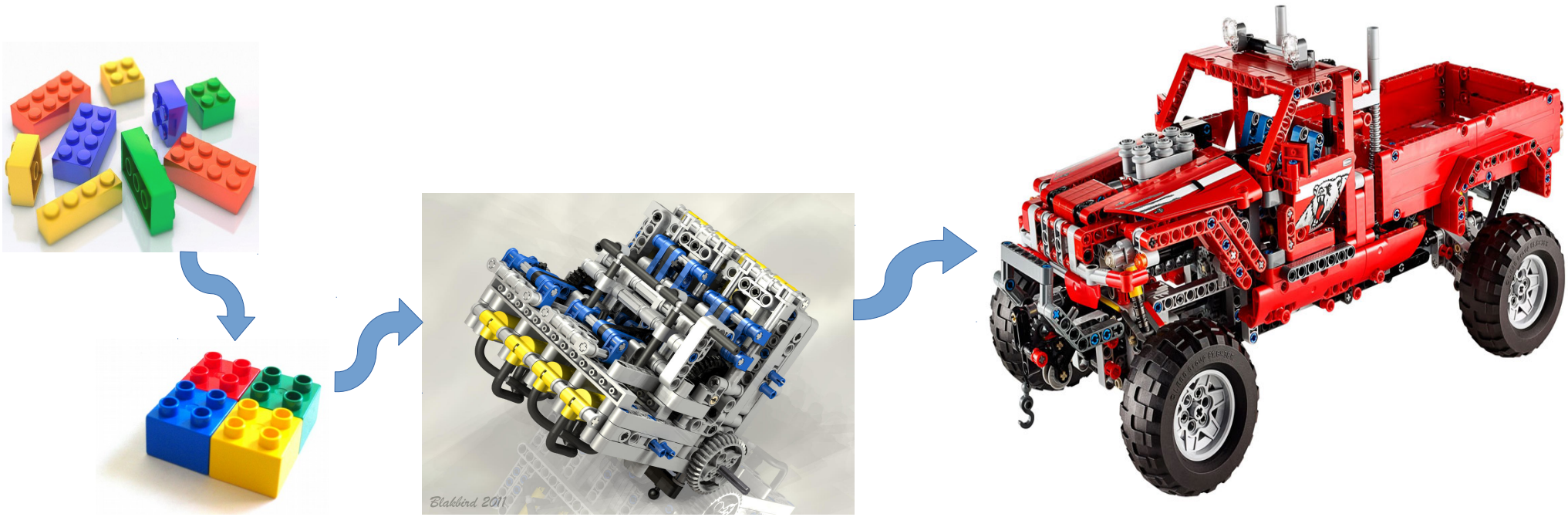
Orientação a Objetos – Mas tudo é objeto?

Prof. Tiago Sombra

Projeto Orientado a Objetos

- Tudo é Objeto!!!
  - Onde quer que você olhe no mundo real você vê objetos:
    - Pessoas, carros, computadores...



Projeto Orientado a Objetos

- O que é POO?
  - É uma maneira de pensar e programar que ajuda na organização do software

Projeto Orientado a Objetos - Abstração

- Representação dos objetos – como os objetos interagem e seus papéis dentro do sistema de software
  - **Identidade –** deve ser única
    - Uma pessoa é diferente de um carro
  - **Propriedades –** características do objeto, que o definem
    - Uma pessoa tem altura, peso, idade
  - **Métodos –** quais são as ações que o objeto irá desempenhar
    - Uma pessoa anda, senta …

- Na POO os objetos são representados por **classes**
  - Contém todas informações sobre o objeto

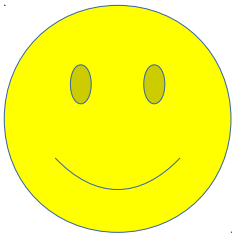
Projeto Orientado a Objetos – Encapsulamento

- Principal técnica que define a POO:
  - Adiciona segurança à aplicação
  - Esconde as propriedades do objeto - “caixa preta”

- Na maioria das linguagens OO é implementado através de métodos e atributos privados
  - Visíveis somente dentro do próprio objeto
  - Evita o acesso direto as propriedades do objeto

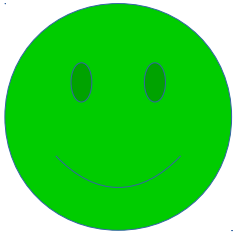
Projeto Orientado a Objetos – Herança

- Objetos herdam propriedades e métodos:
  - Um pessoa herda características de seus pais (ancestrais)



- Nome
- Altura
- telefone

- Avô



- Nome
- Altura
- Telefone
- e-mail

- pai

- Nome
- Altura
- Telefone
- e-mail

- filho

- Otimiza a produção de software
- Esse pilar pode variar entre linguagens de programação

Projeto Orientado a Objetos – Polimorfismo

- Ligado ao conceito de herança:
  - Trata-se da redefinição de métodos herdados de um objeto ancestral

Ligar

- Girar o botão “power” no painel lateral

Ligar

- Pressionar o botão “power” no painel inferior

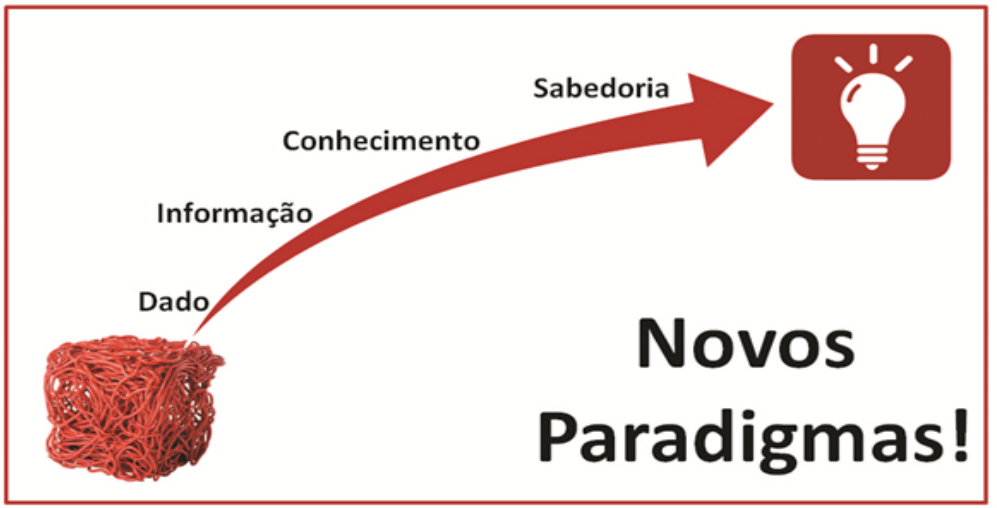
Ligar

- Pressionar o botão “power” no controle remoto

Projeto Orientado a Objetos

Dificuldades

- Complexidade no aprendizado
  - Conceitos



Benefícios

- Facilidade para descrever o mundo
  - Manutenção do código
  - Reutilização de código

Projeto Orientado a Objetos

Projeto Orientado a Objetos

Animal

Projeto Orientado a Objetos

Qual o barulho que esse animal faz?

Animal

Projeto Orientado a Objetos

```
// instancia uma variável para cada animal
Galinha galinha = new Galinha();
Gato gato = new Gato();
Cachorro cachorro = new Cachorro();
Porco porco = new Porco();

// a variavel animal pode ser qualquer animal e irá apontar para um deles
Animal animal;

// nosso colega apontou para um gato
animal = gato;

// ao perguntar qual barulho o animal faz?
animal.fazBarulho();
// a resposta foi: miau !!
```

Classes

- Modelo ou esquema abstrato
- Classes definem os objetos que compõem o sistema de software:
  - Comportamento (métodos)
  - Dados (atributos)
- Representa um ou mais objetos com características afins

Objetos

- Um objeto é uma instância de uma classe
  - Classe : Pessoa
  - Objetos : Anderson, Maria, …
- Seu estado é dado através de seus atributos
  - Basicamente o que caracteriza o objeto. Ex. altura, RG, …
  - Objeto: Anderson → altura=175, RG=1111111111
- Comunica-se com outros objetos através de seus métodos
  - Chamadas de métodos entre objetos, trocando ou não informações

Atributos

- São os dados do objeto:
  - altura, nome, peso …
- Podem ser tipos primitivos ou referências:
  - Primitivos: int, float, char... ;
  - Referências: outros objetos

Métodos

- Definem o comportamento do objeto, suas ações:
  - Anderson fala, anda, come, bebe …
- Permite a comunicação entre objetos:
  - Interface externa;
  - Recebe argumentos (parâmetros) que podem ser utilizados para definir a ação:
    - anda(“direita”); anda(“esquerda”);
  - Podem alterar o estado do objeto (seus atributos)

Métodos – parâmetros

- São dados recebidos pelos métodos e que são visíveis somente por ele mesmo:
  - anda(“direito”); anda(“esquerda”);

- Retorno do método:
  - Caso o método produza algum resultado ou precise passar alguma informação para o objeto solicitante
  - Em Java é necessário informar o tipo de dado que será retornado:
    - ***void*** – indica que o método não possui um retorno;

Visibilidade

- Nos diagramas UML:
  - (+) **público –** outras classes possuem acesso
  - (-) **privado –** acesso somente pela própria classe
  - (#) **protegido –** acessado somente por classes dentro de um mesmo pacote ou por herança
- Na linguagem Java:
  - ***public, private, protected***

Construtor da Classe – Sintaxe Java

- Método com o mesmo nome da Classe:
  - Permite a instanciação dos objetos
  - Pode receber parâmetros como entrada

- Instanciação de objetos em Java:
  - Construtor ***"new"***

```
Pessoa p1 = new Pessoa("Anderson", 1.75);
```

Sintaxe Java

```
public class Pessoa{
   String nome;
   float altura;
   Pessoa(String nome, float altura){
       this.nome = nome;
       this.altura = altura;
   }
   public void anda(String direcao){
      System.out.println("Andei para a
"+direcao);
   }
}
```

Identificando Classes

- **Regra básica**
  - Procurar substantivos e verbos na análise do problema.

- Sistema de gerenciamento de pedidos:

  "*Desenvolver um sistema que realiza o gerenciamento de pedidos, ao receber cada item adicioná-lo ao pedido ao final aplicar o pedido ao sistema geral e enviar para o endereço de entrega*"

- **Substantivos**: sistema, pedidos, item, endereço de entrega,...;
- **Verbos**: adicionar, enviar, aplicar

Exercícios – Definindo classes

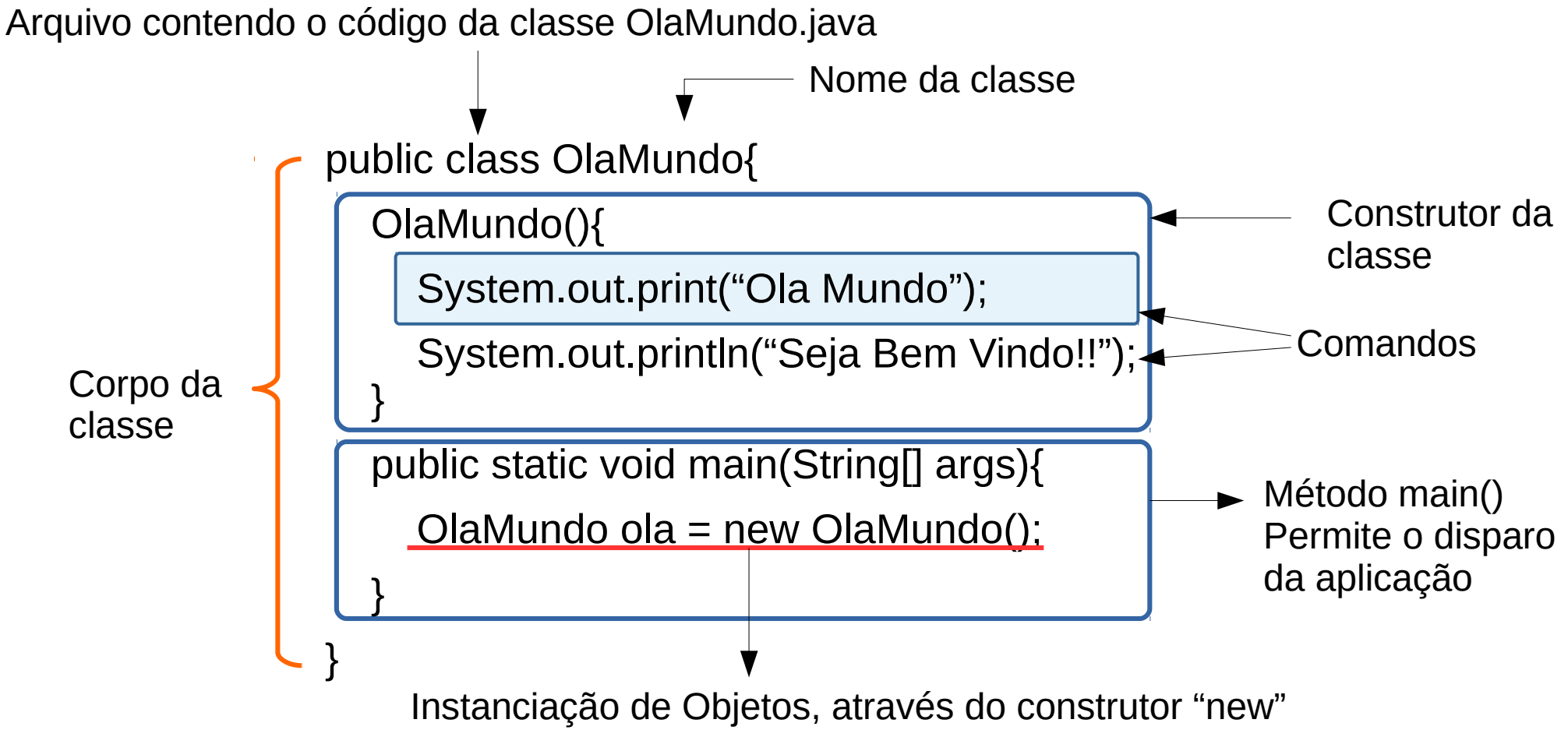
- Defina uma classe que represente uma pessoa (contexto de um cadastro de uma loja), identificando o nome da classe, seus atributos e seus métodos.

- Defina classes que representem cada um dos objetos listados abaixo, identificando o nome da classe, seus atributos e seus métodos.
  - Mesa
  - Cadeira
  - Porta
  - Computador

- Defina uma classe que representa um aluno (contexto de um cadastro da universidade), identificando o nome da classe, seus atributos e seus métodos.

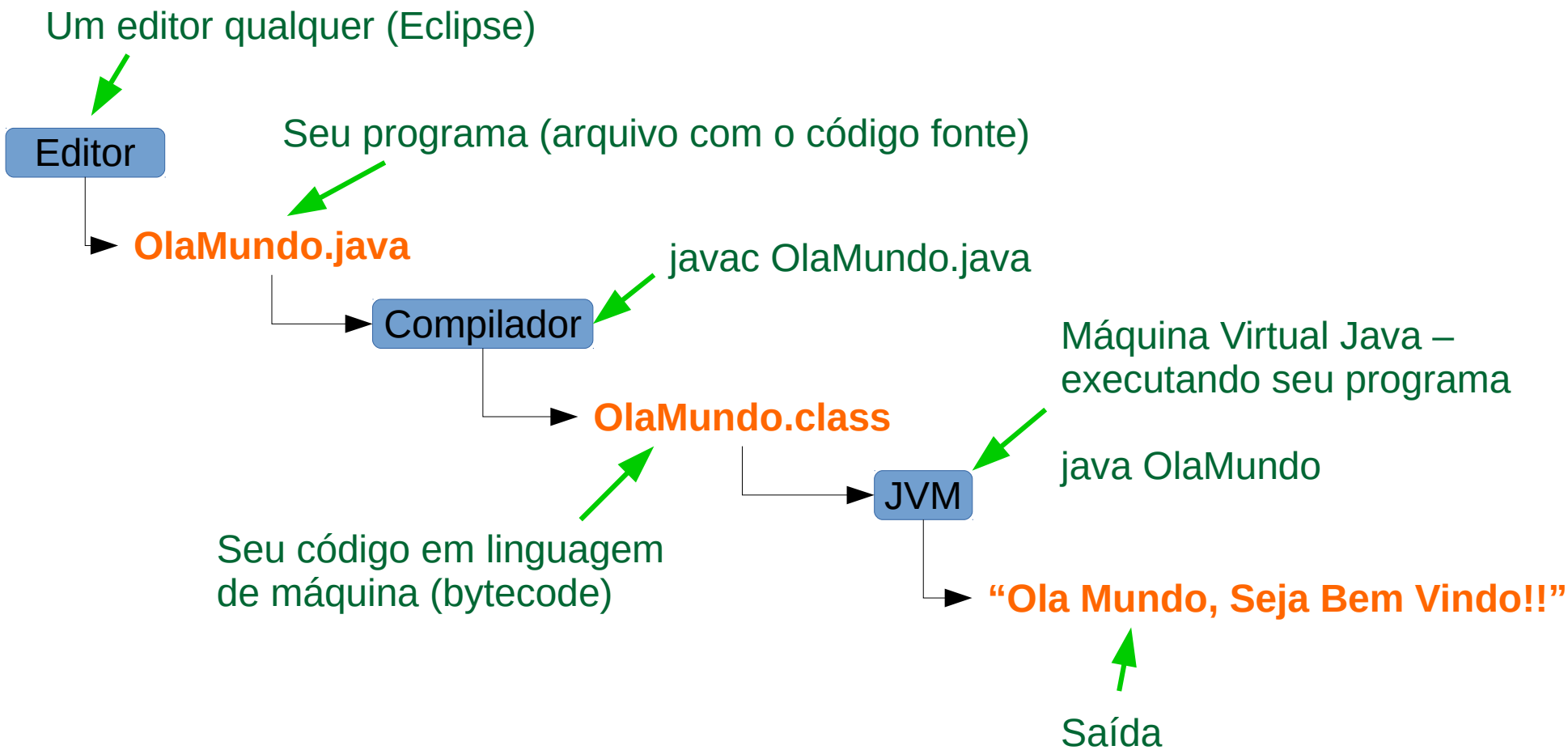
Java Construções Básicas

Estrutura básica de código



Processo de compilação

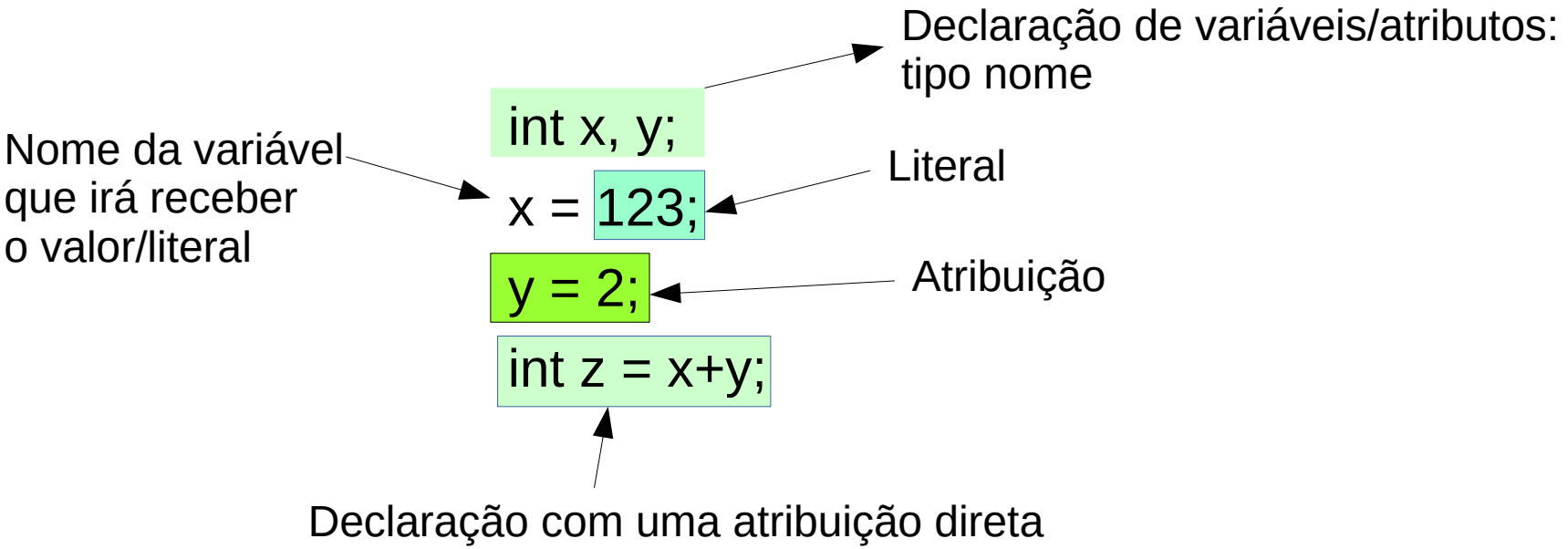
Arquivo contendo o código da classe OlaMundo.java



Tipos de Dados

Tipo	Valores	Operadores	Exemplo de valores
int	Inteiros	+, -, *, /, %	99, -12, 214183123
double	Reais (ponto-flutuante)	+, -, *, /	3.13, -5.2, 3.56e21
boolean	Valores lógicos	&& (e), || (ou), ! (não)	true, false
char	Caracteres		'a', 'A', '1', '@'
String	Sequência de Caracteres	+ (concatenação)	“Anderson”, “ola”, “2.1”

Atribuições



Operadores relacionais

Operador	Significado	verdadeiro	falso
==	Igual a	4==4	4==5
!=	Diferente de	4!=5	5!=5
<	Menor que	4 < 5	5 < 4
<=	Menor ou igual a	4<=4	5<=2
>	Maior que	5>4	2>4
>=	Maior ou igual a	5>=2	2>=5

São utilizados em estruturas condicionais:

```
if(x==y){
        System.out.println(“São iguais”);
}else{
        System.out.println(“São diferentes”);
}
```

Laços

```
int i=0, v=2;
while(i<=n){
    v = v*2;
    i++;
}
```

Inicialização do controlador

Condicional, verifica se vai continuar o bloco de operação

Atribuição equivalente a i=i+1

```
int v=2;
for(int i=0; i<=n; i++){
    v = i*2;
}
```

Inicialização do controlador

Condicional, verifica se vai continuar o bloco de operação

Incremento

Métodos

Assinatura do método (public double somar(double x, double y);)