Encapsulamento e modificadores de acesso

Apresentação

O encapsulamento é um mecanismo de programação que vincula o código e os dados que ele trata. E isso mantém os dois seguros contra a interferência e a má utilização externa. Em uma linguagem orientada a objetos, o código e os dados podem ser vinculados de tal forma que uma caixa preta autônoma seja criada.

Dentro da caixa, estão todos os códigos e os dados necessários. Quando o código e os dados são vinculados dessa forma, um objeto é criado. Em outras palavras, um objeto é o dispositivo que dá suporte ao encapsulamento.

Nesta Unidade de Aprendizagem, você vai conhecer um pouco mais sobre o processo de encapsulamento e a função dos modificadores em um código criado com base na linguagem Java.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Definir encapsulamento.
- Reconhecer modificadores e métodos de acesso.
- Construir uma aplicação utilizando técnicas de encapsulamento.

Desafio

Trabalhar com encapsulamento em orientação a objetos pode nos trazer muitas vantagens, como diminuir a complexidade no desenvolvimento da aplicação. Entretanto, é importante termos um bom conhecimento sobre o assunto para aplicarmos da melhor forma em nossos projetos.

Você trabalha como analista/programador em uma fábrica de *software* e foi encarregado de criar uma aplicação que calcule o volume de concreto para o trabalho com vigas em construção civil. Sabe-se que a fórmula para o cálculo desse volume é (base x altura x comprimento).

O cliente necessita de uma aplicação que solicite, via caixa de diálogo, os dados referentes à base, à altura e ao comprimento das vigas a serem trabalhadas e deve ser informado o resultado por meio de uma caixa de mensagem.

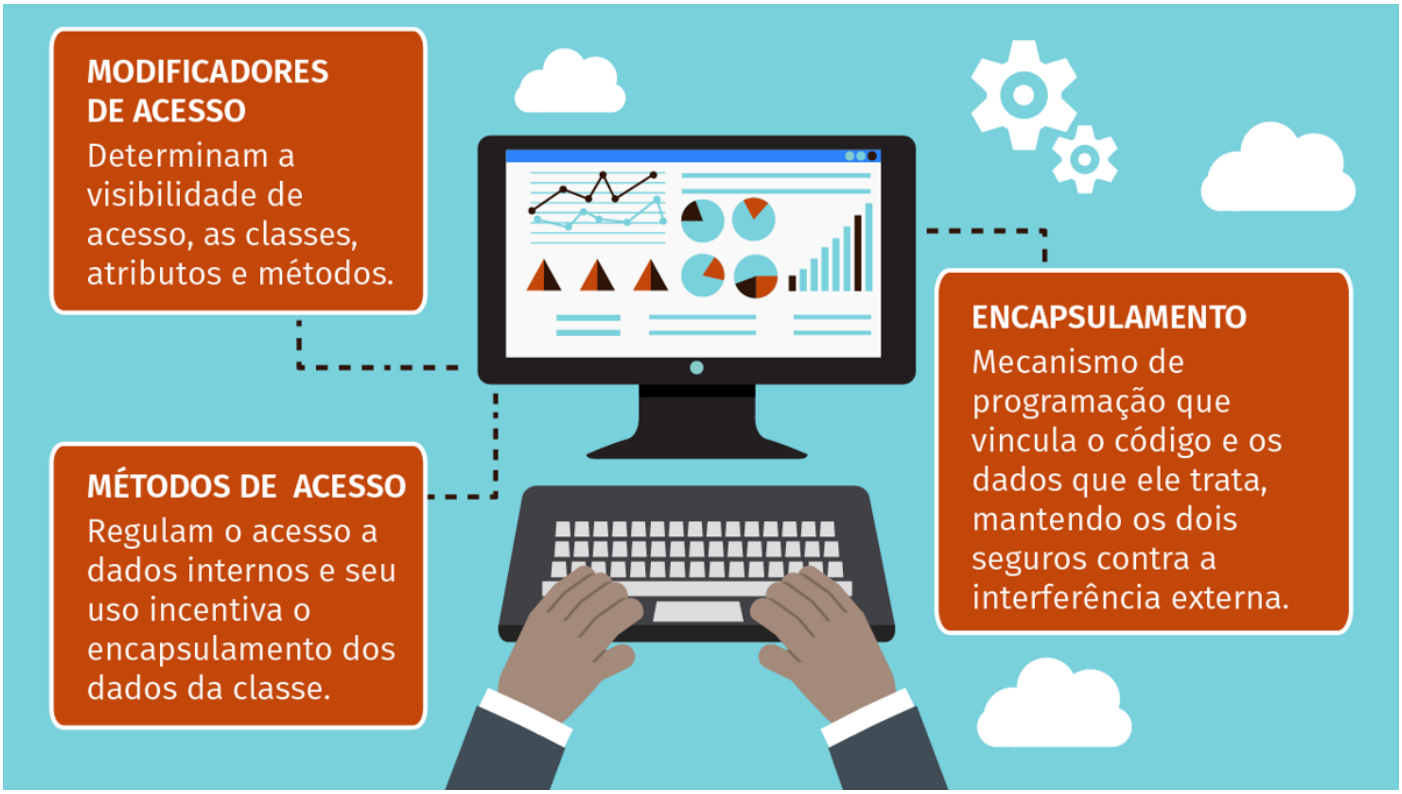
Sua tarefa é implementar essa aplicação em uma linguagem de programação orientada a objetos (POO) e fazer uso das técnicas de encapsulamento.

Infográfico

Existem diferentes recursos que envolvem a POO. Os modificadores de acesso determinam a visibilidade do acesso, as classes, os atributos e os métodos.

O encapsulamento vincula o código aos dados que ele trata e os métodos de acesso regulam o acesso aos dados internos. Todos os mecanismos estão vinculados entre si.

Veja, neste Infográfico, os conceitos de encapsulamento, modificadores e métodos de acesso.



Conteúdo do Livro

Modificadores de acesso possuem um papel importante na orientação a objetos. Portanto, o domínio destes conceitos torna-se vital para o desenvolvimento de softwares.

No capítulo **Encapsulamento e Modificadores de Acesso**, base teórica desta Unidade de Aprendizagem, você verá sobre o mecanismo de encapsulamento e características dos modificadores de acesso.

Boa leitura.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Priscila Gonçalves

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Encapsulamento e modificadores de acesso

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir encapsulamento.
- Reconhecer os modificadores e os métodos de acesso.
- Construir uma aplicação utilizando técnicas de encapsulamento.

Introdução

Neste capítulo, você estudará o encapsulamento e os modificadores de acesso. Além disso, aprenderá como utilizar os modificadores de acessos e quais as diferenças entre eles, tonando-se apto para desenvolver programas utilizando este aprendizado.

Encapsulamento

Encapsulamento em programação orientada a objetos (POO) significa separar o programa em partes, deixando-o mais isolado possível. Dessa forma, é possível torná-lo mais flexível, fácil de modificar e manter, bem como implementar novas funcionalidades.

Trata-se de uma forma muito eficiente de proteger dados que são manipulados dentro da classe, determinando onde ela poderá ser manipulada. Geralmente, utiliza-se o acesso mais restrito (*private*) para que não ocorra acesso público aos membros. O encapsulamento ocorre em dois níveis, conforme você pode ver a seguir.

- Nível de classe: em que se determina o acesso de uma classe inteira, podendo ser *public* ou *package-private*.
- Nível de membro: em que se determina o acesso de atributos ou métodos da classe, podendo ser *public*, *private*, *protected*, *package-private* e *default*.

Para termos métodos encapsulados, fazemos uso de modificadores de acesso, ou seja, para acessar atributo ou método que faça parte do encapsulamento, deve-se utilizar get e set, em que set significa que algum atributo deve ter certo valor; e get é utilizado para recuperar o valor desse atributo.

Na Figura 1, você verá um exemplo de encapsulamento para que possa compreender melhor. Conforme Moreira Neto (2004), o exemplo a seguir apresenta uma classe em que existem dados privativos e construtores e dois métodos (débito e crédito) que podem alterar esses campos. Dessa forma, consegue-se garantir que todas as contas sejam sempre válidas.

Fique atento

Se o método puder ser usado por outras classes (que não sejam subclasses), use *public*; caso contrário, se você quiser que suas subclasses possam alterar algum comportamento, encapsule o comportamento em um método *protected*. Já se o método trata de um detalhe de implementação e você não quer que ninguém modifique (nem mesmo as subclasses), use *private*.

```
public class Conta {
   private int numero;
   private double saldo;
   private double juros;

  //metodos de acesso
        public double getJuros(){
        return juros;
        }

        public int getNumero(){
        return numero;
        }

        public double getSaldo(){
        return saldo;
        }

        public void setJuros(double juros) {
        this.juros = juros;
        }

        public void setNumero(int numero) {
        this.numero = numero;
        }

  //metodos
        public void debito(double valor){
        this.saldo -= valor;
        }

        public void credito(double valor){
            this.saldo += valor;
        }

}
```

Figura 1. Código encapsulamento.
Fonte: Neto (2004, p. 76).

Reconhecimento de modificadores e métodos de acesso

Em POO modificador de acesso é a palavra que define um atributo, método ou classe e pode ser público, privado ou protegido. Entre os três modificadores existem quatro níveis de visibilidade: *private*, *default*, *protected* e *public*. Público (*public*) significa que qualquer classe pode ter acesso; privado (*private*), que somente têm acesso métodos da própria classe, podendo manipular o atributo; protegido (*protected*) pode ser acessado somente pela própria classe ou subclasses; e *default* tem acesso as classes que estiverem no mesmo pacote que a classe que possui o atributo.

Em geral, os modificadores de acesso são utilizados para privar os atributos de serem acessados diretamente, e implementam métodos públicos que possam acessar e alterar os atributos. Esse processo é chamado de encapsulamento.

Métodos privados são utilizados por métodos públicos da mesma classe, para que o código seja reutilizado em mais de um método. Conforme descrito anteriormente, o *private* proíbe o acesso externo, mas também bloqueia a leitura do atributo. Assim, para que possamos recuperar seu valor, precisaremos utilizar um método público na própria classe que devolverá o valor do atributo.

Construção de aplicação utilizando técnicas de encapsulamento

Em POO o mecanismo que faz o vínculo entre o código e os dados que são tratados é chamado de encapsulamento, forma na qual o código é mantido de maneira segura. Utilizando essa técnica, é possível diminuir a complexidade do desenvolvimento da aplicação. É fundamental a utilização de modificadores de acesso para que os códigos possam ser mantidos e novas implementações sejam criadas, sem que ocorram problemas em níveis de acessos. Esses modificadores têm a importante missão de controlar o acesso a membros de classes. A seguir, você verá um exemplo de encapsulamento:

```
class Aluno{
      private String codMatricula;
      private String nome;
      private String cpf;
Aluno ( tMat: String, tNome: String, tcpf: String){
      codMatricula = tMat;
      nome = tNome;
      cpf = tcpf;
}
public String codMatricula(){
      return codMatricula;
}
public String nome(){
      return nome;
}
public String cpf(){
      return cpf;
}
}
```

Link

Acesse o link ou o código a seguir para assistir a um vídeo sobre encapsulamento.

https://qrgo.page.link/1SSvz

Exemplo

O exemplo da tabela a seguir apresenta um tutorial para você entender melhor a questão dos modificadores de acesso.

Modifier	*Class*	*Package*	*Subclass*	*World*
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
No modifier	✓	✓	✗	✗
Private	✓	✗	✗	✗

Outro exemplo prático pode ser observado a seguir na implementação de uma classe *Animal*.

Na classe *Gato*, você poderá chamar todos os métodos de *Animal* declarados como *public* ou *protected* e, se as classes estiverem no mesmo pacote, os métodos *default*. Dessa forma, *Gato* não chama os métodos *private* de Animal. O mesmo raciocínio se aplica aos atributos.

Você também pode sobrescrever (*override*), em *Gato*, os métodos *public* e *protected* de *Animal* e, se estiver no mesmo pacote, também os métodos *default*.

Note que você pode ter em *Gato* um método com a mesma assinatura de um método *private* de *Animal*, mas nesse caso trata-se de um método novo e não uma nova versão do método de *Animal*.

```
class Animal {
  protected void metodoProtected() {
    System.out.println("animal protected");
  }
  private void metodoPrivate() {
    System.out.println("animal private");
  }
}

class Gato extends Animal {
  protected void metodoProtected() {
    System.out.println("gato protected");
  }
  private void metodoPrivate() {
    System.out.println("gato private");
  }
}
```

Como o método *private* de *Animal* não foi sobrescrito por *Gato*, é ele que será chamado quando houver uma referência do tipo *Animal*.

Referências

HORSTMANN, C. *Conceitos de computação com o essencial de C++*. 3. ed. Porto Alegre: Bookman, 2005.

NETO, O. M. *Entendendo e dominando o Java*. São Paulo: Digerati, 2004.

Leituras recomendadas

BARBOSA, M. A. L. *Tipos abstratos de dados e construções encapsuladas*: linguagens de programação. 2012. Disponível em: <http://malbarbo.pro.br/arquivos/2012/1028/11-tipos-abstratos-de-dados-e-construcoes-encapsuladas.pdf>. Acesso em: 17 maio 2018.

HAILTON. *Um pouco sobre polimorfismo Java.* 2009. Disponível em: <https://www.devmedia.com.br/encapsulamento-polimorfismo-heranca-em-java/12991>. Acesso em: 17 maio 2018.

QUAL É A DIFERENÇA entre modificadores public, default, protected e private? 2015. Disponível em: <https://pt.stackoverflow.com/questions/23/qual-%C3%A9-a-diferen%C3%A7a-entre-modificadores-public-default-protected-e-private>. Acesso em: 17 maio 2018.

Fique atento

Os *links* para *sites* da Web fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

O conceito de encapsulamento define algumas ações a serem adotadas em relação ao conteúdo de um código e à aplicação como um todo. O encapsulamento garante que sejam executadas as ações de empacotar, esconder, deixar visível apenas aquilo que interessa ao usuário.

Nesta Dica do Professor, você vai analisar um exemplo prático de encapsulamento e acompanhar a aplicação da técnica com base no conhecimento adquirido nesta Unidade de Aprendizagem.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **O controle de acesso aos atributos e aos métodos de uma classe é muito importante na POO, independentemente da linguagem de programação utilizada. E o encapsulamento é o processo que define diferentes níveis de restrições a serem atribuídos ao código-fonte.**

Em relação ao encapsulamento, assinale a alternativa correta.

A) Encapsular é o ato de criar classes dentro de outras classes sem se preocupar com a organização lógica dos dados contidos no código.

B) Encapsulamentos são colocados em bibliotecas e disponibilizados para reúso em programas além daqueles para os quais eles foram escritos.

C) Em grandes projetos de todas as dimensões, é essencial que apenas um desenvolvedor seja alocado para o projeto. Por isso, apenas uma unidade lógica de programa geralmente pode ser executada.

D) Encapsular é um processo de esconder a complexidade do código em questão e fornecer ao usuário apenas o que lhe convém para a sua atividade.

E) Existe outro tipo de encapsulamento necessário para construir grandes programas: o encapsulamento de nomeação.

2) **Os atributos são as propriedades de um objeto e também são conhecidos como variáveis ou campos. O objetivo é definir o estado de um projeto, permitindo que os valores sofram as devidas alterações.**

Para encapsular um atributo, deixando-o visível apenas para a classe que o contém, qual palavra-chave deve ser utilizada?

A) Public.

B) Static.

C) Void.

D) Não é necessário informar modificador de acesso.

E) Private.

3) **Em Java, existe uma base de encapsulação, em que o comportamento dos dados e o código são definidos, bem como a forma como serão partilhados com um conjunto de objetos. Todas as ações relacionadas com a linguagem de programação Java são bem estruturadas.**

Assinale a alternativa que apresenta a unidade básica de encapsulamento em Java.

A) Pacote.

B) Modificadores de acesso.

C) Classe.

D) Método.

E) Interface pública da classe.

4) **Os elementos básicos de uma classe são chamados de membros e categorizados como variáveis que especificam o estado da classe ou objetos de instância e métodos que especificam o mecanismo por meio do qual uma classe ou objetos podem operar.**

Existem diferentes modificadores para declaração de membros sobre o modificador acessível em subclasses da classe, em subclasses do mesmo pacote e na própria classe. Com relação a isso, assinale a alternativa correta.

A) Protected.

B) Private.

C) Public.

D) Package.

E) Static.

5) Analise o seguinte código em Java:

```
public class Aluno
{
private int matricula;
private String e_mail;
```

```
public int getMatricula(){
return matricula;
}
public void setMatricula(int mat){
this.matricula = mat;
}
public String getEmail(){
return e_mail;
}
public void setEmail(String email){
this.e_mail= email;
}
}
```

Sobre este código, assinale a alternativa correta.

A) Todos os atributos dessa classe poderão ser acessados apenas por seus respectivos métodos.

B) Os métodos cujo nome é precedido da palavra set são métodos de retorno.

C) Toda classe que tem atributos privados não obriga que cada atributo tenha o método que fará acesso a ele.

D) É recomendado que todos os atributos da classe Aluno sejam implementados com o modificador de acesso private para estarem acessíveis a todas as classes.

E) Ao instanciarmos essa classe por meio de uma classe de controle, quando chamarmos os membros por meio da variável de referência, todos os atributos estarão visíveis.

Na prática

Em programação, houve um tempo em que os códigos ficaram muito extensos e, com isso, os programadores tinham muita dificuldade na organização. Portanto, a solução foi dividir em componentes e os usuários não têm a necessidade de conhecer a complexidade do código dos componentes, mas apenas informar as entradas corretas para receber resultados.

Essa técnica é chamada de "encapsular o código". Além da organização, trouxe mais segurança, pois outros programadores usam o componente sem fazer qualquer alteração.

Veja, Na Prática, um exemplo de encapsulamento.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Classes, objetos, pacotes, modificadores de acesso (*public* e *package-friendly*) e encapsulamento

O vídeo demonstra a criação de uma classe, do pacote e os conceitos de reúso de código, demonstrando como os modificadores podem ser acessados para diferentes recursos em seus modos *public* e *package-friendly*. Conceitos importantes de encapsulamento também são abordados.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Métodos, atributos e classes no Java

Veja, neste artigo, quais são e como utilizar os modificadores de acesso a métodos, atributos e classes da linguagem Java.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conceitos de computação com Java – Compatível com Java 5 & 6

Acesse esta obra para aprofundar seus conhecimentos sobre os conceitos e as principais práticas de programação. O livro contém, entre outras coisas, dicas úteis sobre as boas práticas da engenharia de software.

Conteúdo interativo disponível na plataforma de ensino!

Conceitos de linguagens de programação

Veja, nesta obra, uma apresentação sobre as construções fundamentais da linguagem de programação e tenha as ferramentas necessárias para uma avaliação crítica de linguagens existentes e futuras.

Conteúdo interativo disponível na plataforma de ensino!