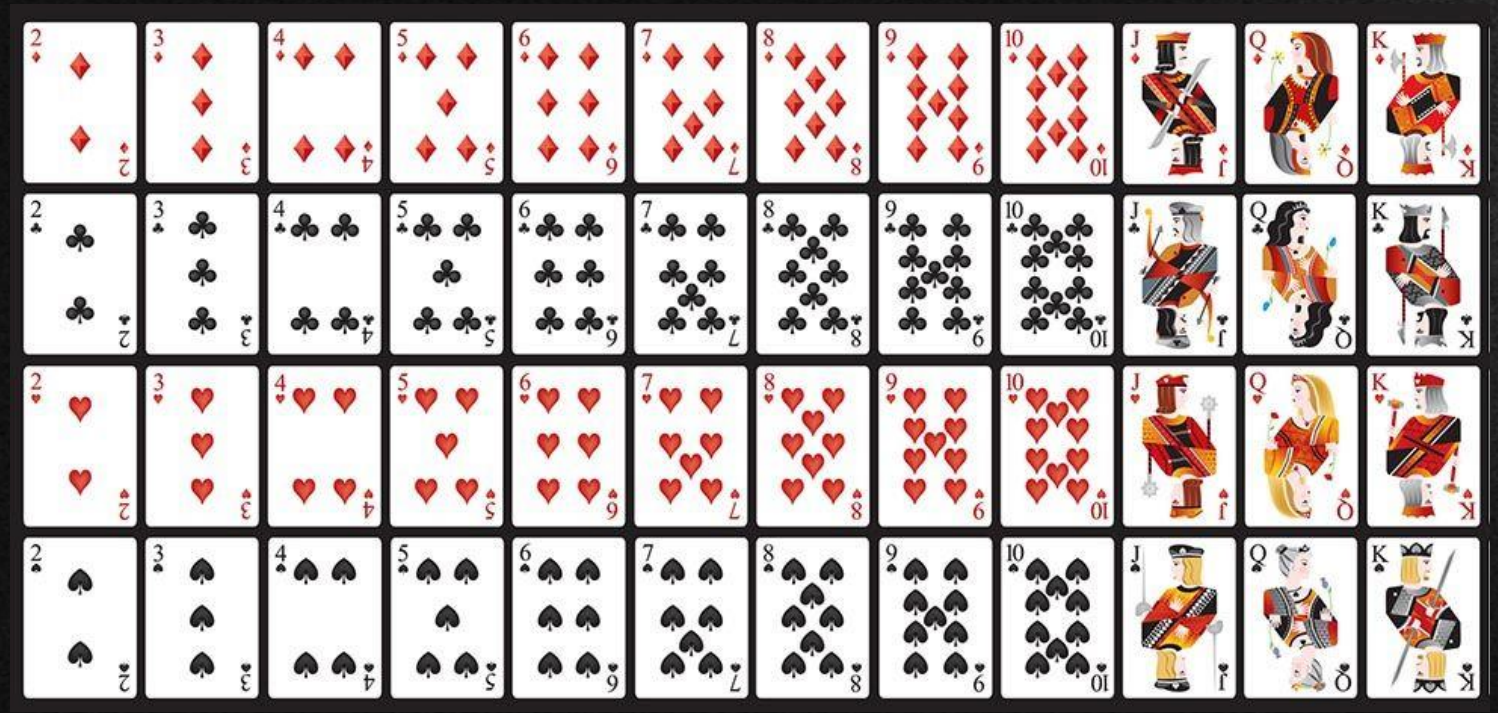
ARRAY

```
String[ ] carros = { "ferrari", "fusca", "camaro", "uno" };

char[ ] controle = { 'w', 'a', 's', 'd' };

int[ ] pares = { 2, 4, 6, 8 };

double[ ] imc = { 18.5, 24.9, 26, 29.1 };
```

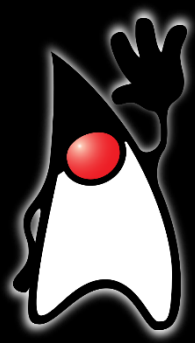


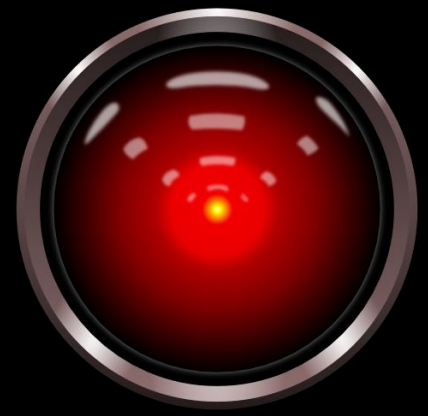
EXERCÍCIO

Desenvolva uma aplicação no console (terminal) para recuperar uma carta específica de um baralho de 52 cartas usando dois arrays (nipes e faces).



Professor
José de Assis

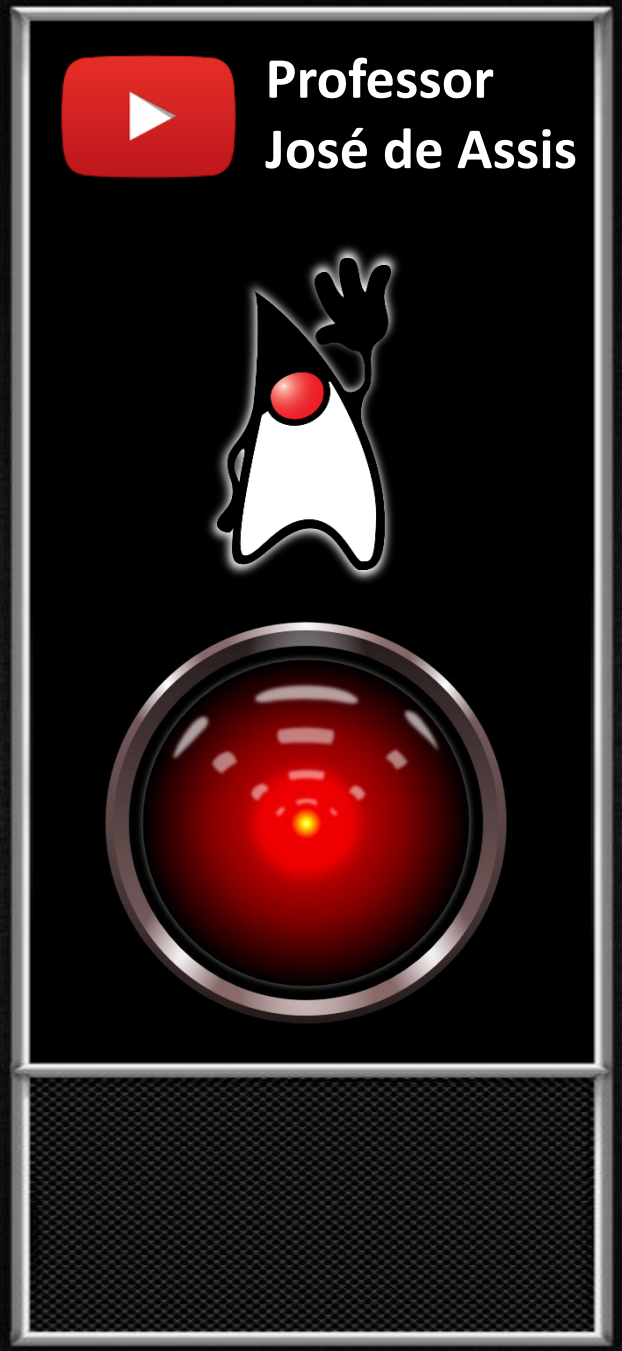




ArrayList

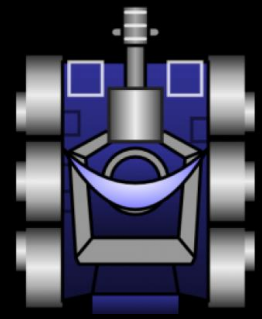
ArrayList é um vetor cujo tamanho pode variar de forma dinâmica de acordo com o tempo.

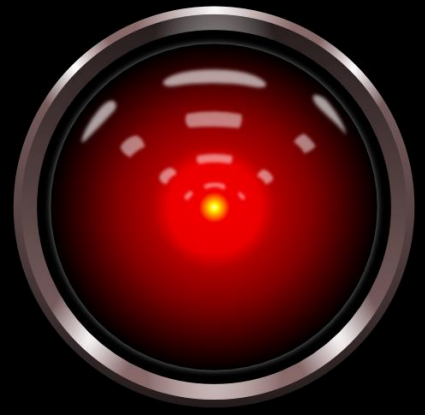
```
// importação
import java.util.ArrayList;

// sintaxe
ArrayList<String> contatos = new ArrayList<>();
```



Robocode

Professor
José de Assis





Robocode: Turn 537, Round 1 of 10, 29 TPS, 30 FPS, Used mem: 30 of 455 MB

Battle Robot Options Help

Corners
Crazy
Fire
Interactive
MyFirstJuniorRo...
MyFirstRobot
PaintingRobot
RamFire
SittingDuck
SpinBot
Target
Tracker
TrackFire
VelociRobot
Walls
Main battle log

Pause/Debug Next Turn Stop Restart 30

Robot Editor

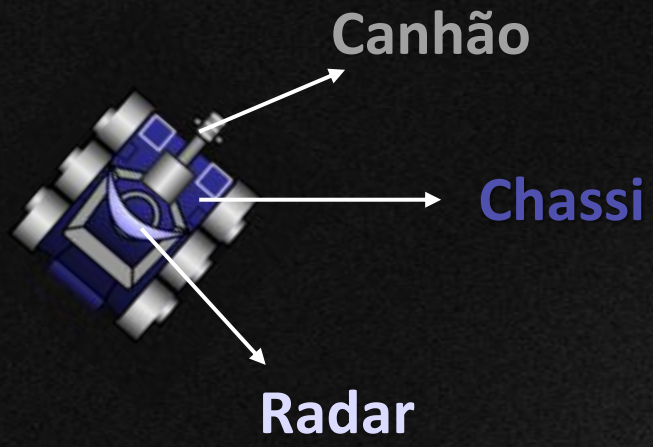
File Edit View Compiler Window Help

Editing - C:\robocode\robots\sample\MyFirstRobot.java

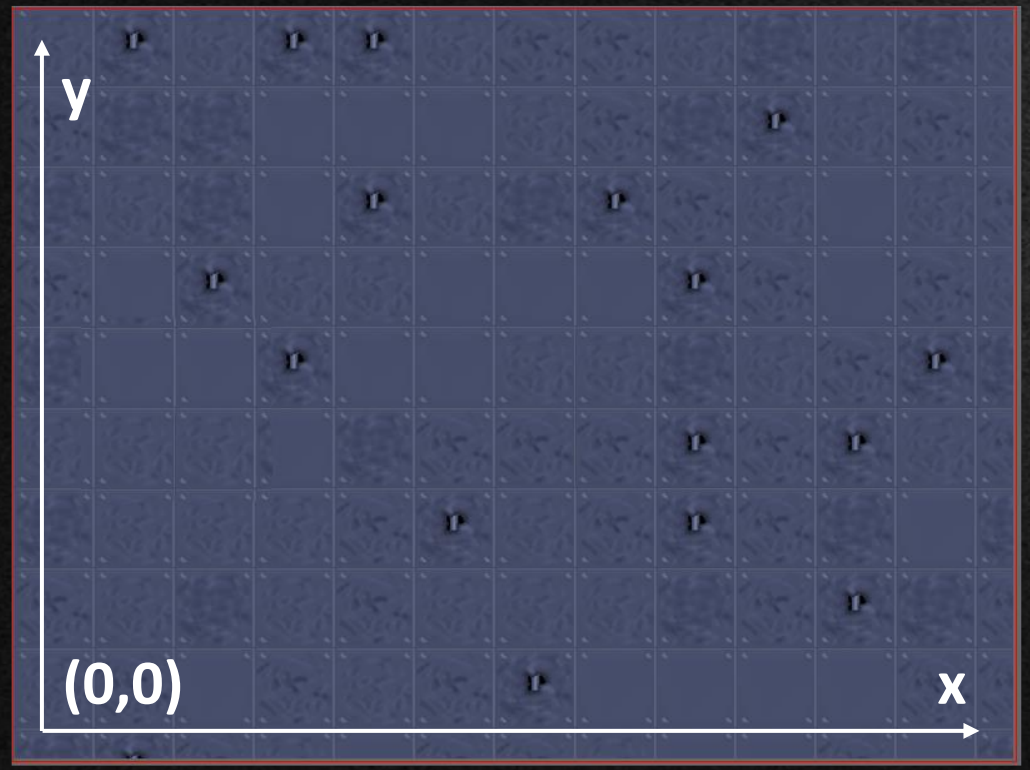
```
1  package sample;
2
3  import robocode.*;
4
5  public class MyFirstRobot extends Robot {
6
7      // Main method
8      public void run() {
9          while (true) {
10             ahead(100); // Move ahead 100
11             turnGunRight(360); // Spin gun around
12             back(100); // Move back 100
13             turnGunRight(360); // Spin gun around
14         }
15     }
16
17     // What to do when thit robot scan another robot
18     public void onScannedRobot(ScannedRobotEvent e) {
19         fire(1);
20     }
21
22     // What to do when this robot are hit by a robot
23     public void onHitByBullet(HitByBulletEvent e) {
24         turnLeft(90 - e.getBearing());
25     }
26 }
```

Line: 19 Column: 11

Anatomia do Tank



Campo de batalha



Funcionamento

threads

public void run()

Gerenciador de batalhas

Subsistema de renderização

Eventos

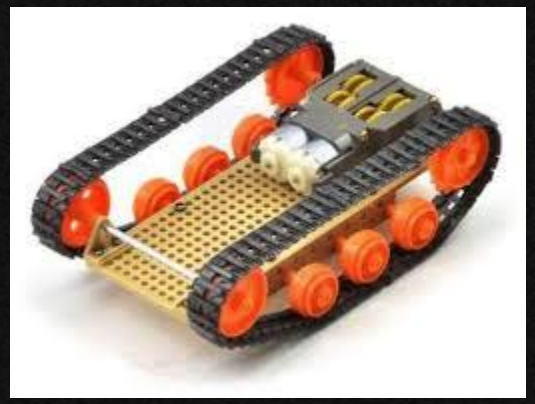
Customização

import java.awt.Color;

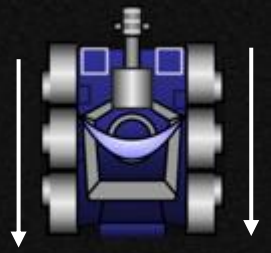
setColors (Color.chassi, Color.canhão, Color.radar, Color.bala, Color.arco)

black	blue	cyan	darkGray	gray	green
lightGray	magenta	orange	red	white	yellow

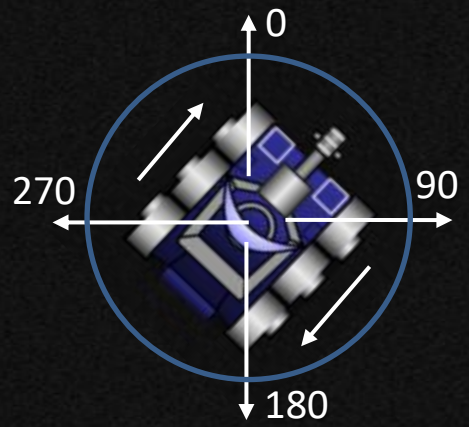
Movimento do chassi robótico



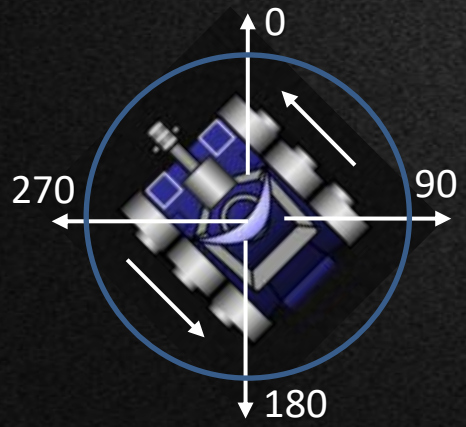
ahead(100)



back(30)

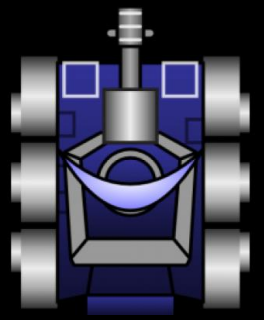


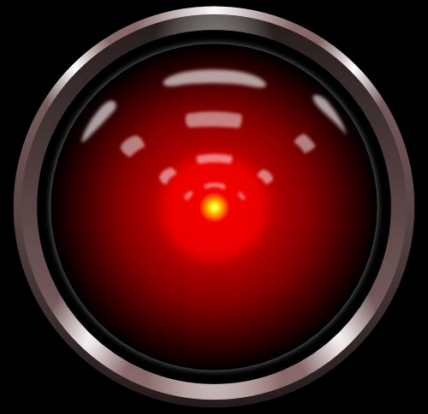
turnRight(45)



turnLeft(90)

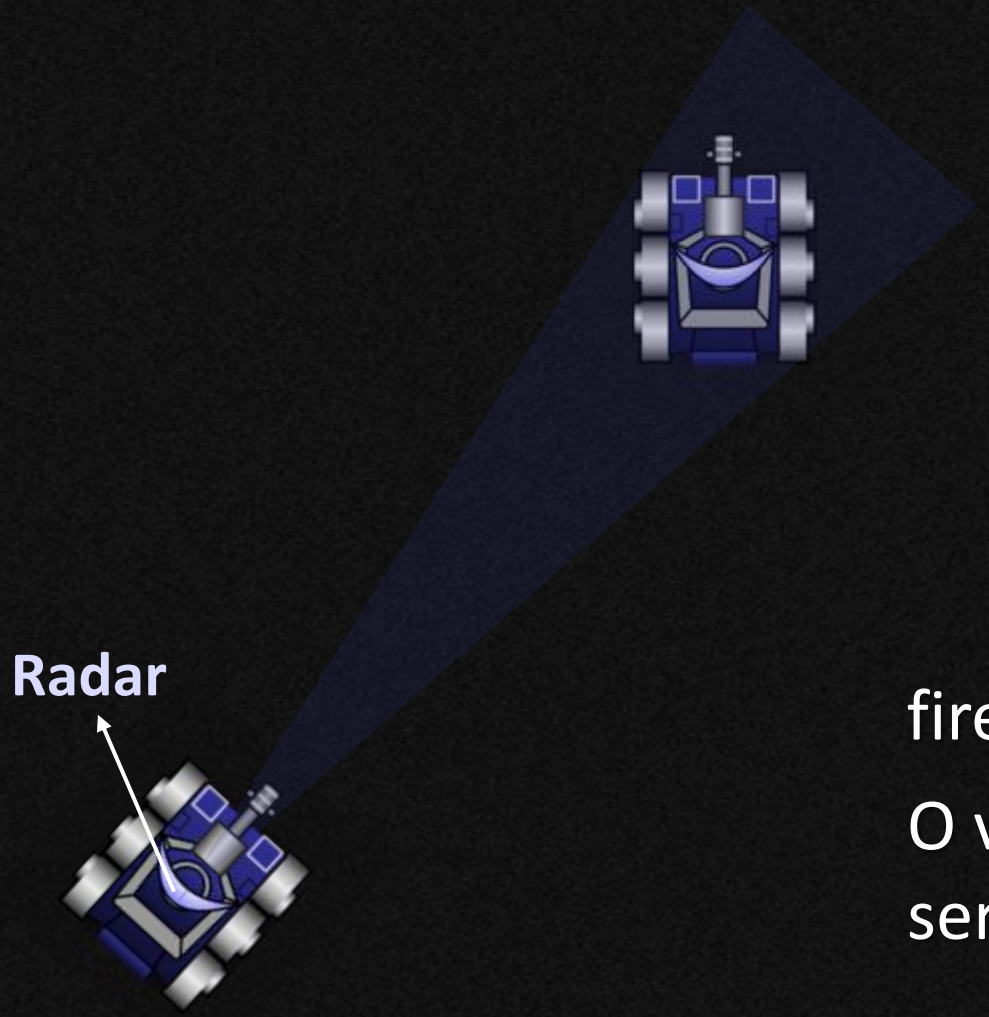
Professor
José de Assis





Evento

onScannedRobot(ScannedRobotEvent e)



fire(1)

O valor do tiro pode ser de **0.1** até **3**

Playlist deste curso:

www.youtube.com/ProfessorJosedeAssis