Encapsulamento, método de acesso e modificadores

Programação de Soluções Computacionais

Prof. Dr. Charles Ferreira
charles.ferreira@anhembi.br

Agenda

Encapsulamento

Modificador de acesso

Set e Get

Exercícios

Encapsulamento

Encapsulamento, metodo de acesso e modificadores

Encapsulamento - O que é?

- **Mecanismo que possibilita restringir a visibilidade de:**
  - variáveis;
  - métodos;
  - ou até a própria classe.

Encapsulamento - Para que serve?

- **Objetivo:**
  - detalhes de implementação ficam ocultos ao usuário da classe;
  - usuário utiliza as funcionalidades da classe ...
  - sem precisar entender como a classe funciona internamente.

Encapsulamento — Vantagens

- **Além de ocultar detalhes de implementação:**
  - torna o código mais legível;
  - minimiza erros de programação;
  - restringe o conteúdo das variáveis;
  - facilita a ampliação do código em função de novas atualizações.

Encapsulamento — Vantagens

- **Manuteção e/ou atualização de código:**
  - O desenvolvedor da classe pode realizar manutenção/melhorias no código;
  - usuário da classe não precisará saber que houveram adaptações.
    - desde que não comprometa a usabilidade tradicional da classe

Encapsulamento, metodo de acesso e modificadores

Modificador de acesso

Modificadores de acesso (Qualificador)

- **Controla o acesso aos membros da classe (atributos e métodos)**
  - **padrão: (sem especificar nenhum qualificador)**
    - membros que não foram marcados com nenhum modificador explicitamente;
  - **public:**
    - permite que qualquer outra parte da aplicação tenha acesso ao membro;
    - Só podem ser acessados por outras classes dentro do mesmo pacote
  - **private**
    - só é acessível dentro da própria classe em que foi declarado.
  - **protected:**
    - membros são acessíveis por classes dentro do mesmo pacote e por classes derivadas;
    - mesmo em pacotes diferentes.

Modificadores de acesso (Qualificador)

- **Modificadores mais utilizados:**
  - **private:** ocultar os atributos da classe.
  - **public:** deixar os métodos visíveis.
- A comunicação com os objetos será feita através de atributos e métodos públicos.

Vamos modificar os atributos para private ...

```java
public class Pessoa{

  private String nome;
  private int idade;
  private double altura;

}
```

```java
public class Main{

  public static void main(String[] args){

    Pessoa p = new Pessoa();
    p.nome = "Fulano";

  }
}
```

- **O que acontece ao tentarmos executar esse código?**

```
Main.java:6: error: nome has private access in Pessoa.java
```

Set e Get

Encapsulamento, metodo de acesso e modificadores

Métodos set e get

- **Atributos private impedem o acesso direto aos atributos;**
  - precisaremos de métodos públicos para modificar o valor do atributo.

  - **setAtributo:**
    - método que **atribui um valor** ao atributo.

  - **getAtributo:**
    - método que **resgata o valor** de um atributo.

Método set()

```
public void setNomeAtributo(tipo valor){
  this.atributo = valor;
}
```

- Modificador de acesso do método é **public**;
- Nome do método sempre inicia com **set** seguido do nome do atributo;
- Método sempre **recebe um parâmetro** do tipo do atributo;
- Valor do parâmetro recebido será **armazenado** no atributo;
- Não tem retorno — sempre é **void**.

Método get()

```
public String getNomeAtributo(){
  return atributo;
}
```

- Modificador de acesso do método é **public**;
- Nome do método sempre inicia com **get** seguido do nome do atributo;
- Método **não tem parâmetros**;
- Tipo de retorno do método **é do tipo** do atributo;
- Método sempre **retorna** o valor do atributo.

Exemplo: set e get

- **Vamos utilizar o set e get para acessar os atributos da classe**

Exemplo: utilização set e get

```java
public class Pessoa{

  private String nome;
  private int idade;
  private double altura;

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

  public String getNome(){
    return nome;
  }

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

  public int getIdade(){
    return idade;
  }

  public void setAltura(double altura){
    this.altura = altura;
  }

  public double getAltura(){
    return altura;
  }

}// fim da classe Pessoa
```

Exemplo:set e get

- **Agora criaremos uma classe com main para testar**

Classe com método main

```
1  public class MainPessoa{
2
3    public static void main(String[] args){
4      Pessoa p = new Pessoa();
5
6      p.setNome("Fulano");
7      p.setIdade(25);
8      p.setAltura(1.6);
9
10     System.out.println("acessando os dados");
11     System.out.println(p.getNome());
12     System.out.println(p.getIdade());
13     System.out.println(p.getAltura());
14   }
15 }
```

Pense e Reflita

- Analisando o código anterior ...
- parece que não houve vantagem em colocar set e get
- Poderíamos simplesmente deixar os atributos públicos ...
- afinal, estamos apenas modificando e acessando seus valores ...

Refletindo ...

- **E se tivemos restrições no acesso aos dados?**
- **Exemplo:**
  - nomes de pessoas não podem conter o caracter #
  - idades e alturas não podem ter valores negativos
- **Como poderíamos fazer esse controle de acesso?**

Exemplo: utilização set e get

```java
public class Pessoa {

  private String nome;
  private int idade;
  private double altura;

  public void setNome(String nome) {
    if (nome.contains("#")) {
      System.out.println("nome com #");
    }else {
      this.nome = nome;
    }
  }

  public String getNome() {
    return nome;
  }

  public void setIdade(int idade) {
    if (idade <= 0) {
      System.out.println("Idade menor ou igual a zero");
    }else {
      this.idade = idade;
    }
  }

  public int getIdade() {
    return idade;
  }

  public void setAltura(double altura) {
    if (altura <= 0) {
      System.out.println("altura menor ou igual a zero");
    }else {
      this.altura = altura;
    }
  }

  public double getAltura() {
    return altura;
  }

}// fim da classe Pessoa
```

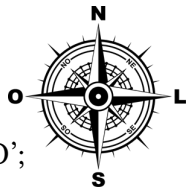
Conclusão

- **Como faríamos a validação dos atributos sem set e get?**
  - Teríamos que colocar a validação na classe com método main
- **Ou seja**
  - iríamos “obrigar” o usuário da classe Pessoa a se preocupar com detalhes de implementação/validação
- **Ao invés disso**
  - colocamos a validação dentro da classe Pessoa
  - facilitando o uso da classe Pessoa

Encapsulamento, metodo de acesso e modificadores

Exercícios

Exercício 1

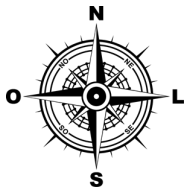


- **Crie a classe Robo**
  - que permite um robô ser navegado pelas direções: ’N’, ’S’, ’L’ ou ’O’;
  - A direção é o único atributo da classe (char);
  - A classe deve verificar se a direção é válida
  - caso não seja, mostrar a mensagem: "Direção inválida".
  - utilize set e get.
- **Na Classe com main:**
  - Faça um programa que pergunte ao usuário qual direção ele deseja mover o robo

Exercício 2

- **Cria uma classe Time**
  - para definir uma hora qualquer representada por três atributos:
    - Hora, minuto e segundo (private)
  - Para alterar a hora atual:
    - faça uma validação dos valores de horas (0 - 23), minutos e segundos (0 - 59)
- **Implemente os seguintes métodos:**
  - visualizarHoraUniversal():
    - retorna uma String com a hora no formato universal de 1 a 24 horas
    - Exemplo: 23:03:06 horas
  - visualizarHoraAmPm():
    - retorna uma String com a hora no formato AM / PM de 1 a 12 horas
    - Exemplo: 11:03:06 PM

Exercício 3



- **Altere a classe Robo**
  - adicione dois novos atributos:
  - posicaoX e posicaoY (ambos private e sem set e get)

- **A cada mudança de direção:**
  - altere a posição X ou Y
  - Exemplo:
    - 'N': `posicaoY++`
    - 'S': `posicaoY--`
    - 'L': `posicaoX++`
    - 'O': `posicaoX--`

- **Adicione um método público**
  - chamado mostrarPosicaoAtual
  - deve imprimir os valores das posições X e Y.

- **Opcional**
  - pergunte se o usuário deseja continuar informando novas direções para o robô

Obrigado

charles.ferreira@anhembi.br