Encapsulamento

Tiago Eugenio de Melo

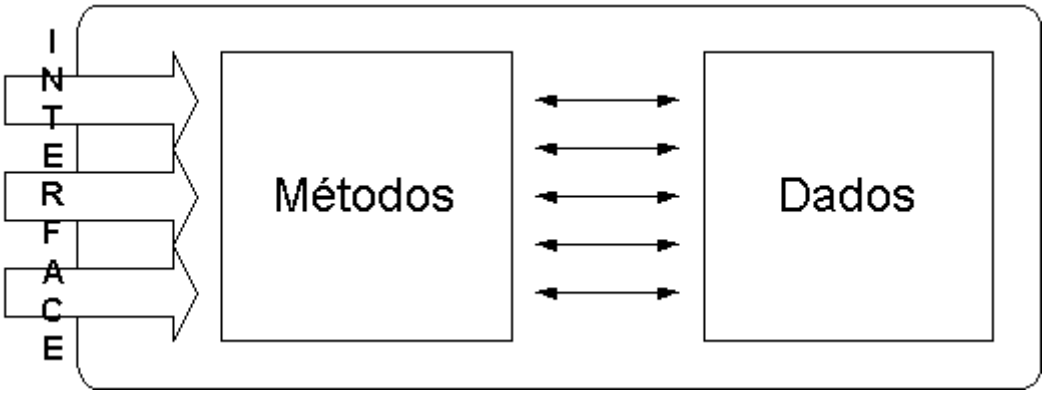
tiago@comunidadesol.org

Encapsulamento

- Definição:
  - Mecanismo que coloca juntos o código (métodos) e os dados (atributos), mantendo-os controlados em relação ao seu nível de acesso.
  - O conceito de encapsulamento está intimamente ligado ao conceito de ocultamento da informação (*information hiding*).

Encapsulamento

- Um objeto encapsula estado (dados) e métodos (código) que podem acessar dados.
- Ilustração do conceito de objeto:



Encapsulamento

- Objetivo:
  - Controlar o acesso de atributos e métodos de um objeto, através de uma interface bem definida.
- Benefícios:
  - Manutenção de software;
  - Evolução de software;

Encapsulamento

- Exemplo:
  - Motor de um automóvel.
  - O motorista não precisa ter conhecimento técnico de como funciona cada parte do motor, mas apenas saber qual é a sua finalidade e como usá-lo.

Encapsulamento

- Vantagens:
  - Proteger os atributos do objeto quanto à manipulação por outros objetos (proteção contra acesso não-autorizado, valores inconsistentes, entre outras possibilidades).
  - Esconder a estrutura interna do objeto de modo que a interação com este objeto seja relativamente simples e, à medida do possível, siga um padrão de desenvolvimento que facilite o entendimento dos programadores que o utilizem.

Abstração de dados e encapsulamento

- As classes, normalmente, ocultam os detalhes de implementação dos seus usuários. Isso se chama **ocultamento de informações**.
- Exemplo:
  - O motorista de um veículo ao fazer uso do motor do carro está usando o motor para se locomover, porém não precisa saber dos seus detalhes de funcionamento.

Abstração de dados e encapsulamento

- Nesse exemplo, o cliente se preocupa com a funcionalidade que o motor oferece, mas não como essa funcionalidade é implementada.
- Esse conceito é conhecido como **abstração de dados**.

Abstração de dados e encapsulamento

- A Programação Orientada a Objetos (POO) tem como principais atividades a criação de tipos e a expressão de interações entre objetos desses tipos.

Abstração de dados e encapsulamento

- Essa atividade está diretamente associada à noção de **tipo abstrato de dados** (ADT - *abstract data type*), que melhora o processo de desenvolvimento de programas, pois permite mais flexibilidade ao programador na criação de novos tipos de dados.

Abstração de dados e encapsulamento

- Assim, pode-se afirmar que um ADT captura duas noções: representação de dados e operações que podem ser realizadas nesses dados.
- Programadores Java utilizam classes para implementar tipos abstratos de dados.

Encapsulamento em Java

- O encapsulamento em Java ocorre nas classes.
- Quando o programador cria uma classe, ele especifica o código e os dados que irão formar essa classe.
- Estes elementos serão chamados de membros da classe.

Encapsulamento em Java

- O **comportamento** e a **interface** de uma classe são definidos a pelos **métodos** que operam nas instâncias de dados.
- O encapsulamento em Java é implementado através dos seus modificadores de acesso público, protegido, privado e implícito.

Encapsulamento em Java

- Considerando que o objetivo de uma classe é encapsular a complexidade, existem mecanismos para ocultar a complexidade da implementação que está dentro da classe.
- Cada método ou variável em uma classe pode ser definida como pública, privada ou protegida.

Encapsulamento em Java

- A interface de uma classe possibilita que todos os usuários externos possam acessar livremente os dados da classe que os métodos públicos permitem.
- Já os métodos privados estabelecem que os dados somente podem ser acessados pelos métodos que são membros da classe.

Encapsulamento em Java

- Considerando que os membros privados de uma classe só podem ser acessados por outras partes do programa através dos métodos públicos desta classe, o programador em Java pode fazer uso do encapsulamento para garantir que ações inapropriadas ou imprevistas não ocorram.

Encapsulamento em Java

- Assim, o programador em Java deve ser bastante cuidadoso ao definir a interface pública de uma classe para não expor demasiadamente o funcionamento da classe.

Encapsulamento em Java

- Encapsulamento em classes

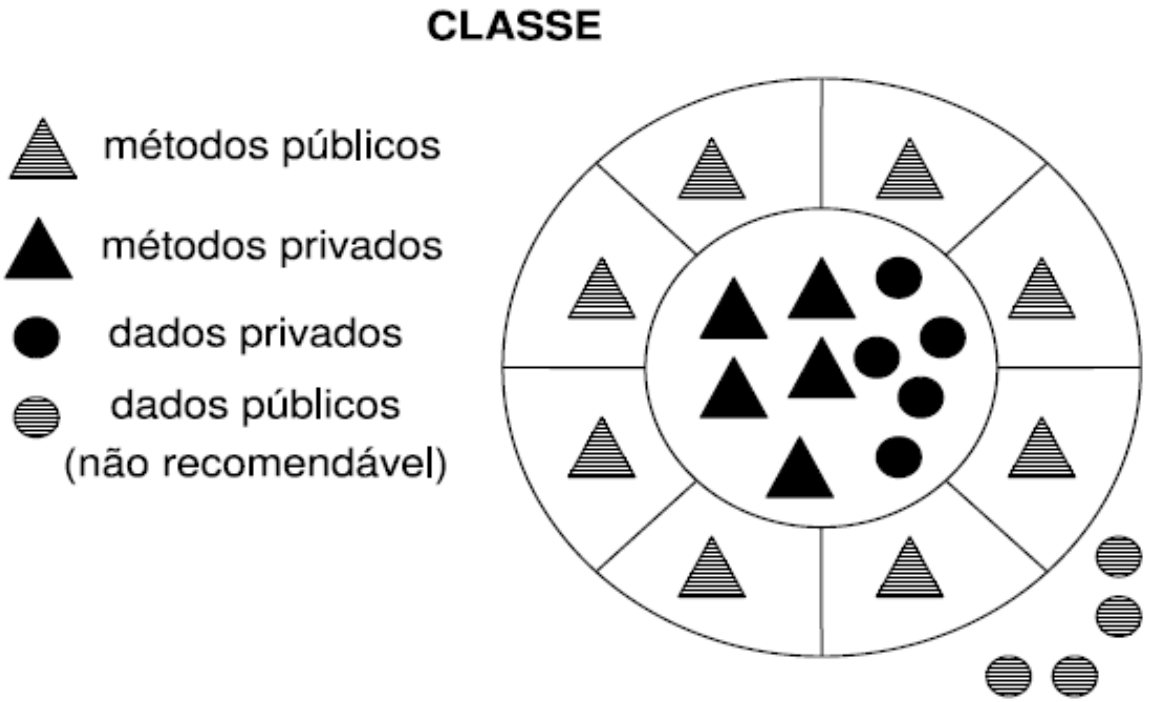


Figura 3.1: Encapsulamento: os métodos públicos podem ser empregados para proteger os dados privados.

Encapsulamento em Java

- O modificador `private` é o mais restritivo e não foi criado para classes, mas apenas para membros de classes.
- Apesar disso, é possível empregar o modificador `private` nas classes.

Encapsulamento em Java

- A dúvida comum que surge é: como uma classe pode acessar uma classe privada?
- A solução é declarar a classe privada como sendo interna.
- Exemplo:

Programa 3.1 Única maneira de usar classes privadas.

```
public class Desenho {
    private class FerramentaDesenho {

    }
}
```

Classes que encapsulam valores primitivos

- Os tipos primitivos em Java são oriundos de classes que possibilitam a representação de valores nativos como classes, o que é particularmente útil para uso em métodos que esperam um argumento que seja um herdeiro da classe Object.

Classes que encapsulam valores primitivos

- Todas as classes que correspondem aos tipos primitivos de Java fazem parte do pacote `java.lang` e, por isso, não é necessário nenhum comando `import` para utilizá-las.

Modificadores de acesso em Java

- O encapsulamento relaciona os dados (atributos) com o código (métodos) que os manipula.
- O encapsulamento também fornece outro recurso importante que é o controle de acesso.
- Através dos modificadores de acesso, os programadores podem controlar o acesso aos membros de uma classe.

Modificadores de acesso em Java

- É através desse controle que o programador garante que não haverá um uso indesejado dos dados de uma determinada classe.
- Normalmente, uma classe é criada como uma espécie de caixa preta, que pode ser usada, porém, somente através dos seus métodos públicos que foram colocados à disposição.

Modificadores de acesso em Java

- O modificador de acesso é uma instrução que define como um membro de uma classe poderá ser acessado.
- Java possui um rico conjunto destes modificadores.
- Alguns aspectos do controle de acesso estão relacionados à herança e ao conceito de pacotes.

Modificadores de acesso em Java

- Java possui os seguintes modificadores de acesso: `public`, `private` e `protected`.
- Java também define um nível de acesso padrão (*default*) e que se aplica somente quando há o uso de herança.
- O modo de acesso *default* também é conhecido como pacote (`package`).

Modificadores de acesso em Java

- Dica

> Um membro em Java pode ter no máximo um modificador de acesso.

Modificadores de acesso em Java

- Modificador de acesso *public*
  - Este modificador permite que o membro público seja acessado por qualquer outro código do programa.
  - O modificador de acesso *public* é o mais liberal e que, portanto, exige maior responsabilidade do programador ao empregá-lo.

Modificadores de acesso em Java

- Modificador de acesso *private*
  - Este modificador determina que o membro privado só pode ser acessado por métodos de dentro da própria classe.
  - O modificador de acesso *private* é o mais restritivo e que deve ser empregado sempre que possível.

Modificadores de acesso em Java

- Modificador de acesso protected
  - Somente os atributos e métodos podem ser declarados como *protected*.
  - Um membro protegido de uma classe está disponível a todas as classes do mesmo pacote, exatamente como um recurso padrão.
  - Além do mais, um recurso protegido de uma classe está disponível a todas as subclasses da classe que possui o recurso protegido.

Modificadores de acesso em Java

- Modificador de acesso padrão (*default*)
  - Quando não é declarado o tipo de moderador, Java adota como o padrão (*default*).
  - Não existe a palavra-chave *default* em Java.
  - Os recursos *default* de uma classe são acessíveis a qualquer classe no mesmo pacote que a classe em questão.

Modificadores de acesso em Java

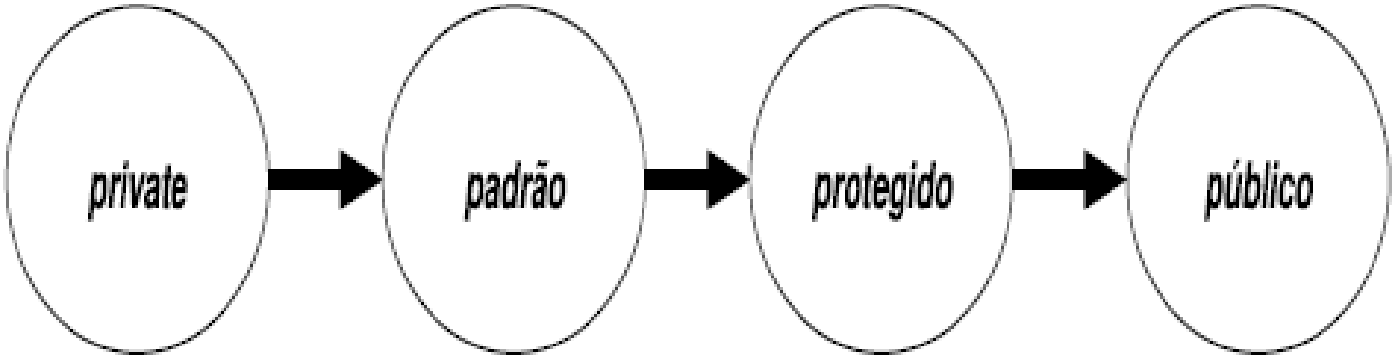
- O que caracteriza um pacote em Java?
  - Quando um programador escreve um aplicativo que envolve várias classes diferentes, é possível que mantenha todos os seus códigos (arquivos .java) e todos os seus arquivos binários (arquivos .class) em um único diretório de trabalho.
  - Ao executar o código, o programador o faz a partir daquele diretório.
  - O ambiente de execução Java considera que todos os arquivos de classe no diretório atual de trabalho constituem um pacote.

Modificadores de acesso em Java

- O que pode acontecer quando um programador Java coloca várias classes num mesmo diretório sem se preocupar com o encapsulamento?
  - Significa que as classes deste diretório são acessíveis a quaisquer classes deste diretório, pois como se fossem do mesmo pacote.
  - Isto pode levar a resultados indesejados.

Modificadores de acesso em Java

- Ordem de restrição dos modificadores de acesso (mais restrito para mais liberal):



Sintaxe dos moderadores de acesso

- Resumo:

Especificador	**Nível**	**Indica que o campo ou método:**
public	público	Pode ser usado livremente pelas instâncias da classe.
protected	protegido	Só pode ser usado na implementação de subclasses.
	pacote	Só pode ser usado por instâncias dentro do mesmo pacote
private	privado	Não pode ser usado fora da implementação da própria classe.

Efeito dos moderadores em Java

Programa 3.3 Exemplo de programa Java que faz uso dos moderadores de acesso *public* e *private*.

```
01.  /* Este programa demonstra a diferença no uso dos moderadores */
02.  class Acesso {
03.      int a; // acesso padrão
04.
05.      public int b; // acesso público
06.
07.      private int c; // acesso privado
08.
09.      // métodos para acessar o atributo c
10.      void setC(int i) { // atribui valor para c
11.          c = i;
12.       }
13.
14.      int getC() { // recebe o valor de c
15.          return c;
16.      }
17.   }
18.
19.  public class TesteAcesso {
20.      public static void main(String args[]) {
21.          Acesso obj = new Acesso();
22.
23.          obj.a = 10; // o atributo a pode ser acessado diretamente
24.
25           obj.b = 20; // o atributo b pode ser acessado diretamente
26.
27.          // obj.c = 100;     o atributo c não pode ser acessado diretamente
28.
29.          // O atributo c deve ser acessado através dos métodos disponíveis
30.
31.          obj.setC(100); // setc é empregado para atribuir um valor para c
32.
33.          System.out.println("Os valores de a, b, e c são: " + obj.a + " " +
             obj.b + " " + obj.getC());
34.      }
35.  }
```

Uso dos métodos *set* e *get*

- Há necessidade de se ter métodos públicos para que se possa acessar os atributos que, em geral, são privados.
- O padrão adotado, pelos programadores em Java, para estes métodos é setNomeAtributo(.) e getNomeAtributo(.) para modificar e receber os valores dos atributos, respectivamente.

Uso dos métodos set e get

- Então, qual é a razão de se colocar os atributos como privados se existem métodos que podem acessá-los?
- Por que não torná-los logo como públicos?
  - Embora os métodos set() e get() possam fornecer acesso a dados private, o acesso é restrito pela maneira como os m´etodos foram implementados pelo programador. Isso ajuda adesenvolver programas mais seguros e confi´aveis.

Uso dos métodos set e get

- Exemplo:
  - Um atributo minuto de uma classe relógio.
  - Sabe-se que esse atributo pode receber valores no intervalo [0..60].
  - Não faria sentido qualquer valor fora desse intervalo.
  - Neste caso, recomenda-se o uso dos métodos gets como uma forma de garantir a integridade dos dados das classes.

Uso dos métodos set e get

- Dica

Os projetistas de classe não precisam fornecer métodos set() ou get() para cada atributo private.
Essas capacidades devem ser fornecidas somente quando fizerem sentido.

Resumo de acesso em Java

- De forma resumida, os modos de acesso de Java são:
  - *public*: um recurso público que pode ser acessado por qualquer classe.
  - *protected*: um recurso protegido só pode ser acessado por uma subclasse da classe que possui o recurso, ou por um membro do mesmo pacote da classe que possui o recurso.

Resumo de acesso em Java

- De forma resumida, os modos de acesso de Java são:
  - *default* : um recurso padrão só pode ser acessado por uma classe do mesmo pacote que a classe que possui o recurso.
  - *private*: um recurso privado só pode ser acessado pela classe que possui o recurso.

Atividades

- Quais são as restrições impostas pelos comandos *public*, *protected* e *private* em Java?
- O modificador de acesso implícito impõe as mesmas restrições do modificador *protected*? Justifique a sua resposta.
- O método *main* de uma classe deve, obrigatoriamente, usar o modificador *public*? Justifique a sua resposta.

Atividades

- Crie uma classe em Java que: a) contenha os atributos nome, idade e altura; b) encapsule os atributos; c) crie um método *main* que mostre os valores que estão nos atributos.

Atividades

- Crie uma classe Retangulo. A classe tem atributos largura e altura, ambos sendo do tipo *float*. A classe deve ter métodos que calculam o perímetro (perimetro()) e a área (area()) do retângulo. A classe tem métodos set e get para a largura (largura) e a altura (altura). Os métodos set devem verificar se largura e altura são, cada um, números de ponto flutuante maiores que 0,0 e menores que 20,0. Escreva um programa em Java para testar a classe Retangulo.

Atividades

- Considere o Programa 3.4 e responda às seguintes questões:
  - A classe Tempo segue os princípios do encapsulamento? Comente a respeito.
  - Como é possível estender o código para atender aos princípios do encapsulamento? Quais seriam as vantagens que isto traria? Faça as modificações necessárias no código.

Atividades

Programa 3.4 Classe Tempo.

```
class Tempo {
    int hora;
    int minuto;

    Tempo() {
    };

    public static void main(String arg[]) {
        Tempo t = new Tempo();
        t.hora = 3;
        t.minuto = 25;
        System.out.println("A hora agora é " + t.hora + ":" + t.minuto);
    }
}
```