



Lógica de Programação

com Java 

START

MENU



Eduarda Farias

Estudante de Ciência da Computação – UFCG



Linkedin



Github



Instagram





Rayane Bezerra

Estudante de Ciência da Computação – UFCG



Linkedin



Github

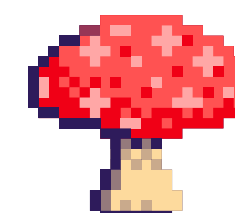


Instagram





Programação



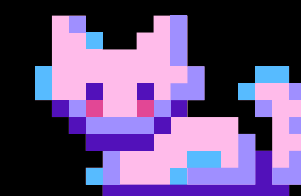
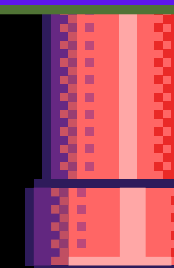
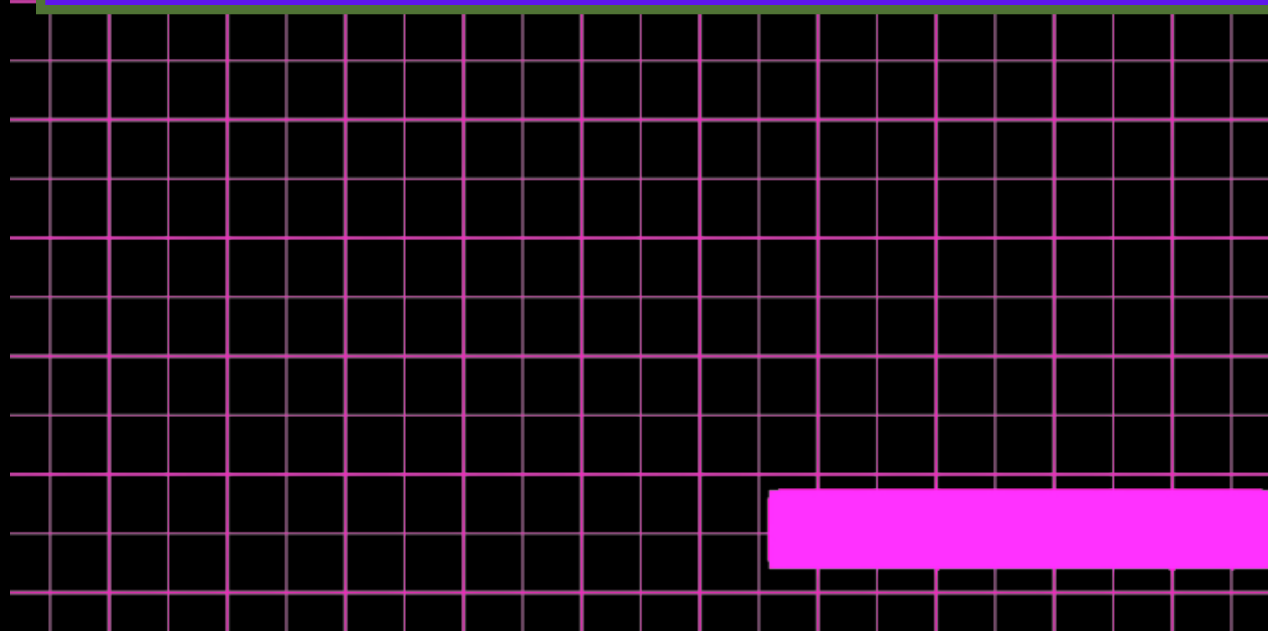
SIM

NÃO





**Vamos passar por
alguns temas...**





Temas



Variáveis



and --- or



Operadores Aritméticos



Condicionais



Laços de Repetição



listas



Classes

Objetos



O que são algoritmos?

Algoritmos são sequências de passos:

Ex:



Receita de pizza



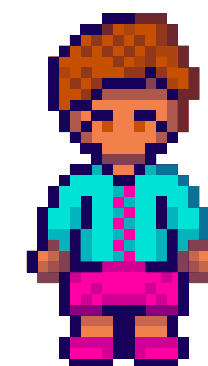
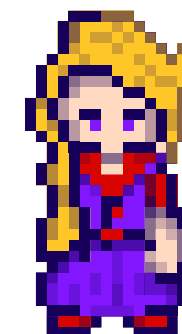
Escutar música

No caso da programação, os algoritmos são sequências de passos usados para resolver um problema, mas, como o computador não entende nossa linguagem usamos as linguagens de programação



E as linguagens de Programação?

Usamos as linguagens de programação,
para nos comunicarmos com o
computador, dizer o que ele deve fazer,
as linguagens de programação
funcionam tradutores



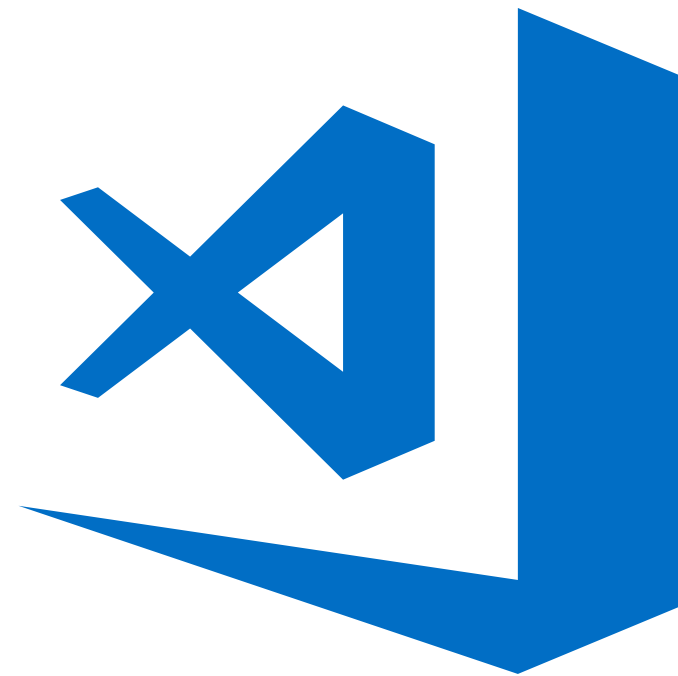
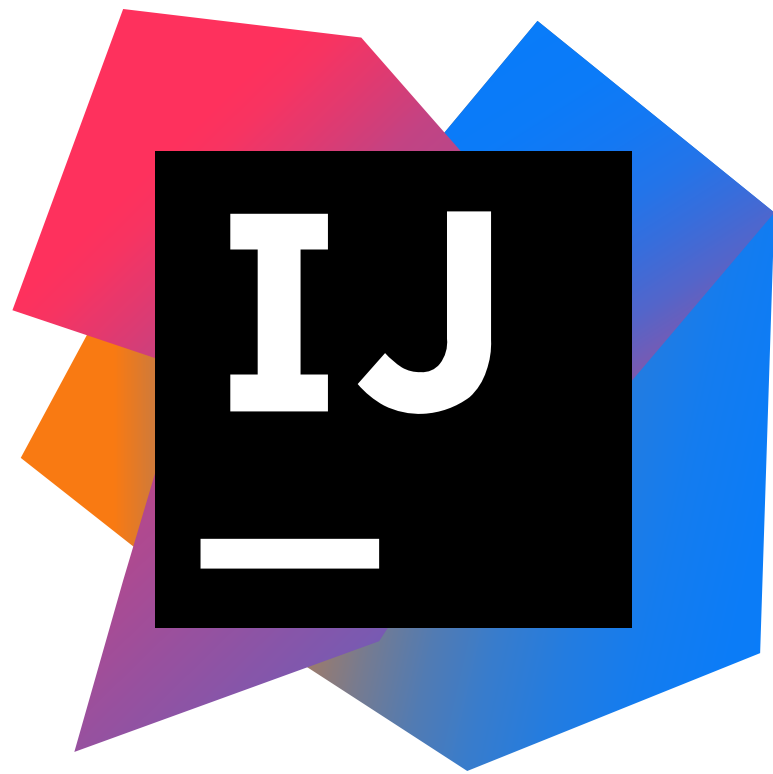


Pra que serve a lógica de Programação?

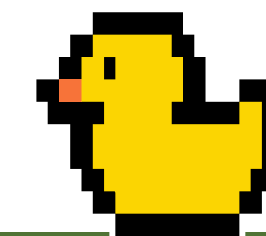
A lógica de programação é um elemento essencial para desenvolver algoritmos de programação, pois é ela quem organiza de maneira coerente, a sequência de ações que um algoritmo precisa executar.

Aprendendo a LÓGICA de programação, podemos aplicar esses conceitos a diferentes linguagens.

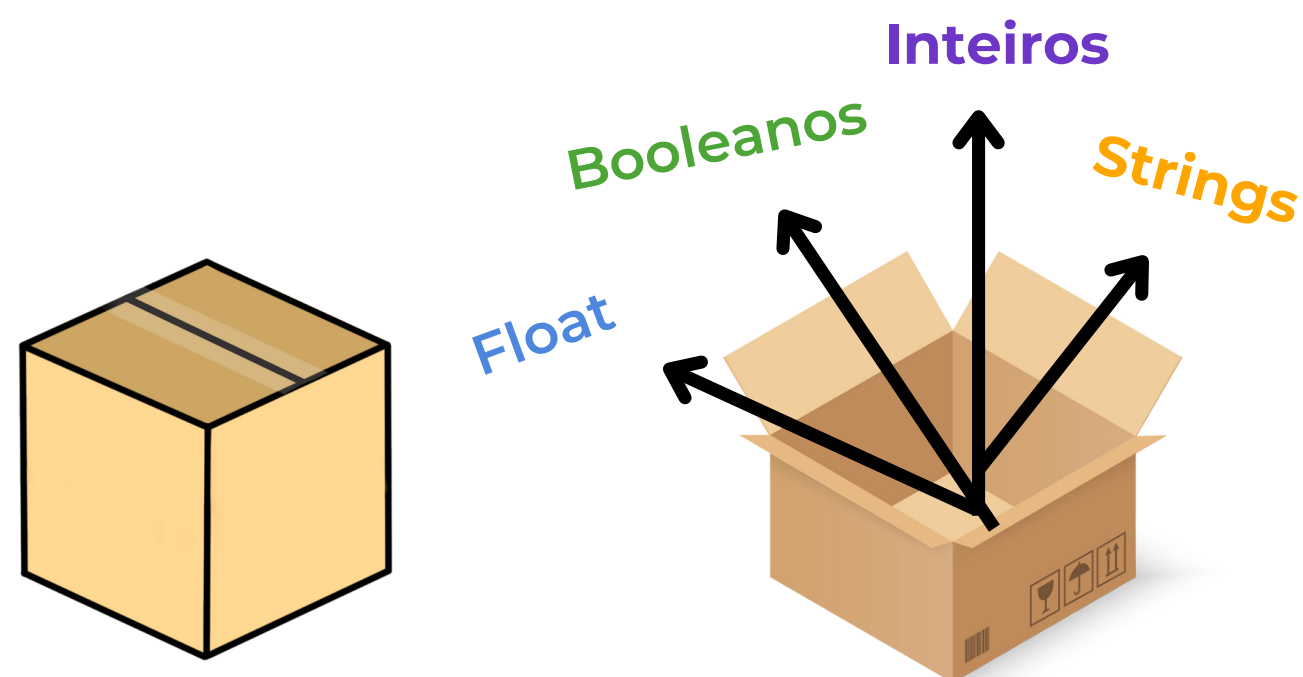
Editores de Código



O que são variáveis ?



Variáveis funcionam como caixas, ou seja, elas guardam alguma coisa dentro delas, nós vamos chamar essas "coisas" de valores e esses valores que são guardados nas variáveis podem ser dos tipos booleanos, strings, inteiros e floats.



Tipos de variáveis em Java:

Booleanos: True, False

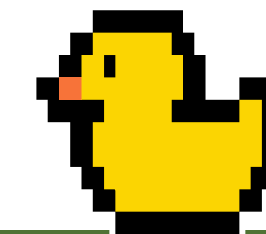
Strings: palavras ou frases

Inteiros: 2, 3, 1000, 478, 780

Floats: 2.35, 000.9, 1.85



Variável do tipo String



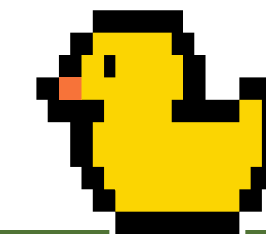
Variáveis do tipo String são as palavras ou caracteres ,
as strings são representadas dentro de aspas (que em
python podem ser simples ou duplas)

```
© VariavelString.java x
1 public class VariavelString {
2     public static void main(String[] args) {
3         String fruta1 = "abacate";
4         String fruta2 = "uva";
5     }
6 }
```

Strings podem ser frases, um
texto ou até mesmo uma palavra,
desde que esteja dentro das
aspas



Variável do tipo Booleano



Variáveis do tipo booleanas, são estados que uma variável podem ter, esses estados são chamados de True ou False, respectivamente Verdadeiro ou falso. E como usamos isso? Quando queremos verificar se algo é verdade ou não, por exemplo :

```
public class VariavelBooleana {  
    Run | Debug  
    public static void main(String[] args) {  
        Boolean rosa = true;  
        Boolean verde = false;  
    }  
}
```



Variável do tipo Inteiros

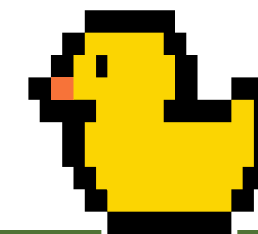


Variáveis do tipo int, são os números inteiros

```
public class VariavelInt {  
    Run | Debug  
    public static void main(String[] args) {  
        int idade = 20;  
    }  
}
```



Variável do tipo Float e Double



Variáveis do tipo float, representam os números decimais com uma ou mais casas decimais

```
public class VariavelFloat {  
    Run | Debug  
    public static void main(String[] args) {  
        double preco = 4.60;  
  
        float pi = 3.14159265f;  
    }  
}
```



Como declarar variáveis em Python?

Alguns pontos importantes:

- Em Java a primeira letra do nome da variável é minúscula
ex: `int numero = 3`
- O formato para declarar uma variável é declarar(escrever) o tipo, nome, e atribuir um valor para ela, e como fazemos isso? Utilizando o sinal de atribuição `" = "`.
- Assim, no exemplo, eu estou declarando uma variável chamada `numero1`, que guarda dentro dela o valor `2`

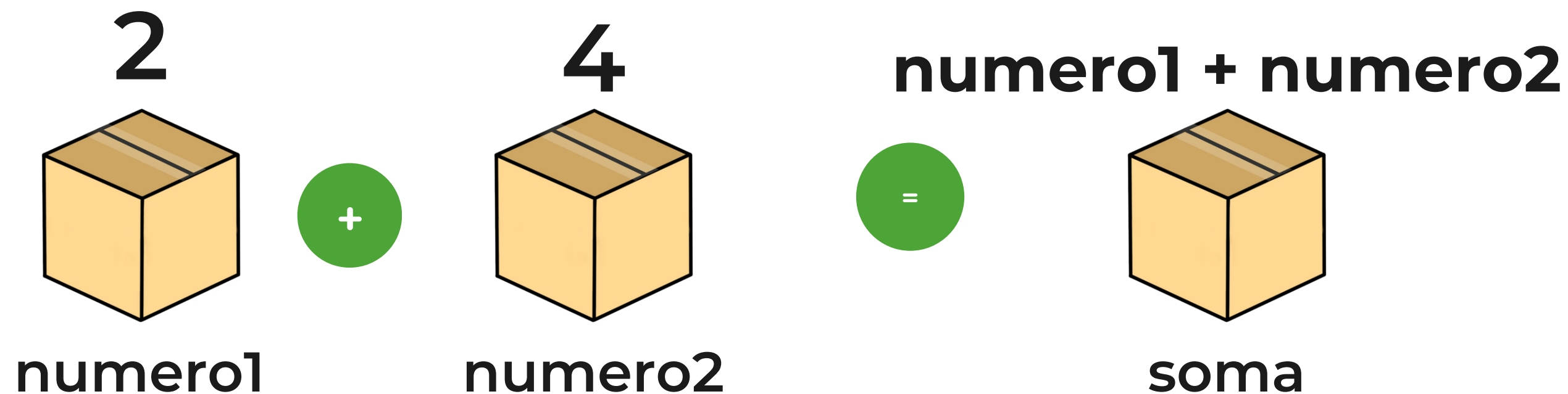
```
public class DeclarandoVariavel {  
    Run | Debug  
    public static void main(String[] args) {  
        int numero1 = 2;  
        int numero2 = 4;  
  
        int soma = numero1 + numero2;  
    }  
}
```



Analizando o exemplo do slide anterior

(Usando o exemplo da caixa)

```
public class DeclarandoVariavel {  
    Run | Debug  
    public static void main(String[] args) {  
        int numero1 = 2;  
        int numero2 = 4;  
  
        int soma = numero1 + numero2;  
    }  
}
```

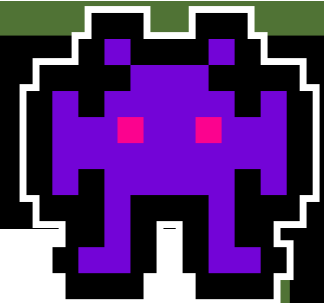


Declarando Variáveis

```
public class Variaveis {  
    Run | Debug  
    public static void main(String[] args) {  
        // String  
        String nome = "maria";  
  
        // String  
        String nomeFaculdade = "UFGC" ;  
  
        // int  
        int idade = 5;  
  
        // preco  
        double preco = 3.00;  
  
        // float  
        float metros = 4.139f;  
  
        // Booleans  
        boolean aprovado = true;  
  
        boolean reprovado = false;  
    }  
}
```

Variáveis devem ter seu nome começado com letras minúsculas, e caso você precise de um nome maior para sua variável utilize Camel Case já que em java não usamos espaços para separar. É importante lembrar de colocar nomes que realmente sejam referentes ao que você vai manipular no código.

Exercício



Imagine um programa que faça o cadastro de uma pessoa que tipos de variáveis você criaria, e quais nomes usariam?



Entradas e Saídas



Entradas e Saídas



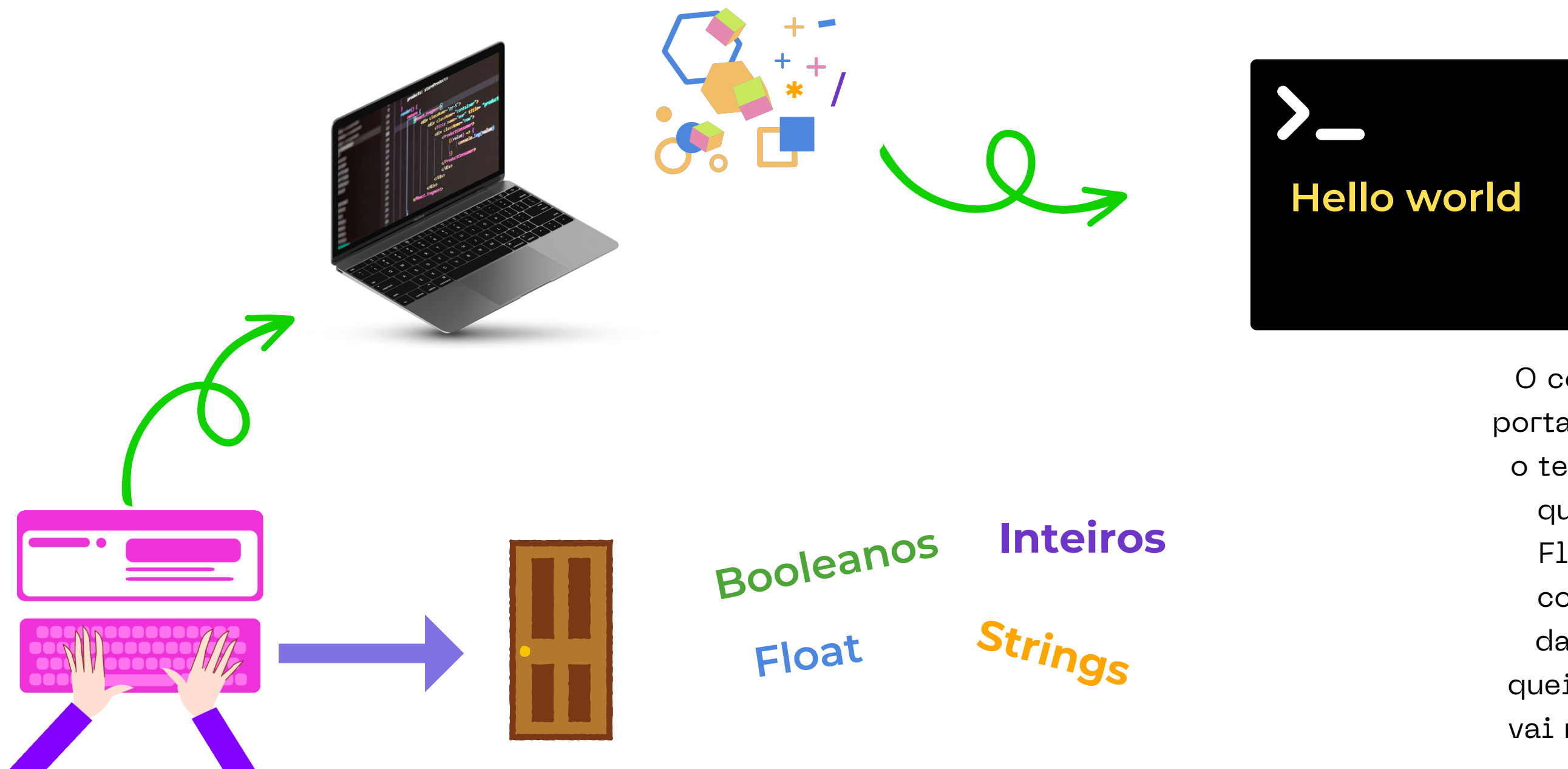
Imagine que o personagem do jogo está em busca de uma poção mágica que lhe dê o poder de se multiplicar, então ele faz os seguintes movimentos:

1. Entra no castelo
2. Bebe a poção
3. É multiplicado
4. Sai do castelo

Entradas e Saídas



Entradas e Saídas



O computador recebe algo por uma porta de entrada, que nesse caso seria o teclado, essas entradas tem tipos, que podem ser Strings, Inteiros, Floats e Booleanos, em seguida o computador vai manipular esses dados de acordo com o que você queira no algoritmo, e após isso ele vai mostrar na tela o que você disse para ele fazer.

Entrada de Dados

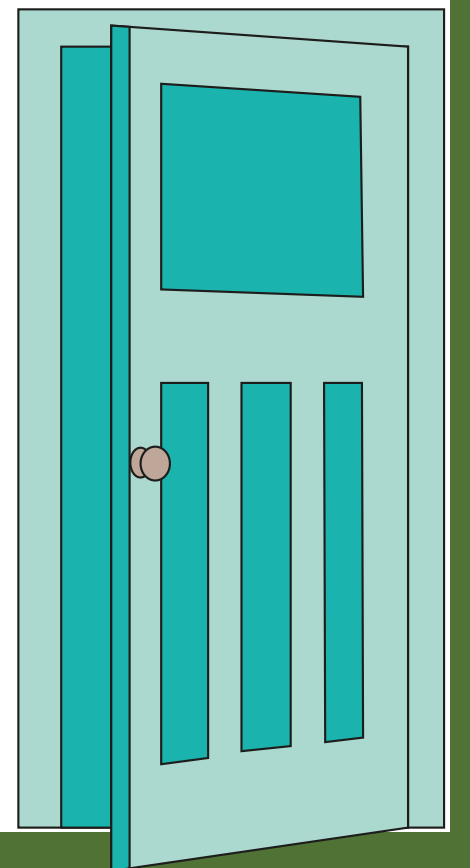
Mas como pegamos esses dados de entrada?

Utilizando a função :

Scanner()

Podemos usar essa função para pegar entradas digitadas.

Mas, atenção, você precisa dizer qual tipo de dado está sendo recebido no input, pois se você tentar passar um dado de um tipo quando na verdade ele espera outro, você terá um erro.



Entrada de Dados



```
import java.util.Scanner;

public class Entradas {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

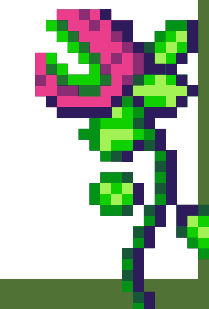
        // Fazendo input de uma String
        String nome = sc.nextLine();

        // Fazendo o input de um float
        float altura = sc.nextFloat();

        // Fazendo input de um inteiro
        int numeroCasa = sc.nextInt();

        // Fazendo o input de um boolean
        boolean aprovado = sc.nextBoolean();
    }
}
```

Entrada de Dados Especiais



Existem algumas entradas de dados que são especiais, mas, como assim? Simples, é como se elas conversassem com você. Você deve ter percebido que ele deixou apenas que você digitasse seus valores de entrada, mas agora, com esse tipo de entrada vamos conseguir “passar uma mensagem no input”, que será exibida na tela ao pedir um dado.

Entrada de Dados Especiais

```
import java.util.Scanner;

public class EntradasEspeciais {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nomeMessage = "Digite seu primeiro nome: ";
        String sobrenomeMessage = "\nDigite seu segundo nome:";
        String idadeMessage = "\nDigite sua idade:";

        System.out.println(nomeMessage);
        String nome = sc.nextLine();

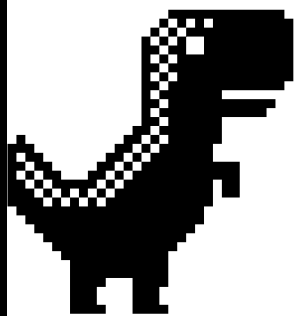
        System.out.println(sobrenomeMessage);
        String sobrenome = sc.nextLine();

        System.out.println(idadeMessage);
        int idade = sc.nextInt();
    }
}
```



Saídas

Ok, já vimos que podemos fazer o computador receber dados através da função `Scanner()`, mas, e depois? Eu posso mostrar esses dados? Posso fazer alguma manipulação com eles e printar no terminal? SIM! Você pode fazer isso utilizando a função `System.out.println()` e dentro dos parênteses você digita o que quer exibir na tela



Saídas

```
public class Saida {  
    Run | Debug  
    public static void main(String[] args) {  
        int soma = 2 + 2;  
  
        System.out.println(x:"Hello World!");  
  
        System.out.println(x:2);  
  
        System.out.println(soma);  
  
        System.out.println(2*5);  
    }  
}
```

A função `println()`, mostra na tela o que você pede que ela mostre, e pode ser uma variável, uma string, o resultado da operação de uma variável. Além disso, função `print` permite também que eu coloque uma operação diretamente nela



Saídas Especiais

Assim como as entradas, as saídas também possuem saídas especiais, essas saídas são as chamadas **saídas formatadas**

Como vocês viram, antes não podemos misturar dados de tipos diferentes, mas suponha que eu tenha o seguinte código:

```
double precoAbacate = 5.20;  
double precoMamao = 4.50;  
double valorTotal = precoAbacate + precoMamao;
```

Como faço para fazer um print misturando uma string e dois floats para printar na saída a seguinte frase "O valor total da compra é 9,70" ? Para isso usamos a formatação de Saída, concatenando através do "+" o que queremos printar.

Saídas Especiais

```
public class SaidasFormatadas {  
    Run | Debug  
    public static void main(String[] args) {  
        double precoAbacate = 5.20;  
        double precoMamão = 4.50;  
        double valorTotal = precoAbacate + precoMamão;  
  
        System.out.println("O preço do abacate é " + precoAbacate + " e o preço do mamão é "  
            + precoMamão + " o valor total da sua compra é: " + valorTotal );  
    }  
}
```

Saídas Especiais

Ainda tenho outra forma de usar format para formatar minhas saídas.
E com essa formatação, especificar a quantidade de casas decimais que eu quero

```
System.out.printf("O preço do abacate é %.2f", precoAbacate)
```

Só que agora eu uso o %.2f no lugar da variável que eu vou formatar e depois digo qual variável eu estou formatando.

Saídas Especiais

```
public class SaidasFormatadas {  
    Run | Debug  
    public static void main(String[] args) {  
        double precoAbacate = 5.20;  
        double precoMamao = 4.50;  
        double valorTotal = precoAbacate + precoMamao;  
  
        System.out.printf("O preço do abacate é: %.2f\n"  
            + "O preço do mamão é: %.2f\n"  
            + "O valor total da sua compra é: %.2f",  
            precoAbacate, precoMamao, valorTotal  
        );  
    }  
}
```

Operadores Aritméticos

Operadores	Função	Exemplo
+	Adição	4 + 12
-	Subtração	3 - 2
*	Multiplicação	5 * 15
/	Divisão	10 / 2
%	Resto da divisão (também chamado de módulo)	10 % 2

Os operadores aritméticos são
extremamente parecidos com os que
usamos diariamente

Usando Operadores Aritméticos

```
public class Operadores {  
    Run | Debug  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.println(x:"Por favor insira o primeiro numero: ");  
        int num1 = sc.nextInt();  
        int num2 = sc.nextInt();  
  
        int total = num1 + num2;  
  
        System.out.println(total);  
        System.out.println(x:"-----");  
        System.out.println(num1 + num2);  
    }  
}
```

Operadores de Comparação

Operadores	Função	Exemplo
=	Atribuição	fruta = "Manga"
==	Comparação de igualdade	3 == 2
>	Maior que	10 > 7
<	Menor que	7 < 10
>=	Maior ou igual	15 >= 15
<=	Menor ou igual	10 <= 15
!=	Diferente de	3 != 8

Os operadores de comparação também são parecidos com os que usamos diariamente

Atenção

Atribuição

idade = 20

idade2 = 15

Igualdade

idade1 == idade2

No começo é comum confundir os operadores de atribuição e igualdade, lembre-se que atribuição é =, e eu uso quando quero atribuir(guardar) um valor dentro da minha variável. Já == é quando eu quero verificar se duas coisas são iguais ou não



Comparação

```
public class Comparacao {  
Run | Debug  
    public static void main(String[] args) {  
        int idade1 = 20;  
        int idade2 = 30;  
  
        System.out.println(idade1 == idade2);  
    }  
}
```



Condicionais

A estrutura condicional é uma seção que ajuda a definir condições para a execução de determinados blocos de comandos. o programa deve parar para executar um teste e decidir qual caminho seguir.

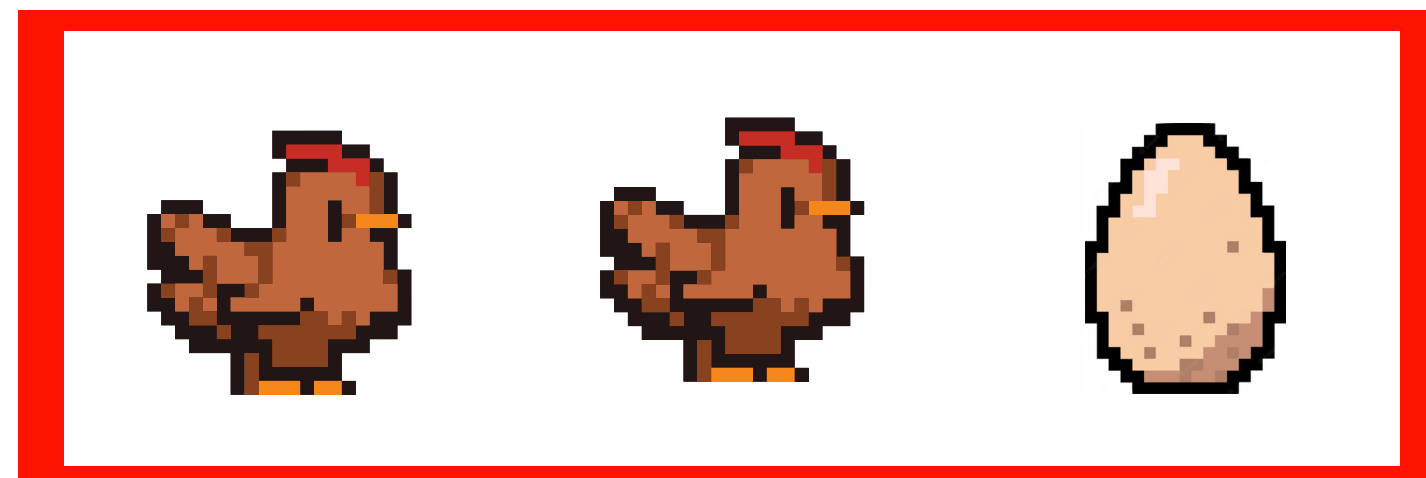
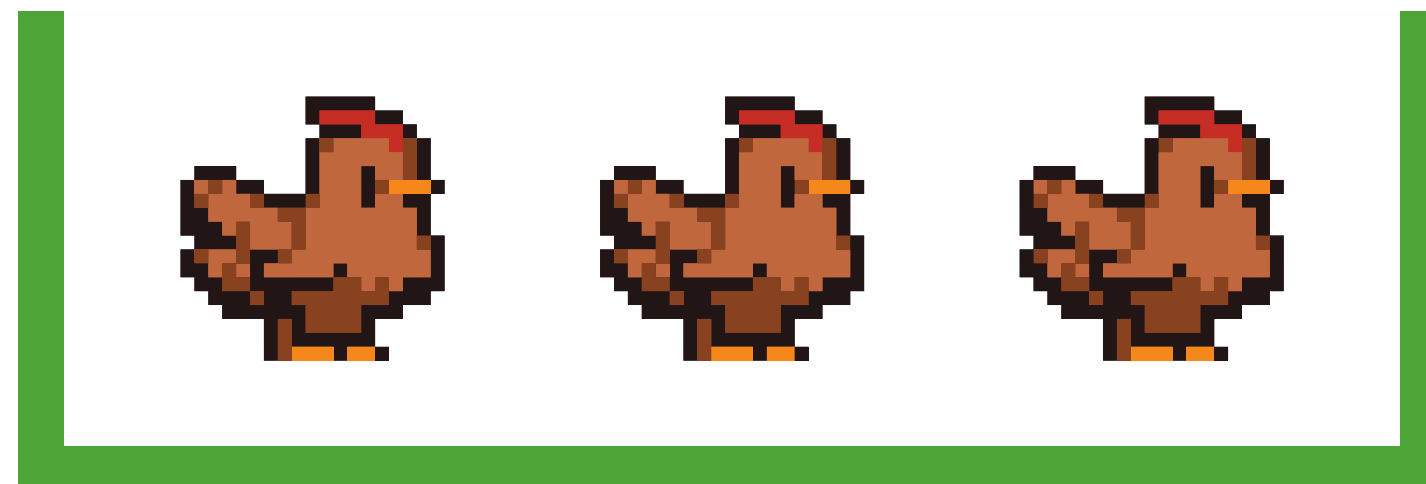
Funciona assim: Se essa condição X é satisfeita, então execute esse bloco de comandos; se não, execute esse outro bloco de comandos.

and

Utilizando And e Or você consegue verificar condições, vamos ver como elas funcionam:

and: verifica se todas as condições são verdadeiras, somente se todas forem verdadeiras é que retorna true e no caso dos códigos e condicionais executa o bloco de código

&&

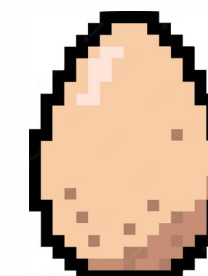


or

Utilizando And e Or você consegue verificar condições, vamos ver como elas funcionam:

or: verifica se existe pelo menos uma das condições que sejam verdadeiras, se ao menos uma for verdadeira é retornado true e no caso dos códigos e condicionais executa o bloco de código

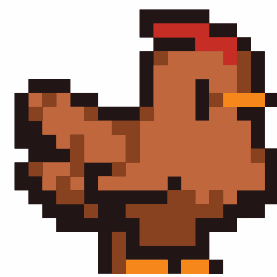
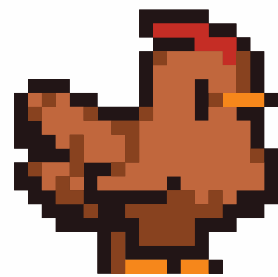
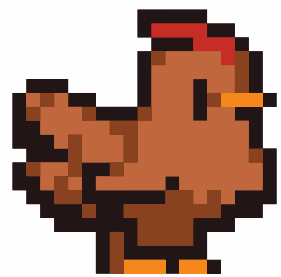
||



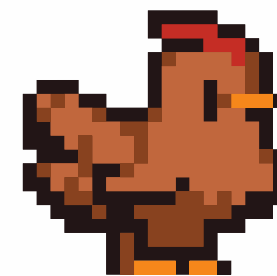
AND

vs

OR



&&

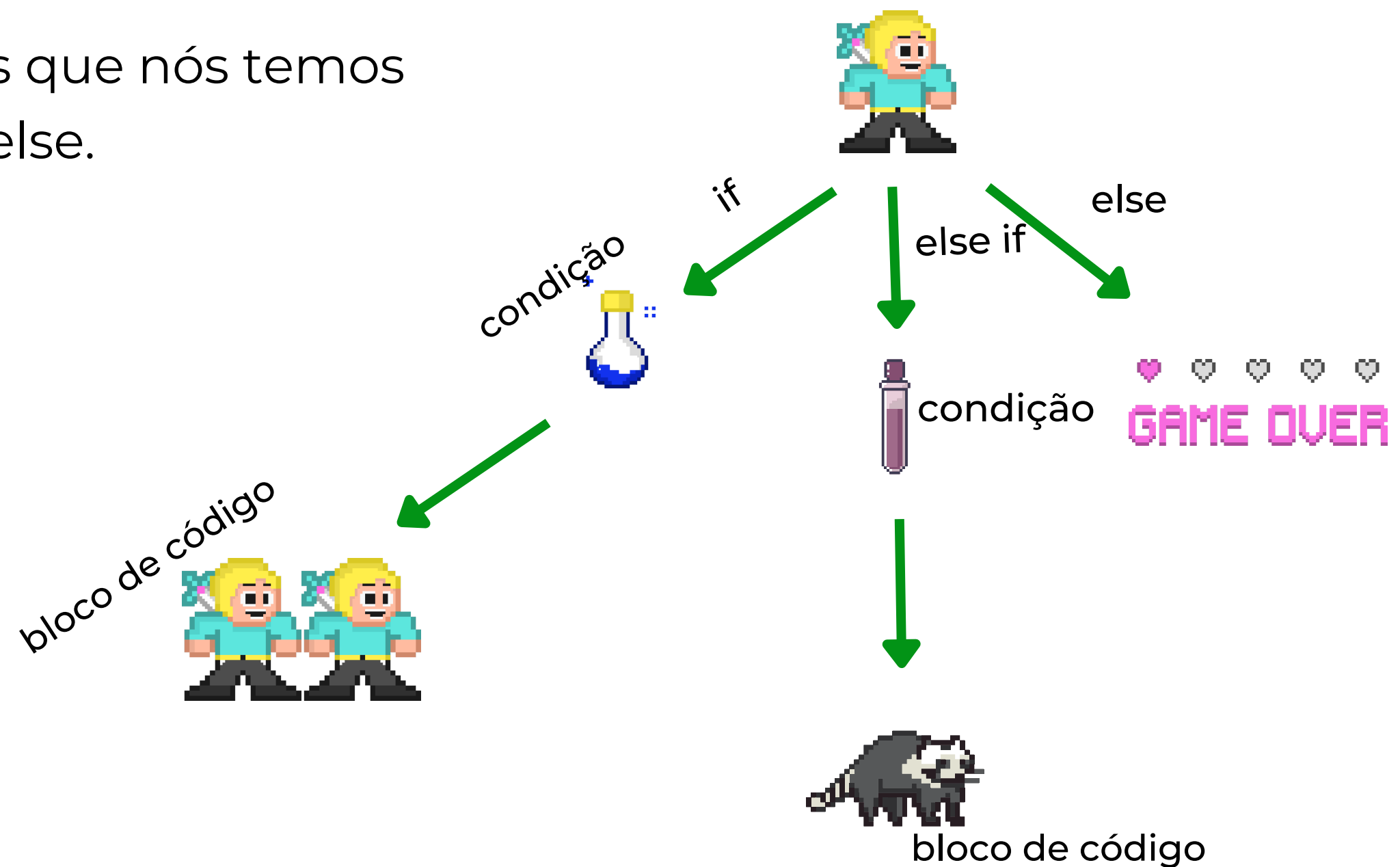


||

Condicionais

As estruturas condicionais que nós temos são o if, elif e else.

As condicionais usadas são if, else if e else, na mesma estrutura do slide anterior. O else é diferente, pois ele não precisa de uma condição, pois já é implícito que sua condição é "nenhuma das condições anteriores foi satisfeita", se nenhuma foi satisfeita só sobrou essa, logo só pode ser essa.



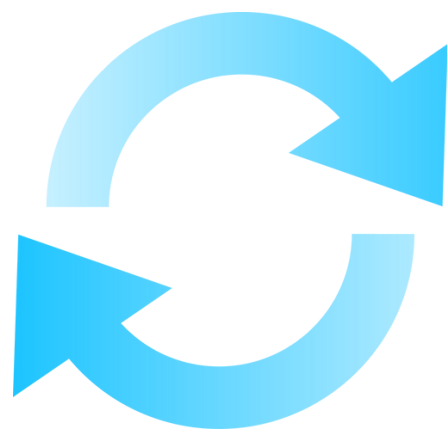
Condicionais

```
public class Condicionais {  
    Run | Debug  
    public static void main(String[] args) {  
        int numero = 5;  
        String message = "";  
        if (numero > 0) {  
            message = "O número é positivo";  
        } else if (numero < 0) {  
            message = "O número é negativo.";  
        } else {  
            message = "O número é zero.";  
        }  
  
        System.out.println(message);  
    }  
}
```



Uma nova etapa

Laços de Repetição



Imagine se você tivesse que repetir um trecho do seu código de programação todas as vezes que você quisesse realizar uma ação repetitiva... Seu código seria enorme! É por isso que dentro da lógica de programação você pode contar com loops também chamados de laços de repetição. Eles permitem que você realize uma mesma ação repetidamente até que uma condição, dada por você, seja atendida.

Contando

Estamos acostumados a contar a partir de 1, mas na programação geralmente começamos a contar do 0

0	1	2	3	4	5
---	---	---	---	---	---

Laços de Repetição

Laços de repetição ou loops são usados para não ficarmos repetindo um mesmo trecho de código, vamos ver os exemplos no próximo slide, onde o objetivo é fazer um código que print os números de 1 até 10.



Laços de Repetição

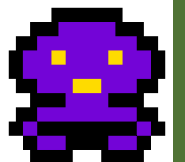
```
public class VantagensFor1 {  
    Run | Debug  
    public static void main(String[] args) {  
        System.out.println(x:0);  
        System.out.println(x:1);  
        System.out.println(x:2);  
        System.out.println(x:3);  
        System.out.println(x:4);  
        System.out.println(x:5);  
        System.out.println(x:6);  
        System.out.println(x:7);  
        System.out.println(x:8);  
        System.out.println(x:9);  
        System.out.println(x:10);  
    }  
}
```

```
public class VantagensFor2 {  
    Run | Debug  
    public static void main(String[] args) {  
        for (int i = 0; i <= 10; i++) {  
            System.out.println(i);  
        }  
    }  
}
```


Laços de Repetição

Agora que você já viu a importância dos loops vamos ver o que podemos fazer com eles e como podemos fazer !

Podemos usar os loops para repetir um mesmo trecho de código, e também para iterar(passear) por uma String, por uma lista e até mesmo por dicionários suas chaves e valores



For

O uso mais básico do for é para printarmos um intervalo no for.

Começo

Fim

```
public class VantagensFor {  
    Run | Debug  
    public static void main(String[] args) {  
        for (int i = 0; i <= 10; i++) {  
            System.out.println(i);  
        }  
    }  
}
```


Variável de controle, é atualizada a cada rodada do for

Laço de repetição: For

```
public class Listas {  
    Run | Debug  
    public static void main(String[] args) {  
        String[] cursos = new String[4];  
        String[] nomes = {"Eduarda, Ana, Lara"};  
  
        for (int i = 0; i < nomes.length; i++) {  
            System.out.println(nomes[i]);  
        }  
    }  
}
```

Eduarda, Ana, Lara

Laço de repetição: For

 for_range.py X

 Logica-de-Programacao-python >  for_range.py > ...

```
1   engenharias = ["Civil", "Elétrica", "Mecânica"]
2
3   for item in range(len(engenharias)):
4       print(engenharias[item])
```

Aqui estamos pegando o elemento que está dentro de cada item da lista. Você pode ler esse código como:

Para cada item na lista das engenharias printe o elemento da lista de engenharias que está no índice item. Confuso? Rode o código para entender melhor.



Laço de repetição: While

O while é uma estrutura de repetição que executada enquanto uma condição for verdadeira ou não. Como assim? Existem alguns tipos de while, mas vamos nos concentrar em dois tipos, `while(True)` e `while(Condição)`, vamos entender a diferença entre eles. Mas antes, vamos observar que se tanto while quanto o for servem para executar um bloco de código repetidas vezes, então, como saber qual usar?

FOR vs While

Geralmente utilizamos o `for()` quando sabemos quantas vezes iremos executar aquele mesmo bloco de código, já o `while` é usado quando não sabemos quantas vezes vamos precisar repetir, justamente por isso que usamos essa estrutura de repetição até que uma condição seja alcançada.

Formato geral de um while

```
while condição:  
    bloco_de_código
```

Enquanto a condição for verdadeira execute esse bloco de código

While condição

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

A condição é dita no começo do while

While condição

1º Rodada:

contador = 0

print(0)

contador += 1

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

While condição

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

1º Rodada:

contador = 0
print(0)
contador += 1

2º Rodada:

contador = 1
print(1)
contador += 1

While condição

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

1º Rodada:

contador = 0
print(0)
contador += 1

2º Rodada:

contador = 1
print(1)
contador += 1

3º Rodada:

contador = 2
print(2)
contador += 1

While condição

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

1º Rodada:

contador = 0
print(0)
contador += 1

2º Rodada:

contador = 1
print(1)
contador += 1

3º Rodada:

contador = 2
print(2)
contador += 1

4º Rodada:

contador = 3
print(3)
contador += 1

While condição

```
public class While1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador < 5){  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

O 5 não pode ser exibido,
pois a condição é que o
contador seja menor que 5
e 5 não é menor que 5,
portanto o loop para.

1º Rodada:
contador = 0
print(0)
contador += 1

2º Rodada:
contador = 1
print(1)
contador += 1

3º Rodada:
contador = 2
print(2)
contador += 1

4º Rodada:
contador = 3
print(3)
contador += 1

5º Rodada:
contador = 4
print(4)
contador += 1

While True

```
while1.py U X
Logica-de-Programacao-python > while
1   contador = 0
2
3   while contador < 5:
4       print(contador)
5       contador += 1
6
```

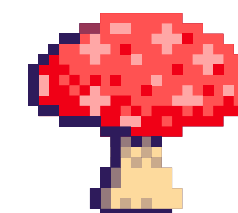
A condição é dita no
começo do while

```
public class While2 {
    Run | Debug
    public static void main(String[] args) {
        int contador = 0;
        while (true){
            if (contador < 5){
                System.out.println(contador);
                contador++;
            }
        }
    }
}
```

A condição é dita dentro
do while

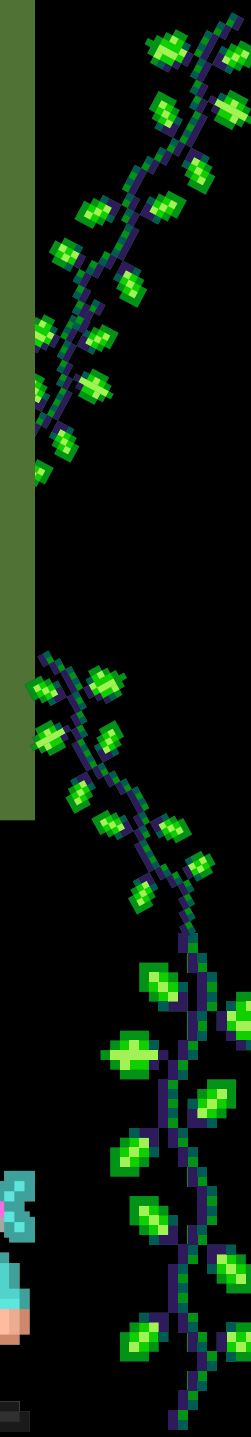
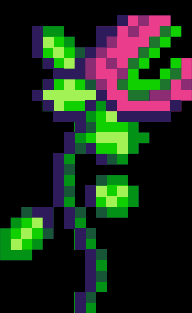


Programação Orientada a objetos (POO)



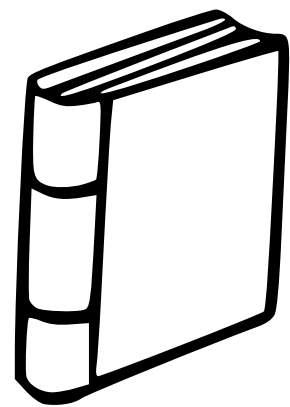
SIM

NÃO



Classes e Objetos

A classe funciona como um molde para um objeto

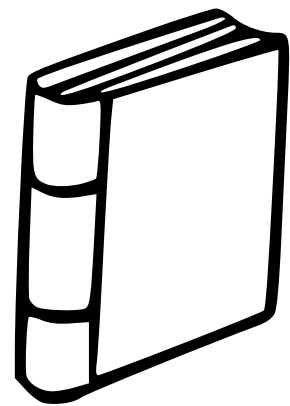


Livro

Classes e Objetos

A classe funciona como um molde para um objeto

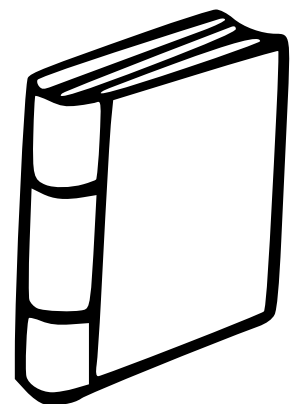
- Nome



Livro

Classes e Objetos

A classe funciona como um molde para um objeto

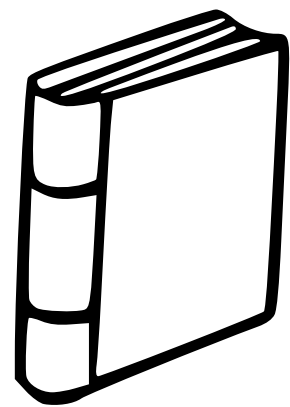


Livro

- Nome
- Gênero

Classes e Objetos

A classe funciona como um molde para um objeto

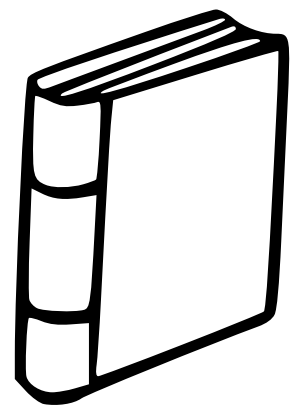


Livro

- Nome
- Gênero
- Quantidade de Páginas

Classes e Objetos

A classe funciona como um molde para um objeto

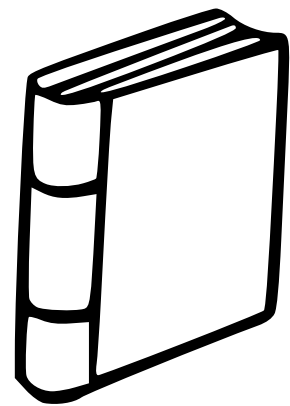


Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor

Classes e Objetos

A classe funciona como um molde para um objeto

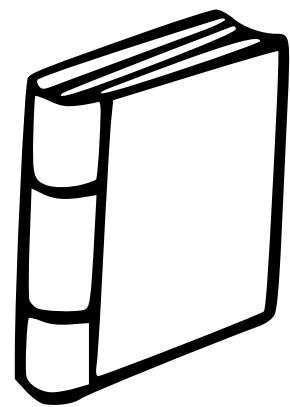


Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação

Classes e Objetos

A classe funciona como um molde para um objeto

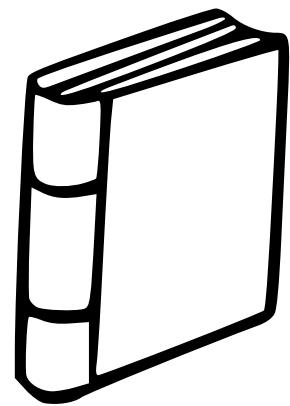


Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação
- Editora

Classes e Objetos

A classe funciona como um molde para um objeto

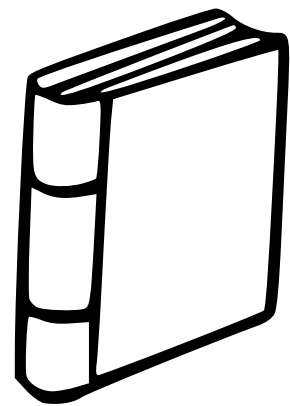


Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação
- Editora
- Preço

Classes e Objetos

A classe funciona como um molde para um objeto

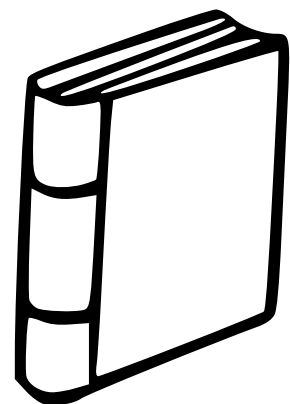


Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação
- Editora
- Preço
- Faixa etária

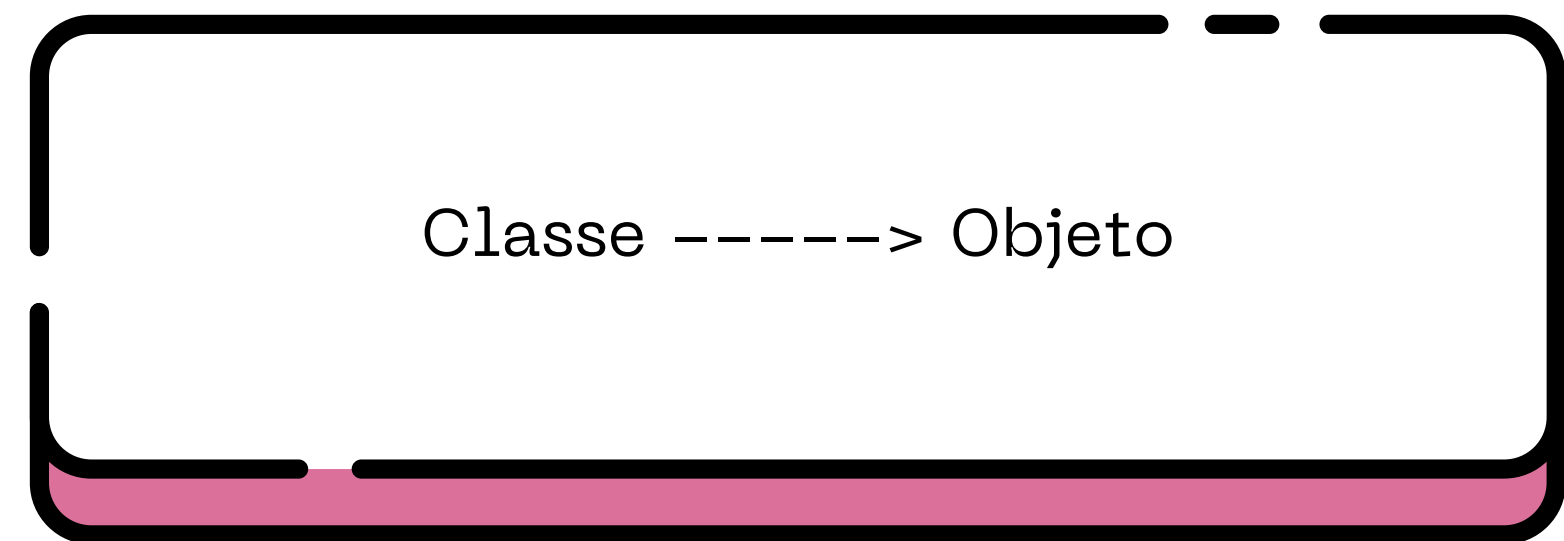
Classes e Objetos

A classe funciona como um molde para um objeto



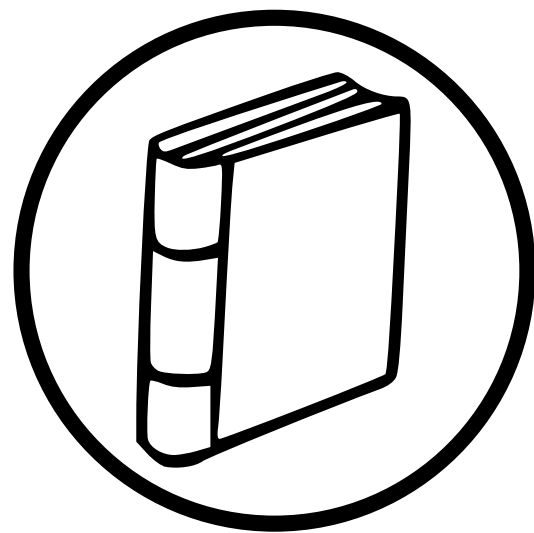
Livro

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação
- Editora
- Preço
- Faixa etária



Classes e Objetos

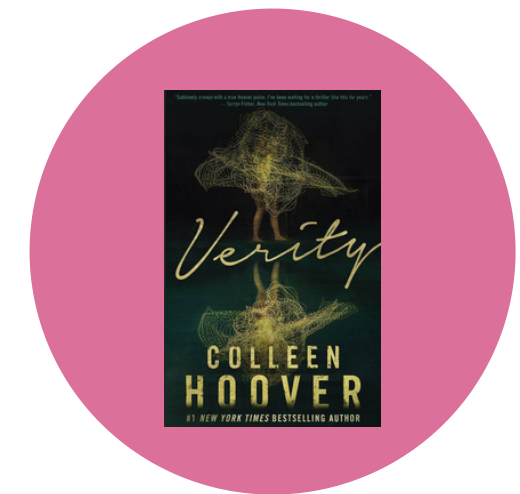
A classe funciona como um molde para um objeto



Livro

Classe

- Nome
- Gênero
- Quantidade de Páginas
- Autor
- Ano de publicação
- Editora
- Preço
- Faixa etária



Criando uma classe

Ao criar uma classe precisamos responder três perguntas:

- O que essa classe tem?
- O que ela faz?
- Como estou agora?

Criando uma classe

Ao criar uma classe precisamos responder três perguntas:

- O que essa classe tem? Atributos(características)
- O que ela faz?
- Como estou agora?

Criando uma classe

Ao criar uma classe precisamos responder três perguntas:

- O que essa classe tem? Atributos(características)
- O que ela faz? Métodos (Comportamentos)
- Como estou agora?

Modelando uma classe Livro

Atributos

- Nome
- Gênero
- Quantidade de páginas
- Autor
- Ano de Publicação
- Editora
- Preço
- Faixa Etária

Métodos

- Apresentar livro
- Categorizar Livro
- desconto

Modelando uma classe Banco

Atributos

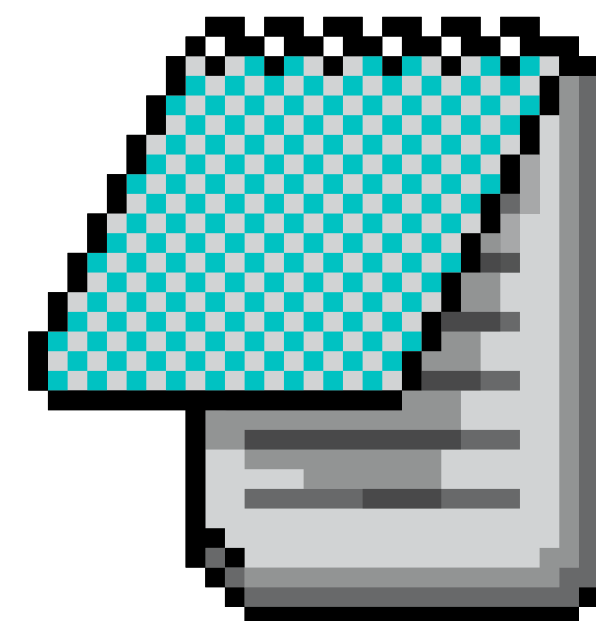
- Número da conta
- Tipo da conta
- Dono da conta
- Saldo
- Status

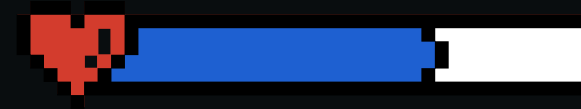
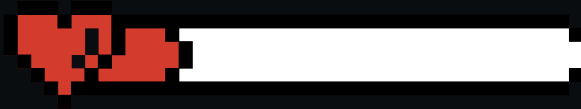
Métodos

- Estado atual da conta
- Abrir uma conta
- Fechar uma Conta
- Depositar
- Sacar



Desafios





Obrigada
pela atenção
até aqui!

