Assim como no mundo real, os objetos na OO também se relacionam e este relacionamento ocorre de diferentes formas e graus.

Basicamente podemos identificar os relacionamentos temporários e duradouros entre objetos.

Um relacionamento temporário é aquele em que, para cumprir determinada responsabilidade, um objeto necessita de outro objeto (conhecê-lo e solicitar suas operações) apenas durante a execução desta responsabilidade. Este tipo de relacionamento é chamado de dependência e ocorre no nível dos métodos.

Os relacionamentos duradouros são aqueles em que durante a vida de um objeto ele está ligado a outro para cumprir várias de suas responsabilidades. Estes relacionamentos podem ser de associação ou de agregação e ocorrem no nível dos atributos.

1.1        Dependência

Como dito, a relação de dependência ocorre entre dois objetos quando temporariamente um necessita do outro – assim, o objeto A necessita/depende do objeto B.

No diagrama de classe da UML isto é representado por uma linha pontilhada, terminada em uma seta da classe dependente para a classe utilizada.

Em termos de código, a dependência é percebida quando um objeto é utilizado como variável local a um método (parâmetro ou declarado localmente).

Como exemplo, imagine a situação onde um objeto de projetor possua um método limpar(). Para executar esta operação ele precisa temporariamente de um objeto de Pano, que o servirá apenas naquele instante.

class Projetor {

...

public void limpar(){

     Pano umPano = new Pano();  // exemplo de dependência

     umPano.esfregar();         // por declaração local

...

}

public void limpar(Pano p){  // exemplo de dependência

     ...                                // por parâmetro

     p.esfregar();

     ...

}

}

class Pano {

...

}

1.2        Associação

A relação de associação ocorre entre dois objetos quando este relacionamento é uma das características de um dos objetos, isto é, um dos atributos relevantes de um objeto é o outro objeto. Assim, o relacionamento é mais duradouro entre o objeto A e o objeto B.

O motivo ou significado do relacionamento deve ser expresso, ou seja, deve-se nomear o relacionamento.

No diagrama de classe da UML isto é representado por uma linha, com a **nomeação** da associação indicada.

Na associação, além de nomeá-la, devemos também indicar sua **multiplicidade**. Lembre-se da teoria dos conjuntos. Para a multiplicidade vamos identificar quantos objetos de um conjunto se relacionam com quantos objetos do outro conjunto.

Como exemplo, imagine que um Projetor está alocado ou instalado em uma Sala. Identificar a multiplicidade é fazer duas perguntas:

* + Quantos Projetores podem estar alocados em uma mesma Sala (0, 1 ou vários)?
  + Em quantas Salas pode estar alocado o mesmo Projetor (0, 1 ou várias)?

Então a leitura se dá: um objeto de Projetor está alocado a nenhum ou no máximo a um objeto de Sala. Um objeto de Sala aloca nenhum ou vários objetos de Projetor.

Em termos de código, a associação é implementada colocando como **atributo** de uma das classes um *handle* para objeto(s) da outra.

Qual classe conterá o atributo? Depende do contexto do problema e da maneira como se quer projetar a solução. Normalmente a classe que participa com multiplicidade 1 é atributo da outra. Mas para deixar isto bem claro no diagrama, ainda existe a notação da **navegabilidade** (uma seta no final da linha de associação), que indicará a partir de qual objeto existe a referência para o outro.

Como atributo, devemos lembrar que seu conteúdo será encapsulado. Isto requer a implementação de métodos para manipular esta referência.

class Projetor {

...

     private Sala umaSala;

public void setSala(Sala s){

...

     this.umaSala = s;

...

}

}

class Sala {

...

}

* Mais informações nos dias 2 e 3 do livro
  + SINTES, Anthony. Aprenda Programação Orientada a Objetos. Makron books, 2002.

1.3        Agregação

A relação de agregação ocorre entre dois objetos quando está envolvido o conceito de “todo-parte”, isto é, um objeto forma ou compõe o outro objeto.

Neste momento não discutiremos a diferença entre agregação e composição, assumindo que ocorrendo “todo-parte” será modelada a agregação.

No diagrama de classe da UML isto é representado por uma linha com um losango junto à classe que indica o todo na relação. Não há necessidade de nomear uma agregação, pois seu significado já está determinado: a parte (Classe B) está contida no todo (Classe A); o todo tem uma parte.

Também aqui devemos indicar sua multiplicidade, fazendo as mesmas perguntas da associação. Imagine que o campus universitário é formado por Blocos de edifícios e que estes são formados por Salas:

* + Quantos Salas podem estar existir em um mesmo Bloco (0, 1 ou várias)?
  + Em quantos Blocos pode estar alocada a mesma Sala (0, 1 ou vários)?

Então a leitura se dá: um objeto de Bloco é formado por no mínimo 1 ou vários objetos de Sala. Um objeto de Sala faz parte de 1 objeto de Bloco.

Em termos de código, a agregação é implementada colocando como atributo de uma das classes (todo) um *handle* para uma coleção de objetos da outra (parte). E normalmente esta é a navegabilidade esperada (que fica implícita no diagrama).

Como atributo, devemos lembrar que seu conteúdo será encapsulado. Isto requer a implementação de métodos para manipular esta coleção.

class Bloco {

...

     private List salas = new ArrayList();

public void addSala(Sala s){

...

     salas.add(s);

...

}

}

class Sala {

...

}