Residência em Software

Residência em Tecnologia da Informação e Comunicação

Classes e Objetos

Professor:

Alvaro Degas Coelho

Java

INSTITUIÇÃO EXECUTORA

cepedi UESC

COORDENADORA

MCTI FUTURO Softex

APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Residência em Software

Encapsulamento

- Regra de Ouro
- Nenhuma classe deve expor nada além do estritamente necessário para cumprir suas responsabilidades.

Residência em Software

Exemplo

- No sistema de controle acadêmico da aula passada
- Olhando o método checarPreRequisitos(Aluno aluno)
  - Classe Disciplina
  - Verifica se aluno já cursou todos os pré-requisitos da disciplina
- Como implementar?

Residência em Software

Exemplo

- Como implementar?
- Uma (possível) solução
- Pegar a lista de disciplinas cursadas pelo aluno
  - lista1 = aluno.getListaDisciplinasCursadas()
- Pegar a lista de disciplinas que são pré-requisitos da disciplina em questão
  - lista2 = this.getListaPreRequisitos()
- Verificar se lista2 está inteiramente contida em lista1
  - If (contem(lista1,lista2)...

Residência em Software

Exemplo

- If (lista1.contem(lista2)…
- Ignoremos (por enquanto) o fato de que existe um método para isso na classe Collections
- A pergunta é: o método contem() deve ser exposto (public)? Porque?

Residência em Software

Encapsulamento

- Atributos devem SEMPRE estar escondidos
  - Private
- Métodos que prestam serviços a outras classes (ou atendem a responsabilidades do próprio sistema) devem esytar expostos
  - Public
- Métodos que não se encaixem no caso anterior devem estar escondidos
  - Private

Residência em Software

Encapsulamento

- Construtores
- Uma expressão muito comum do requisito de encapsulamento
- Quando queremos instanciar um objeto, que características desejamos que ele tenha?
  - Por exemplo: podemos exigir que algum CPF seja estabelecido para cada novo Aluno? Podemos exigir que a Disciplina à qual uma turma está associada seja conhecida desde o momento da instanciação do objeto Turma?

Residência em Software

Um exemplo

- Considere o código abaixo

```
public class Produto {
    private String nome;
    private double preco;
    private int quantidade;

    public Produto(String nome, double preco, int quantidade) {
        setNome(nome);
        setPreco(preco);
        setQuantidade(quantidade);
    }
```

Residência em Software

Um exemplo

- Observe que escolhemos usar setters dentro do construtor. Porque?

```
public class Produto {
    private String nome;
    private double preco;
    private int quantidade;

    public Produto(String nome, double preco, int quantidade) {
        setNome(nome);
        setPreco(preco);
        setQuantidade(quantidade);
    }
```

Residência em Software

Um exemplo

- Mais detalhes do código

```
private void setNome(String nome) {
    if (nome == null || nome.trim().isEmpty()) {
        throw new IllegalArgumentException("Nome do produto não pode ser nulo ou vazio.");
    }
    this.nome = nome;
}
```

Residência em Software

Um exemplo

- Mais detalhes do código

```
public void setPreco(double preco) {
    if (preco < 0) {
        throw new IllegalArgumentException("Preço do produto não pode ser negativo.");
    }
    this.preco = preco;
}
```

Residência em Software

Um exemplo

- Mais detalhes do código

```
public void setQuantidade(int quantidade) {
    if (quantidade < 0) {
        throw new IllegalArgumentException("Quantidade do produto não pode ser negativa.");
    }
    this.quantidade = quantidade;
}
```

Residência em Software

Um exemplo

- O controle de valores inválidos fica nos métodos set
  - Gera exceptions
- Se não usar os getters no construtor?
  - Verifica 2x (uma no construtor, outra nos gets)
- E se mudar a restrição?
  - Trabalho em dobro!

Já mencionamos que Encapsulamento é a Regra de Ouro?

Residência em Software

Exercícios

- Exercício 1: Conta Bancária
- Crie uma classe ContaBancaria com os atributos privados saldo e titular (cpf).
  - Crie um construtor que não permita instanciar uma conta sem titular. E crie a conta com saldo igual a 0.
  - Implemente métodos depositar, sacar e verificar o saldo. Certifique-se de encapsular os métodos de maneira apropriada.
  - Implemente um método transferir (conta1, conta2, valor)
- Assuma que o saldo não pode ser negativo (use exceptions)

Residência em Software

Exercícios

- Exercício 2: Carrinho de Compras
- Crie uma classe CarrinhoDeCompras que represente um carrinho de compras de um cliente em uma loja online.
  - O construtor gera um carrinho vazio mas associado a um cliente
  - Um método fecharCompra() torna aquele carrinho pronto para pagamento. Não pode mais inserir nem excluir itens nele
  - Implemente métodos públicos para adicionar, remover itens (com as respectivas quantidades) e calcular o total da compra.
- Use exceptions
- Assuma que existe uma coleção de Itens (id, descricao, preço)

Residência em Software

Exercícios

- Exercício 3: Sensor de Temperatura
- Desenvolva uma classe SensorTemperatura que encapsule a leitura de temperatura de um dispositivo.
- Implemente métodos públicos para obter a temperatura em diferentes escalas (Celsius, Fahrenheit, etc.)
  - registraTemperatura(valor, escala)
- Proteja o acesso direto aos dados brutos do sensor.