



Objetos e Classes em Java

Programação Orientada a Objetos

Tércio de Moraes

¹Ciência da Computação
Campus Arapiraca
Universidade Federal de Alagoas – UFAL

Arapiraca, 2 de agosto de 2024





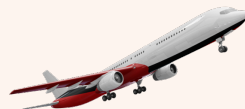
Agenda

- 1 Entendendo classes e objetos
- 2 Java e Orientação a Objetos
- 3 A seguir

Objetos

Objetos

- ➡ É uma entidade física ou conceitual
- ➡ Possui uma **identidade**, **propriedades** e **operações**
- ➡ Pode ser construído e destruído
- ➡ O objeto **carro**
 - ➡ **Propriedades**: modelo, fabricante, ano, motor, combustível ...
 - ➡ **Operações**: ligar, desligar, acelerar, ligar faróis ...
- ➡ O objeto **aeronave**
 - ➡ **Propriedades**: modelo, fabricante, ano, motor, combustível, **altitude** ...
 - ➡ **Operações**: ligar, desligar, acelerar, ligar faróis, **acionar flaps** ...





Objetos em Java

Tipos de Dados

➡ Muitas variáveis que criamos em Java são **objetos**

➡ Por exemplo:

➡ **String**

➡ **Vector**

➡ Estes **tipos de dados** são **classes**

➡ **Tipos de dados** que iniciam com **letra maiúscula** são **classes de objetos**

Exemplo



```
1 public class ObjectsDemo{
2     public static void main(String[] args){
3         String nome      = "Joao dos Santos";
4         int varNum        = 36;
5         Integer objNum    = new Integer(36);
6         sysout("Valor minimo de um inteiro: "
7 + objNum.MIN_VALUE);
8         sysout("Valor maximo de um inteiro: "
9 + objNum.MAX_VALUE);
10        // Tente fazer o mesmo para varNum
11    }
12 }
```




Objetos em Java


Tipos de Dados

- ➡ Muitas variáveis que criamos em Java são **objetos**
- ➡ Por exemplo:
 - ➡ **String**
 - ➡ **Vector**
- ➡ Estes **tipos de dados** são **classes**
- ➡ **Tipos de dados** que iniciam com **letra maiúscula** são **classes de objetos**

Exemplo



```
1 public class ObjectsDemo{
2     public static void main(String[] args){
3         String nome      = "Joao dos Santos";
4         int varNum        = 36;
5         Integer objNum    = new Integer(36);
6         sysout("Valor minimo de um inteiro: "
7             + objNum.MIN_VALUE);
8         sysout("Valor maximo de um inteiro: "
9             + objNum.MAX_VALUE);
10        // Tente fazer o mesmo para varNum
11    }
12 }
```





Objeto :: Um Exemplo com Vector



```
1 import java.util.Vector;
2
3 public class VectorDemo {
4     public static void main(String[] args) {
5         Vector vetor = new Vector();
6         vetor.add("Maria da Silva");
7         vetor.add("Joao dos Santos");
8         System.out.println("Tamanho da lista: " + vetor.size());
9         System.out.println("Elemento na posicao 0: " + vetor.elementAt(0))
10        ;
11        System.out.println("Elemento na posicao 1: " + vetor.elementAt(1))
12        ;
13    }
14 }
```



Objeto :: Relação entre **Propriedades** e **Operações**

Voltando à aeronave ...

- Os controles da aeronave devem ter **sistemas de segurança** próprios
- Por exemplo, sistemas de alarmes avisam quando algo está errado
- Algumas funções não podem ser executadas indiscriminadamente

Propriedades e operações

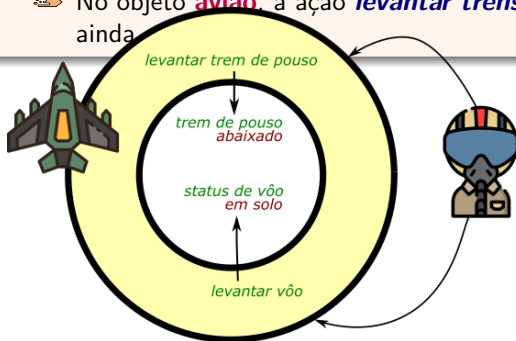
- **Propriedades** de um objeto precisam ser **protegidas** de ações externas
- **Métodos** exercem este papel de **proteger** uma **propriedade** de um objeto
 - No objeto **avião**, a ação **levantar trens de pouso** deve ser bloqueada caso o avião ainda esteja em solo



Objeto :: Relação entre **Propriedades** e **Operações**

Propriedades e operações

- **Propriedades** de um objeto precisam ser **protegidas** de ações externas
- **Métodos** exercem este papel de **proteger** uma **propriedade** de um objeto
 - No objeto **avião**, a ação **levantar trem de pouso** deve ser bloqueada caso o avião ainda

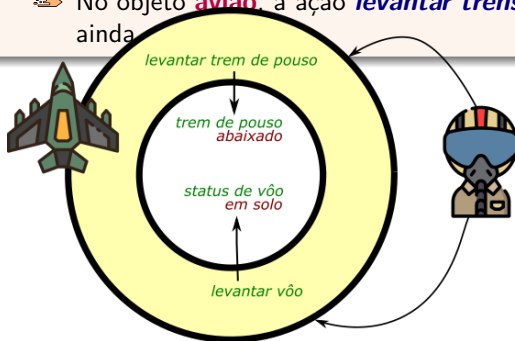




Objeto :: Relação entre **Propriedades** e **Operações**

Propriedades e operações

- **Propriedades** de um objeto precisam ser **protegidas** de ações externas
- **Métodos** exercem este papel de **proteger** uma **propriedade** de um objeto
 - No objeto **avião**, a ação **levantar trem de pouso** deve ser bloqueada caso o avião ainda



- Como definir um Tipo **“Avião”**
- Como **proteger** estes dados?
- Como implementar em **Java**?





Classes :: Voltando um Pouco para Tipos de Dados

Inteiro

- **Propriedade:** números inteiros
- **Operações:** soma, multiplicação, divisão

Float

- **Propriedade:** números reais
- **Operações:** soma, multiplicação, divisão

Operações sobre valores

- Estrutura de tipos de dados:
 - **Conjunto de valores** que podem ser armazenados
 - **Conjunto de operações** que podem ser realizadas sobre os respectivos valores
- Sempre que necessário, **criamos** variáveis de tipos de dado
- Como constrói um...

Podemos construir tipos de dados de acordo com o problema abordado: **Classes**



Classes :: Voltando um Pouco para Tipos de Dados

Inteiro

- **Propriedade:** números inteiros
- **Operações:** soma, multiplicação, divisão

Float

- **Propriedade:** números reais
- **Operações:** soma, multiplicação, divisão

Operações sobre valores

- Estrutura de tipos de dados:
 - **Conjunto de valores** que podem ser armazenados
 - **Conjunto de operações** que podem ser realizadas sobre os respectivos valores
- Sempre que necessário, **criamos** variáveis de tipos de dado
- Como constrói um **tipo de dado**?

*Podemos construir tipos de dados de acordo com o problema abordado: **Classes***



Classes :: Voltando um Pouco para Tipos de Dados

Inteiro

- **Propriedade:** números inteiros
- **Operações:** soma, multiplicação, divisão

Float

- **Propriedade:** números reais
- **Operações:** soma, multiplicação, divisão

Operações sobre valores

- Estrutura de tipos de dados:
 - **Conjunto de valores** que podem ser armazenados
 - **Conjunto de operações** que podem ser realizadas sobre os respectivos valores
- Sempre que necessário, **criamos** variáveis de tipos de dado
- Como constrói um **tipo de dado**?

Podemos construir tipos de dados de acordo com o problema abordado: Classes



Classes :: Voltando um Pouco para Tipos de Dados

Inteiro

- **Propriedade:** números inteiros
- **Operações:** soma, multiplicação, divisão

Float

- **Propriedade:** números reais
- **Operações:** soma, multiplicação, divisão

Operações sobre valores

- Estrutura de tipos de dados:
 - **Conjunto de valores** que podem ser armazenados
 - **Conjunto de operações** que podem ser realizadas sobre os respectivos valores
- Sempre que necessário, **criamos** variáveis de tipos de dado
- Como constrói um **tipo de dado**?

Podemos construir tipos de dados de acordo com o problema abordado: Classes



Classes

Voltando ao avião...

- As **propriedades** do avião são, relativamente, mais complexas: múltiplos **atributos** de **tipos de dados diferentes**
- As **operações** são executadas com base nos valores de seus **atributos/propriedades**
- As operações são **métodos** de como um avião deve funcionar
- **Métodos/operações** lêem e atualizam os **atributos**
- Exemplo: decolar requer a **verificação** de vários **atributos** da aeronave antes, durante e após a decolagem, bem como a **atualização** de dados de **atributos** (eg.: altitude)

Projeto de avião

- Todo **avião** (um objeto) é construído a partir de um **projeto/template**
- Em OO, a **classe** é o projeto de **objeto**
- Questão: como a matrícula do curso funcionaria sem Computação?



Classes :: A Classe Avião

Exemplo de objetos

Contexto **Georreferenciamento**

Podemos representar nosso avião pela sua **posição** no globo e sua **altitude** em relação ao nível do mar

Atributos:

- `id`
- `latitude`
- `longitude`
- `altitude`

Métodos:

- `Ler atributos`
- `Atualizar atributos`



Classes :: A Classe Avião

Método de leitura

`getLatitude(), getLongitude(), ...`

Exemplo de objetos

➤ Contexto **Georreferenciamento**

➤ Podemos representar nosso avião pela sua **posição** no globo e sua **altitude** em relação ao nível do mar

➤ Atributos:

➤ `id`

➤ `latitude`

➤ `longitude`

➤ `altitude`

➤ Métodos:

➤ **Ler atributos**

➤ **Atualizar atributos**



Classes :: A Classe Avião

Exemplo de objetos

➤ Contexto **Georreferenciamento**

➤ Podemos representar nosso avião pela sua **posição** no globo e sua **altitude** em relação ao nível do mar

➤ Atributos:

- **id**
- **latitude**
- **longitude**
- **altitude**

➤ Métodos:

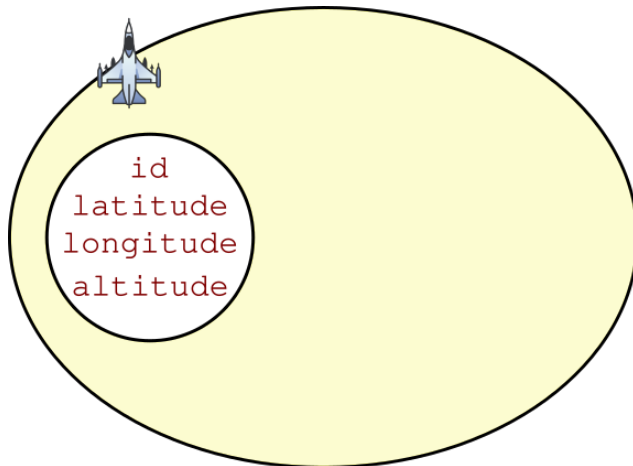
- **Ler atributos**
- **Atualizar atributos**

Método de escrita

setLatitude(), setLongitude(), ...

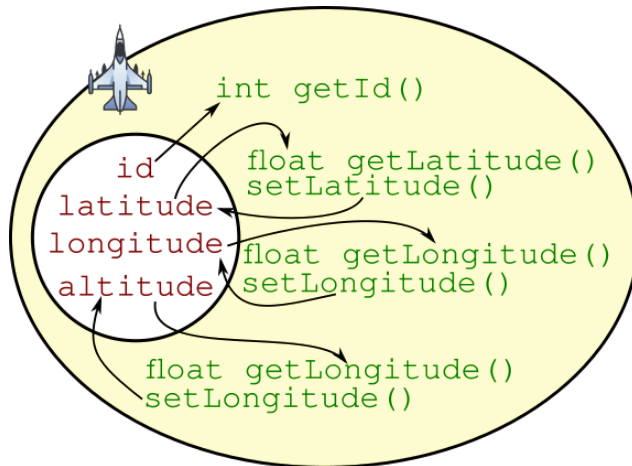


Classe :: Anatomia da Classe Avião



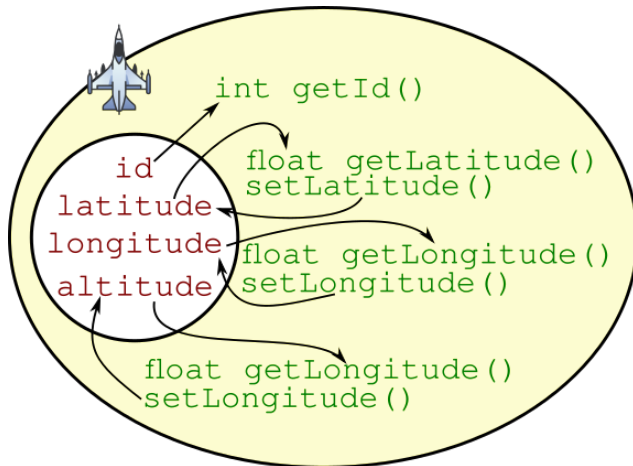


Classe :: Anatomia da Classe Avião





Classe :: Anatomia da Classe Avião



Por que as setas de acesso apontam de forma diferente?



Classes :: Criando Classes em Java

Estrutura de uma classe

- Atributos
- Métodos

Sintaxe

```
1 public class <NomeClasse>{  
2     <atributos>  
3     <metodos>  
4 }
```

Classe Avião

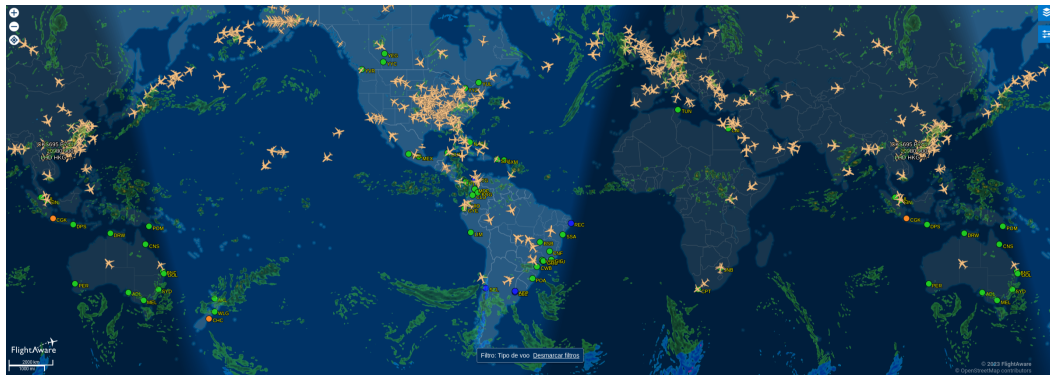
```
1 public class AirPlane{  
2     int id;  
3     float latitude;  
4     float longitude;  
5     float altitude;
```

```
1     public int getId(){  
2         return id;  
3     }  
4     public void setId(int id){  
5         id = id  
6     }  
7 }
```





O Objeto Avião e o BoobleMAPS



<https://pt.flightaware.com/live/map>



Construindo e Usando Objetos



```
1 public class BoobleMaps{
2     public static void main(String[] args) {
3         AirPlane av00 = new AirPlane();
4         av00.setId(00);
5         av00.setLatitude(-9.7014f);
6         av00.setLongitude(-36.6868f);
7         av00.setAltitude(400);
8         System.out.println("Posicioamento da aeronave " + av00.getId());
9         System.out.println("  Latitude:   " + av00.getLatitude() + " graus");
10        System.out.println("  Longitude:  " + av00.getLongitude() + " graus");
11        System.out.println("  Altitude:   " + av00.getAltitude() + " m");
12    }
13 }
```



Construtores de Objetos

Construtor padrão

- Toda classe tem um **construtor padrão**
- **Não é necessário implementar** o construtor padrão:

```
➤ AirPlane av00 = new AirPlane();
```
- O que acontece nesta linha de código:
 - **Alocação de memória** para o objeto
 - **Endereço de memória** armazenado em **av00**
- Variáveis armazenam a **referência** ao objeto

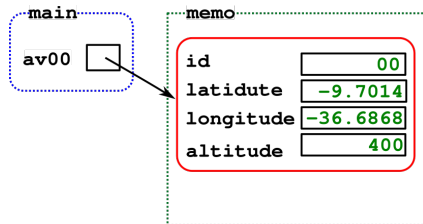


Construtores de Objetos

Construtor padrão

- Toda classe tem um **construtor padrão**
- **Não é necessário implementar** o construtor padrão:

```
AirPlane av00 = new AirPlane();
```
- O que acontece nesta linha de código:
 - **Alocação de memória** para o objeto
 - **Endereço de memória** armazenado em **av00**
- Variáveis armazenam a **referência** ao objeto





Construtores de Objetos

Construtor padrão

- Toda classe tem um **construtor padrão**
- **Não é necessário implementar** o construtor padrão:
 - `AirPlane av00 = new AirPlane();`
- O que acontece nesta linha de código:
 - **Alocação de memória** para o objeto
 - **Endereço de memória** armazenado em `av00`
- Variáveis armazenam a **referência** ao objeto

Sintaxe

```
1 public <NomeClasse>(){ // Mesmo nome da classe
2     <listaComandos>
3 }
```



Construtores Personalizados

Podemos criar vários construtores

- Objeto com **valores iniciais**
- Podemos definir **parâmetros de entrada**
- Sintaxe:

```
1 public <NomeClasse>(<listaParametros>){  
2     <listaComandos>  
3 }
```



Construtores Personalizados



Podemos criar vários construtores

- ➡ Objeto com **valores iniciais**
- ➡ Podemos definir **parâmetros de entrada**
- ➡ Sintaxe:

```
1 public <NomeClasse>(<listaParametros>){  
2     <listaComandos>  
3 }
```

```
1 public AirPlane(int id){  
2     setId(id);  
3 }  
4  
5 public AirPlane(float lat,  
6     float lon){  
7     int id= Math.trunc(lat+lon);  
8     setId(id);  
9 }
```

- ➡ **Instanciando** o objeto:
- ➡ AirPlane av01 = **new** AirPlane(32);
- ➡ **Crie** objetos com o outro construtor
- ➡ **Crie** outros construtores





Construtores Personalizados

Podemos criar vários construtores

- Objeto com **valores iniciais**
- Podemos definir **parâmetros de entrada**
- Sintaxe:

```
1 public <NomeClasse>(<listaParametros>){  
2     <listaComandos>  
3 }
```

Atenção

- Classe com seus **próprios construtores não tem acesso** ao **construtor padrão**
- O construtor padrão deve ser **explícito**



```
1 public AirPlane(int id){  
2     setId(id);  
3 }  
4  
5 public AirPlane(float lat,  
6     float lon){  
7     int id= Math.trunc(lat+lon);  
8     setId(id);  
9 }
```

- Instanciando** o objeto:
- AirPlane av01 = **new** AirPlane(32);
- Crie** objetos com o outro construtor
- Crie** outros construtores





Voltando ao Exemplo do **main**



```
1 public class BoobleMaps{
2     public static void main(String[] args) {
3         AirPlane av00 = new AirPlane();
4         av00.setId(00);
5         av00.setLatitude(-9.7014f);
6         av00.setLongitude(-36.6868f);
7         av00.setAltitude(400);
8         System.out.println("Posicioamento da aeronave " + av00.getId());
9         System.out.println("    Latitude:    " + av00.getLatitude() + " graus");
10        System.out.println("    Longitude:   " + av00.getLongitude() + " graus");
11        System.out.println("    Altitude:    " + av00.getAltitude() + " m");
12    }
13 }
```

Utilizamos **métodos** para acessar os **atributos**



Voltando ao Exemplo do **main**: Acesso aos Atributos

Podemos **acessar** os **atributos** diretamente?



```
1 public class BoobleMaps{
2     public static void main(String[] args) {
3         AirPlane av00 = new AirPlane();
4         av00.setId(00);
5         av00.setLatitude(-9.7014f);
6         av00.setLongitude(-36.6868f);
7         av00.setAltitude(400);
8         System.out.println("Posicioamento da aeronave " + av00.id);
9         System.out.println("  Latitude:   " + av00.latitude + " graus");
10        System.out.println("  Longitude:  " + av00.longitude + " graus");
11        System.out.println("  Altitude:   " + av00.altitude + " m");
12    }
13 }
```



Voltando ao Exemplo do **main**: Acesso aos Atributos

Podemos **acessar** os **atributos** diretamente? **DEVEMOS?**



```
1 public class BoobleMaps{
2     public static void main(String[] args) {
3         AirPlane av00 = new AirPlane();
4         av00.setId(00);
5         av00.setLatitude(-9.7014f);
6         av00.setLongitude(-36.6868f);
7         av00.setAltitude(400);
8         System.out.println("Posicioamento da aeronave " + av00.id);
9         System.out.println("  Latitude:   " + av00.latitude + " graus");
10        System.out.println("  Longitude:  " + av00.longitude + " graus");
11        System.out.println("  Altitude:   " + av00.altitude + " m");
12    }
13 }
```




A seguir

- Introdução à linguagem Java ✓
- Classes e objetos ✓
- **Encapsulamento** e **Modificadores de acesso**
- Abstração, herança e interface
- Polimorfismo

