Departamento de Engenharias e Computação
Colegiado de Ciência da Computação
Disciplina: Linguagem de Programação III

Cap 2 – Conceitos Básicos de Orientação a Objetos

Parte 2 – Encapsulamento e Modificadores de Acesso

Professor: Otacílio José Pereira

Plano de Aula

- ***Objetivos***
  - Compreender o que é encapsulamento
  - Exercitar modificadores de acesso como meio para implementar o encapsulamento

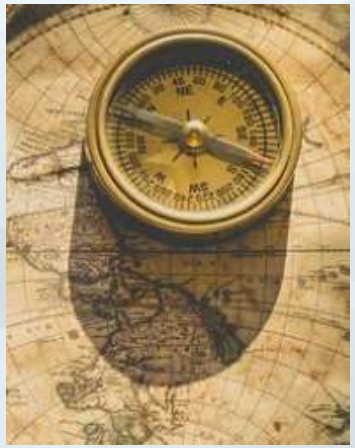
- ***Tópicos***
  - Contexto da disciplina
  - Encapsulamento: Refletindo sobre um problema
  - A classe como uma “cápsula”
  - Modificadores de acesso
  - Getters e setters
  - Exercitando com Conta
  - Exercício de Sala

Contexto

- Onde estamos?
- Foco agora!
- Cenário de exemplo

Onde estamos?

- *Considerando nosso planejamento inicial*



- Capítulo 1 – Introdução
- Capítulo 2 - Conceitos básicos de Orientação a Objetos
  - Parte 1 : Classes, objetos, atributos e métodos
  - Parte 2 : Encapsulamento, modif. de acesso, getters e setters
  - Parte 3 : Construtores
  - Parte 4 : Relação entre classes
- Capítulo 3 – Herança e Polimorfismo
- Capítulo 4 – Classes abstratas e interfaces
- Capítulo 5 – Generics, collections e outros tópicos
- Capítulo 6 – Desenvolvimento de um projeto em OO

Foco

Onde estamos?

- Neste momento que estamos discutindo os conceitos básicos de OO, organizamos nossa trajetória nos seguintes tópicos
  - **(Ok) Parte 1**: Classes, objetos, atributos e métodos e discussões iniciais sobre orientação a objetos
  - **(Foco agora) Parte 2**: Encapsulamento, modificadores de acesso, getters e setters e sobrecarga de métodos
  - **Parte 3**: Construtores e instanciação de objetos
  - **Parte 4**: Atributos e métodos de classe
  - **Parte 5**: Outros elementos básicos

Encapsulamento

- Refletindo um problema
- Encapsulamento
- Vantagens no projeto
- Modificadores de acesso

Refletindo um problema

- Conforme vimos na aula passada
  - Classes são moldes para a criação de objetos, as instâncias ou exemplares de uma classe
    - Temos a classe Conta com os seus atributos
    - Existem um objeto Conta com número 125 e saldo 550,00 e um outro objeto ou exemplar com número 200 e saldo de 230,00
  - Em nossa implementação, podemos acessar os atributos para alterar seus valores

```
Conta c1 = new Conta ();
c1.Numero = 125;                    // Acesso ao atributo diretamente
c1.Saldo = 300,00;
c1.Depositar(350,00);
```

Refletindo um problema

- Qual valor será impresso no código a seguir?

```
Conta c1 = new Conta ();
c1.Numero = 125;          c1.Saldo = 0;
c1.Depositar(150.00);
c1.Depositar(250.00);
c1.Sacar(100.00);
c1.Saldo = 3000,00;
c1.Sacar(600.00);
c1.Saldo = 1800.00;
c1.Depositar(100.00);
System.out.print(“ Saldo : “ + c1.Saldo());
```

- **Discuta qual o problema desta solução?**

Encapsulamento

- **Definição e conceito**
  - Encapsulamento é uma técnica que visa ocultar os atributos de uma classe/objeto e fazer com que estes sejam acessados apenas por métodos
  - Isso permite melhor controle sobre os valores de atributos de um determinado objeto
  - Para compreender a vantagem do uso de encapsulamento retome o exemplo dos valores do atributo Saldo e a sua manipulação exclusivamente pelos métodos Sacar e Depositar

Encapsulamento

- **Alguns outros exemplos**
  - Ao encapsular informações, você esconde ou abstrai os detalhes de como a implementação de algo foi realizado.
  - Exemplo 1:
    - Imagine um cenário em que um cliente possui as informações de data de nascimento, primeiro nome e segundo nome.
    - Imagine que você tem o método “RecuperarIdade”. A idade é um atributo armazenado ou é um cálculo a partir da data de nascimento?
    - E para um método RecuperarNomeCompleto, como ele poderia ser realizado.
    - Conclusão:
      - A preocupação está em obter ou manipular a informação e não como ela está codificada internamente dentro da classe (ou da cápsula de encapsulamento)!

Modificadores de Acesso

- **Como implementar o encapsulamento?**
- **Modificadores de acesso**
  - Você perceberá que atributos, métodos e mesmo classes contém associados palavras **public** ou **private (**em herança veremos também **protected)**
  - Estes são modificadores de acesso, indicam para quais partes do programa um determinado atributo ou método pode ser acessado
  - Por exemplo, um método público (public) permite acesso pelas várias partes do programa

Modificadores de Acesso

- **Public**
  - Permite acesso a qualquer parte do sistema
- **Private**
  - Os atributos e métodos podem ser acessados apenas na classe em que são definidos
- **Protected**
  - Os atributos e métodos podem ser acessados na classe em que são definidos e nas classes que herdam desta classe, isto é, nas classes filhas
  - Por exemplo, imagine que existe uma classe ContaCorrente baseada em Conta, no código da classe conta conrrente seria possível acessar a variável saldo na classe conta

- Uso em outras partes do projeto
  - Estes modificadores podem ser associados a classes, isso porque no projeto existe o conceito de pacotes, conjunto de classes, e pode-se isolar uma classe apenas em seu determinado pacote

Exercício

- No exemplo anterior, mude o modificador de acesso de Saldo para private e verifique se o código funciona normalmente.

```
Conta c1 = new Conta ();
c1.Numero = 125;c1.Saldo = 0;
c1.Depositar(150.00);
c1.Depositar(250.00);
c1.Sacar(100.00);
c1.Saldo = 3000,00;
c1.Sacar(600.00);
c1.Saldo = 1800.00;
c1.Depositar(100.00);
System.out.print(“ Saldo : “ +
c1.Saldo());
```

Encapsulamento

- Getters e Setters
- Codificando exemplos
- Usando no corpo principal

Encapsulamento

- No exemplo anterior, encapsulamos o atributo Saldo e sabendo-se que ele começou com valor 0.0, é normal considerarmos que o valor deste atributo será gerenciado apenas por métodos Depositar e Sacar.

- A solução foi muito interessante para o problema específico de Saldo mas é comum realizarmos o encapsulamento de todos os atributos,

- Em nosso exemplo, colocaríamos todos os atributos como private.

```
private int Agencia;
private int Conta;
private String NomeTitular;
private float Saldo;
```

- A questão que fica é?
- Se não é possível acessar os atributos diretamente, como então alterar ou recuperar valores?

Encapsulamento

- Para implementar um encapsulamento, além de determinar os atributos como “private” são criados os métodos para operar seus valores

- Métodos get e set (ou getters e setters)
  - Ou seja, o acesso aos atributos ocorre por métodos
  - No caso, surge o papel dos métodos denominados get e set,
  - O método get recupera o valor do atributo
  - E por sua vez o método set altera o valor do atributopara recuperar e para modificar valores de um objeto

```
private int Agencia;
private int Conta;
private String NomeTitular;
private float Saldo;
```

```
public void setNome(String pNome)
{
    NomeTitular = pNome;
}

public String getNome()
{
    return NomeTitular;
}
```

```
public static void main(String[] args) {
    Conta c;

    c = new Conta(7632, 232, "Teobaldo Bisâncio");

    c.setNome("Teobaldo Bisâncio Neto");

    System.out.print(c.getNome());

}
```

Método getNome()

- Este método, por precisar de recuperar a informação para “alguém” de fora da classe, ele retorna o valor do atributo e por sua vez não precisa de passagem de parâmetro

```
public void setNome(String pNome)
{
    NomeTitular = pNome;
}

public String getNome()
{
    return NomeTitular;
}
```

```
public static void main(String[] args) {
    Conta c;

    c = new Conta(7632, 232, "Teobaldo Bisâncio");

    c.setNome("Teobaldo Bisâncio Neto");

    System.out.print(c.getNome());

}
```

Método setNome()

- Para um setter, ele precisa recuperar um valor de fora da classe e "acomodar" no atributo dentro da classe
- Neste caso ele recebe um parâmetro e não precisa retornar valor

```
public void setNome(String pNome)
{
    NomeTitular = pNome;
}

public String getNome()
{
    return NomeTitular;
}
```

```
public static void main(String[] args) {
    Conta c;

    c = new Conta(7632, 232, "Teobaldo Bisâncio");

    c.setNome("Teobaldo Bisâncio Neto");

    System.out.print(c.getNome());

}
```

Exercício

- Exercício 5
  - Discuta como seria o encapsulamento na classe **Conta**

- Exercício 6
  - Acrescente os modificadores de acesso aos atributos
  - Implemente os métodos “get” e “set” para cada classe
    - Dica: O ambiente pode gerar estes métodos na classe
  - Explicite o modificador de acesso public para os métodos **Sacar e Depositar**

Encapsulamento

- Pontos chaves do exercício
  - Classe Conta
  -

```
public class Conta {

    private int     Agencia;
    private int     NumeroConta;
    private String  NomeTitular;
    private double  Saldo = 0;

    public int getAgencia() {
        return Agencia;
    }

    public int getNumeroConta() {
        return NumeroConta;
    }

    public String getNomeTitular() {
        return NomeTitular;
    }

    public double getSaldo() {
        return Saldo;
    }

    public void setAgencia(int pAgencia) {
        this.Agencia = pAgencia;
    }
```

  - Versão no projeto: v3_Encapsulamento

Classe Banco

- Agora para acessar os atributos é necessário usar os get's e set's

```
System.out.printf(" Entre com a agência da conta         : ");
// c1.Agencia = s.nextInt() //Antes,acesso direto sem "setAgencia"
c1.setAgencia(s.nextInt());
```

```
double v = c1.getSaldo();
```

Conclusões

- Retomando Plano de Aula
- Revisão
- Para saber mais

Plano de Aula

- ***Objetivos***
  - Compreender o que é encapsulamento
  - Exercitar modificadores de acesso como meio para implementar o encapsulamento

- ***Tópicos***
  - Contexto da disciplina
  - Encapsulamento: Refletindo sobre um problema
  - A classe como uma “cápsula”
  - Modificadores de acesso
  - Getters e setters
  - Exercitando com Conta
  - Exercício de Sala

Sugestão de estudos

- **Reforço com algumas pesquisas**
  - Pesquisa sobre Projeto e Arquivos no NetBeans.
- **Faça uma leitura de um livro ou de um bom site da Internet para ampliar sua compreensão**

  Apostila do Alura Livro do Deitel w3schools.com
- **Se quiser assuntos independentes mas curiosos, veja algum vídeo sobre cenários de tecnologia e discuta quais Classes e Objetos estão relacionados**
  - **Agricultura 4.0**
  - **Internet das Coisas**
  - **Inteligência Artificial**

- **Após estas "viagens" e explorações, faça as listas de exercícios mais diretas, práticas para explorar os assuntos do dia a dia da disciplina.**

Departamento de Engenharias e Computação
Colegiado de Ciência da Computação
Disciplina: Linguagem de Programação III

Cap 2 – Conceitos Básicos de Orientação a Objetos

Parte 2 – Encapsulamento e Modificadores de Acesso

Professor: Otacílio José Pereira