**RESUMO DA AULA 06 - ENCAPSULAMENTO, PACOTES E MODIFICADORES**

ALUNO: Denilson José do Bom Jesus Silva de Lima

**Encapsulamento**

Princípio que diz que as classes devem trabalhar em seus atributos de forma exclusiva, deixando o mínimo de suas funcionalidades visível para as demais classes. Por exemplo:

Um método exclusivo para uma classe, que tem seu uso interno a mesma, deve ser criado em "private", o que o torna uso exclusivo e não visível para outras classes.

O Baixo Acoplamento que traz maior flexibilidade para implementar mudanças, a manutenção e o gerenciamento e a Alta Coesão que está ligada ao "princípio de

responsabilidade única" e diz que uma classe deve ter apenas uma função e não mais do que isso, não devendo assim assumir nada que não lhe diz respeito - são também conceitos importantes a serem levadas em consideração. Isso para evitar características como: rigidez, fragilidade e imobilidade.

Em outras palavras, deve-se desenvolver (*no exercício de escrita da aplicação*) formas de restrição de acesso como as classes, atributos e métodos. Sendo as classes que compartilham algo em comum devendo ser agrupadas em pacotes.

Os níveis de Acoplamento são: conteúdo -> controle - > dados estruturados -> dados -> mensagem.

1. Acoplamento de Conteúdo consiste em uma classe manipular diretamente os dados de outra classe, como consequência, uma mudança nos dados causa mudanças em outras classes.

2. Acoplamento de Controle consiste num módulo que controla a lógica de outro passando sua informação sobre o que fazer.

3. Acoplamento de Dados Estruturados consiste em módulos compartilharem uma estrutura de dados usando apenas uma parte dessa estrutura.

4. Acoplamento de Dados consiste em objetos compartilharem dados (*somente*) através de parâmetros.

5. Acoplamento de Mensagem consiste em ao invés das classes trocarem informações entre si, elas se comunicam de forma comum com uma interface pública, ou seja, o objeto usa uma interface pública para troca de mensagens.

Extra: Sem Acoplamento - os objetos não têm conhecimento um do outro e não se comunicam.

Os níveis de coesão são: coincidente -> lógica -> procedural -> comunicacional -> funcional.

1. Coesão Coincidente consiste no agrupamento aleatório de código com nenhuma relação entre as partes.

2. Coesão Lógica consiste em agrupar códigos que são "logicamente" categorizados (*exemplo: mesma função*).

3. Coesão Procedural consiste em ordenar o código na ordem que as coisas acontecem.

4. Coesão Comunicacional consiste em agrupar o código porque opera sobre um mesmo conjunto de dados.

5. Coesão Funcional consiste em agrupar o código que contribui para uma mesma tarefa definida.

**Pacote**

São as formas de agrupamento, que tem dois tipos principais: Classes e Pacotes.

Classes possuem a função de agrupar as categorias comuns de métodos, atributos, etc e de definir tipos; e Pacotes agrupam definições de classes e interfaces

relacionadas, estruturar sistemas extensos e oferecer um nível mais alto de abstração.

A indicação do Pacote onde está a Classe encontra-se na primeira linha do código (Ex:)

package pacote;

Cada pacote é associado a um diretório de mesmo nome e as classes criadas dentro dele ficam salvas nesse diretório com os arquivos ".class". Sendo o separador de diretórios a barra (/) e o separador de pacotes o ponto (.). Importante saber que "subdiretório" não corresponde a "subpacote", os pacotes permanecem individuais.

(Ex:)

package pacote.pacote1;

Quando se empacota uma classe, o nome do pacote é incorporado ao nome da classe. (Ex:)

package pacote;

public class Classe { (...) } // a classe passa a ser pacote.Classe

A importação de pacotes pode ser Específica (*escrevendo diretamente qual classe do pacote importar*) ou genérica (*com um asterisco "\*" importando todas as classes do pacote e deixando o controle de uso quando alguma classe precisar ser chamada*). É possível (*e recomendado*) estruturar aplicações com pacotes, criando um para cada conjunto de funcionalidade comum.

Uma boa prática é definir o empacotamento antes de começar a desenvolver a aplicação.

**Modificadores de Acesso: public, protected, default e private.**

Servem como o controle para definir quem tem e não tem acesso a funcionalidades do projeto sendo essa restrição relacionada ao local onde se está seja em classes,

pacotes, sub-hierarquias ou no projeto em si.

1. public -> ao ser definida assim, é visível e usável em qualquer lugar, mesmo em pacotes diferentes.

2. protected -> só pode ser utilizada no pacote ou nas subclasses da classe onde é criado.

3. default -> declaração só pode ser utilizada no pacote onde é criado (*não é escrito, a ausência de declaração resulta em declaração de default*).

4. private -> só pode ser utilizada na classe onde é criada.

**Modificadores de Acesso: static e final**

1. static -> mecanismo exclusivo de Java que usa-se em atributos e métodos, é acessado através da classe pertencendo assim a classe e não aos objetos dela, podendo somente ser acessado por outro método também static.

2. final -> é aplicável a classes, atributos e métodos, sendo impossível uma classe final ser herdade. Atributos final não podem ter seus valores alterados (*a*

*referencia permanece*) e seus métodos não podem ser sobrescritos.