Modigicadores de Acesso e Encapsulamento

Programação Orientada a Objetos

Tércio de Morais

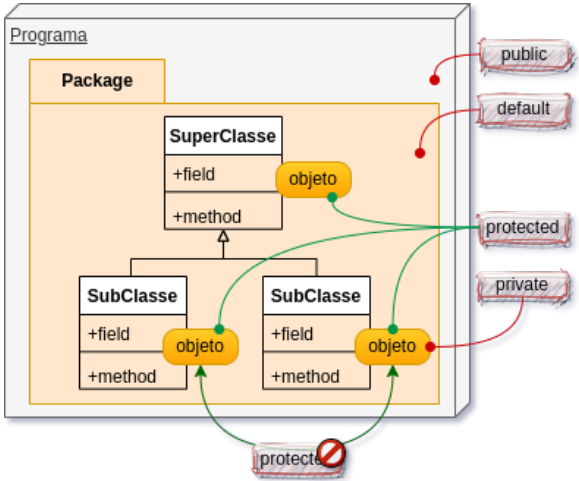
[1]Ciência da Computação
Campus Arapiraca
Universidade Federal de Alagoas – UFAL

Arapiraca, 15 de agosto de 2024

Ciência da Computação
UFAL Arapiraca

Agenda

"Previously in POO . . . "



Escopos de Variáveis

Modificadores de acesso

- Os modificadores determinam a **"visibilidade"** de uma variável:
  - `public` – Acessível de **qualquer lugar**
  - `protected` – Acessível a partir da **classe** que pertence e suas **subclasses**
  - `default (package)` – Acessível no contexto do ***namespace*** da classe (*package*)
  - `private` – Acessível **apenas no objeto** da classe em que foi definida
- **Modificadores de acesso** também são usados para **métodos** e **classes**

Sintaxe

Sintaxe:

```
<modificador> <tipo_dado> <nome_var>;
```

Exemplo:

```
private String palavra;
```

Exemplo de Modificadores


```java
public class Class1 {
  int number1 = 1;
}
```

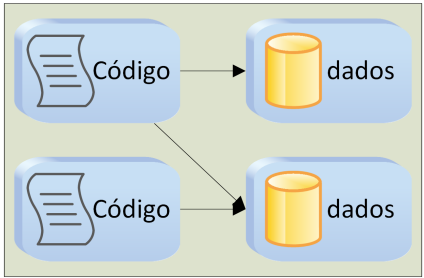
```java
public class Class2 {
  int number2 = 2;
}
```

```java
public class EscopeDemo {
  public static void main(String[] args) {
    Class1 obj1 = new Class1();
    Class2 obj2 = new Class2();
    obj1.number1 = 20;
    System.out.println(obj1.number1);
    System.out.println(obj2.number2);
  }
}
```

Altere os modificadores das variáveis **number1** e **number2** nas classes **Class1** e **Class2**

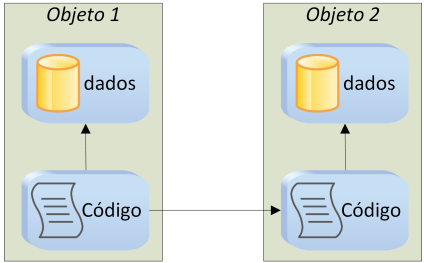
Fatos Básicos da Programação Imperativa

- Um dos principais problemas da programação estruturada é o **"acesso descontrolado"** a um conjunto de dados
- Podemos **acessar** um conjunto de dados de **qualquer parte** de um programa



Fatos Básicos da Programação Imperativa

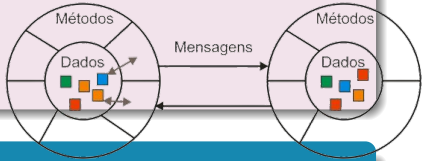
- **Modificadores de acesso** permitem **controlar** o acesso a um conjunto de dados
- O desenvolvedor tem **controle** sobre quais programas **podem alterar** e **como podem alterar** informações



Encapsulamento

Por que voltamos a falar sobre modificadores

- **Modificadores** como ferramenta de implementação do **encapsulamento**
- Mais precisamente, o modificador `private`

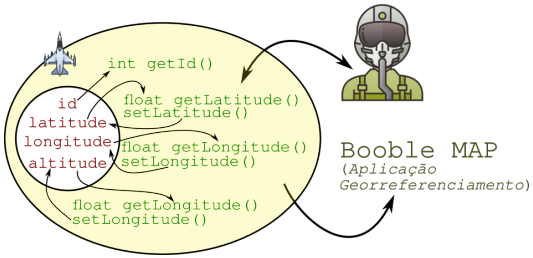


O que é encapsulamento

- Técnica para **ocultar** uma **propriedade/atributo** de um objeto
- Um **atributo** não pode ser **acessado diretamente** por códigos externos ao objeto
- o **encapsulamento** permite uma programação **mais segura**
- O **acesso aos atributos** só ocorre através de **métodos** onde regras de manipulação do atributo são implementadas

Encapsulamento em Java

- Voltando ao nosso exemplo do avião

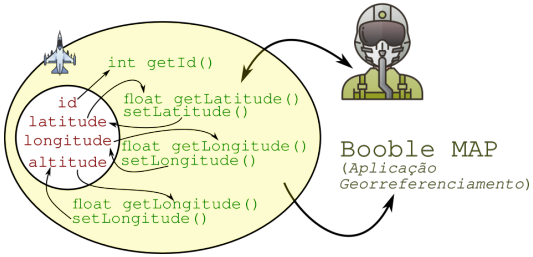


- `Latitude` tem valores entre 0^o e 90^o
- `Longitude` tem valores entre 0^o e 180^o

```
1 public class AirPlane{
2   private int id;
3   private float latitude;
4   private float longitude;
5   private float altitude;
6   ...
7 }
```

Encapsulamento em Java

- Voltando ao nosso exemplo do avião



- **Latitude** tem valores entre 0^o e 90^o
- **Longitude** tem valores entre 0^o e 180^o

```
1  public class AirPlane{
2    ...
3    public float getLatitude() {
4      return latitude;
5    }
6    public void setLatitude(float
   latitude) {
7        if(latitude >= 0
8            && latitude <= 90){
9          this.latitude = latitude;
10       }
11   }
12   ...
13 }
```

Encapsulamento em Java :: Outro Exemplo

Semáforo

Crie uma classe **Semáforo** obedecendo as regras de encapsulamento e as seguintes restrições

- O estado do semáforo deve ser representado apenas por cores válidas (vermelho, verde ou amarelo)
- O semáforo é composto por **três lâmpadas** onde **apenas uma deve estar acesa**
- Apenas transições válidas devem ocorrer (**verde** → **amarelo**, **amarelo** → **vermelho** → **verde**)

Mãos na Massa

Ponto e Reta

Construir duas classes que representem um **ponto** e uma **reta** para que seja possível estabelecer e saber a **posição** de um ponto no plano cartesiano e **calcular** o tamanho de uma reta.