Orientação a objetos

Encapsulamento

Encapsulamento

- Em Java, o encapsulamento é implementado usando modificadores de acesso, que são palavras-chave que determinam o nível de acesso aos membros de uma classe (atributos e métodos).
- Os modificadores de acesso em Java são:
  - public: permite que qualquer classe possa acessar o membro.
  - private: permite que somente a própria classe possa acessar o membro.
  - protected: permite que as subclasses e classes do mesmo pacote possam acessar o membro.
  - default (package) : permite que as classes do mesmo pacote possam acessar o membro (é o modificador padrão, se nenhum outro modificador for especificado).

Encapsulamento

- E para termos acesso a estes dados, precisamos também compreender a função dos *getters* e *setters*.
- Os *getters* e *setters* são métodos que permitem o acesso e a modificação dos atributos privados de uma classe, respeitando o conceito de encapsulamento.
- Eles são fundamentais na orientação a objetos, pois permitem que os dados sejam protegidos e acessados de forma controlada.
- Os *getters* são usados para obter o valor de um atributo, enquanto os *setters* são usados para alterar o valor de um atributo.

Encapsulamento

- Em Java, os getters e setters seguem um padrão de nomenclatura, em que o nome do método começa com "get" ou "set", seguido do nome do atributo com a primeira letra em maiúscula.

- Por exemplo, se tivermos um atributo privado chamado "nome", o getter correspondente seria "getNome()" e o setter seria "setNome()".

Encapsulamento - Exemplo

```java
public class Pessoa {
    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }
}
```

```java
public class Pessoa {
    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }
}
```

Encapsulamento - Exemplo

- Neste exemplo, temos uma classe Pessoa com dois atributos privados (nome e idade) e os respectivos *getters* e *setters*.
- A ideia aqui é que os atributos não possam ser acessados ou modificados diretamente de fora da classe, apenas através dos métodos públicos.
- O construtor da classe é responsável por inicializar os atributos.
- Para acessar e modificar os atributos da classe, podemos usar os métodos *getters* e *setters*, como mostrado no próximo slide:

Encapsulamento - Exemplo

```
3  public class Principal {
4
5      public static void main(String[] args) {
6          Pessoa pessoa = new Pessoa("Ricardo Frohlich", 22);
7          System.out.println(pessoa.getNome());
8          System.out.println(pessoa.getIdade());
9
10         pessoa.setNome("Rodrigo Ramos");
11         pessoa.setIdade(25);
12
13         System.out.println(pessoa.getNome());
14         System.out.println(pessoa.getIdade());
15     }
16
17 }
```

Encapsulamento

- Os *getters* e *setters* são uma prática comum na programação orientada a objetos e são amplamente utilizados em diversas aplicações.
- Observe que, ao acessar os atributos nome e idade, estamos usando os métodos públicos getNome() e getIdade(), em vez de acessá-los diretamente.
- Isso garante que o encapsulamento seja respeitado e que os atributos não sejam acessados ou modificados diretamente.
- Eles permitem que os dados de uma classe sejam acessados e modificados de forma controlada e segura, contribuindo para a manutenção e evolução do código.

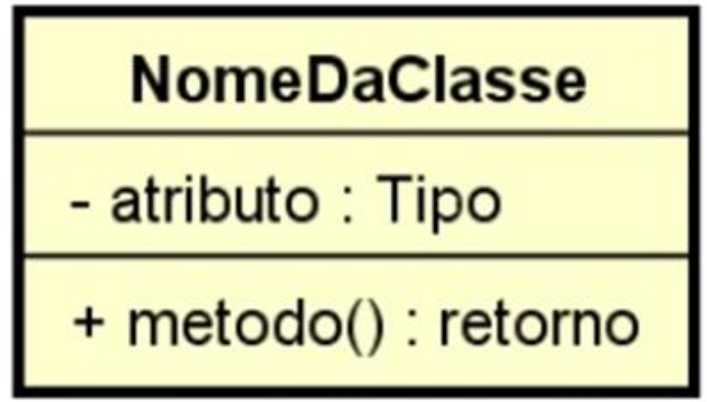
Encapsulamento

- O encapsulamento é um dos conceitos fundamentais da orientação a objetos e é essencial para garantir a integridade dos dados de uma classe.

- É importante lembrar que o encapsulamento não impede completamente o acesso aos atributos de uma, mas fornece uma camada de proteção adicional e ajuda a tornar o código mais robusto e seguro.

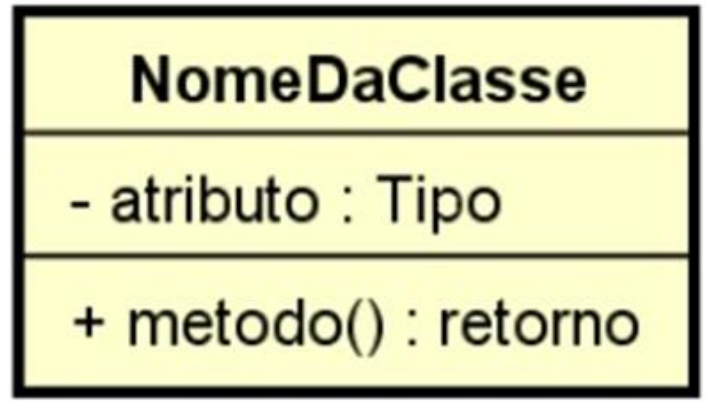
Encapsulamento – Diagrama de classes

- E como compreender em um diagrama de classes?
- O Diagrama é dividido em três partes:
  - Nome da classe
  - Atributos
  - Métodos



Encapsulamento – Diagrama de classes

- Atributos
  - <acesso> <nome> : <tipo>
- Métodos
  - <acesso> <nome> (<parâmetros>) : <tipo>
- <acesso> = público: sinal de “+”
- <acesso> = privado: sinal de “-”
- <acesso> = protegido: sinal de “#”
- <acesso> = default (package): sinal de “~”



Jogo rápido

- Crie uma classe Aluno que possua os atributos nome, nota1 e nota2.
- Proteja os atributos utilizando encapsulamento.
- Crie os métodos *get* e *set* para cada atributo.
- Crie um método calculaMedia que calcule a média das notas do aluno e retorne o resultado.

Resolução

```java
public class Aluno {
    private String nome;
    private double nota1;
    private double nota2;
    public Aluno(String nome, double nota1, double nota2) {
        super();
        this.nome = nome;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public double getNota1() {
        return nota1;
    }
    public void setNota1(double nota1) {
        this.nota1 = nota1;
    }
    public double getNota2() {
        return nota2;
    }
    public void setNota2(double nota2) {
        this.nota2 = nota2;
    }
    public double calculaMedia() {
        return (nota1+nota2)/2;
    }
}
```

```java
public class Aluno {
    private String nome;
    private double nota1;
    private double nota2;
    public Aluno(String nome, double nota1, double nota2) {
        super();
        this.nome = nome;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public double getNota1() {
        return nota1;
    }
    public void setNota1(double nota1) {
        this.nota1 = nota1;
    }
    public double getNota2() {
        return nota2;
    }
    public void setNota2(double nota2) {
        this.nota2 = nota2;
    }
    public double calculaMedia() {
        return (nota1+nota2)/2;
    }
}
```

Resolução

```
5  public class Principal {
6
7      public static void main(String[] args) {
8          Aluno a = new Aluno("Ricardo", 7.43 , 5.8);
9          System.out.println("Nome do aluno: "+a.getNome());
10         System.out.println("Nota 1: "+a.getNota1());
11         System.out.println("Nota 2: "+a.getNota2());
12         System.out.println("Média final: "+a.calculaMedia());
13     }
14 }
```

Resolução

```
5  public class Principal {
6
7      public static void main(String[] args) {
8          Aluno a = new Aluno("Ricardo", 7.43 , 5.8);
9          System.out.println("Nome do aluno: "+a.getNome());
10         System.out.println("Nota 1: "+a.getNota1());
11         System.out.println("Nota 2: "+a.getNota2());
12         System.out.println("Média: "+a.calculaMedia());
13         a.setNota1(7.90);
14         a.setNota2(8.43);
15         System.out.println("Média final: "+a.calculaMedia());
16     }
17 }
```

Exercícios

- 1 - Crie uma classe ContaBancaria que possua os atributos saldo e limite. Proteja os atributos utilizando encapsulamento. Crie os métodos get e set para cada atributo. Crie um método saque que permita ao usuário sacar um valor da conta, desde que não ultrapasse o limite da conta. Faça leitura pelo teclado.

- 2 - Crie uma classe Circulo que possua o atributo raio. Proteja o atributo utilizando encapsulamento. Crie os métodos get e set para o atributo. Crie um método calculaArea que calcule a área do círculo e retorne o resultado. Faça leitura pelo teclado dos valores.

Exercícios

- 3 - Crie uma classe Retangulo que possua os atributos base e altura. Proteja os atributos utilizando encapsulamento. Crie os métodos get e set para cada atributo. Crie um método calculaArea que calcule a área do retângulo e retorne o resultado. Faça leitura pelo teclado dos valores.

- 4 - Crie uma classe Carro que possua os atributos marca, modelo e ano. Proteja os atributos utilizando encapsulamento. Crie os métodos get e set para cada atributo. Crie um método exibeDetalhes que exibe os detalhes do carro. Faça leitura pelo teclado dos valores.