Interação entre objetos

Nas aulas anteriores foi visto o que são classes e objetos e como são implementados. Agora daremos um passo adiante. Para construir aplicações interessantes não é suficiente construir objetos de funcionamento individual. Em vez disso, os objetos devem ser combinados de modo a cooperar para realizar uma tarefa comum. Aqui veremos uma pequena aplicação com uma classe que tem atributos de outra classe.

Considere a classe `Estudante`, presente na Lista 6, ela pode ser parte de um projeto. Vamos criar um projeto e a classe `Estudante` fará parte deste projeto. Considere o projeto chamado `Lab-classes`, dentro deste projeto adicione a classe `Estudante` (lembre-se de colocar dentro de um pacote). Dentro do mesmo pacote, crie uma classe chamada `Turma`. A Figura 1 mostra como pode ficar a árvore do projeto. A classe principal que pode ser usada com o método `main` pode ficar em outro pacote.

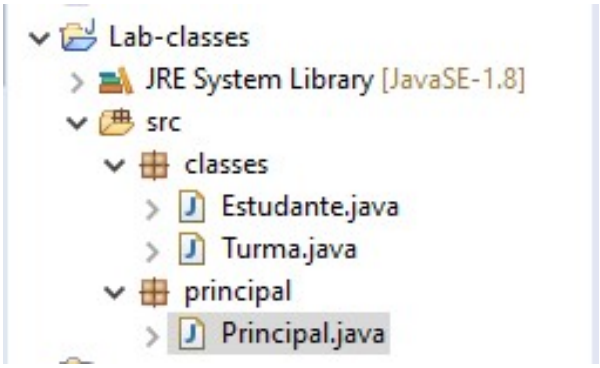

Figura 1: Árvore do projeto `Lab-classes`.

A classe `Turma` servirá para criar turmas com informações de sala, horário, e lista de inscritos. A lista de inscritos serão objetos da classe `Estudante`. A Figura 2 mostra quais são os atributos da classe `Turma`.

```
17  */
18 public class Turma {
19     private String instrutor;
20     private String sala;
21     private String diaHorario;
22     private ArrayList<Estudante> estudantes;
23     private int capacidade;
24
```

Figura 2: Atributos da classe `Turma`.

Observe que na linha 22 há a declaração do atributo `estudantes`. O comando cria uma lista da classe `ArrayList` com elementos do tipo `Estudante`. Este comando gera

uma colaboração entre as classes `Turma` e `Estudante` do tipo "tem uma", ou seja, a turma tem uma lista de estudantes. Este tipo de colaboração pode ser representado através de diagramas de classes como mostrado na Figura 3 onde a seta indica esta relação.

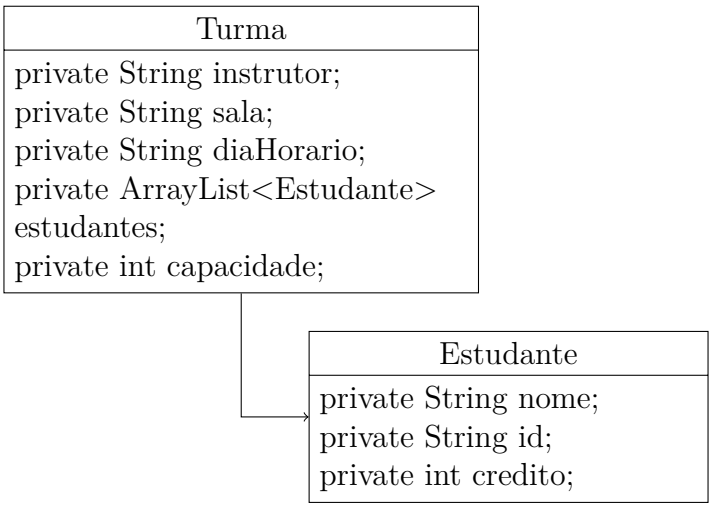

Figura 3: Representação de um diagrama de classes.

O construtor da classe `Turma` cria uma turma com uma capacidade máxima definida, ver Figura 4. Os demais atributos são inicializados com valor padrão. Observe que na linha 33 há a instanciação da lista de `Estudantes`. Já que o atributo `estudantes` é do tipo objeto, deve haver a instanciação usando o comando `new`. E assim deve ocorrer com qualquer declaração de atributo do tipo objeto. Na Figura 2 há apenas a criação da referência e no construtor que ocorre a instanciação propriamente dita. Se a instanciação do atributo do tipo objeto tiver parâmetros, é nesse momento que ele é passado.

```
25    /**
26     * Cria uma turma com um número máximo de inscritos.
27     * Todos os outros atributos são configurados com valores padrão.
28     */
29    public Turma(int maxNumeroEstudantes) {
30        instrutor = "";
31        sala = "";
32        diaHorario = "";
33        estudantes = new ArrayList<Estudante>();
34        capacidade = maxNumeroEstudantes;
35    }
```

Figura 4: Construtor da classe `Turma`.

O método para inscrever estudantes na turma é feito usando como parâmetro um objeto Estudante, ver Figura 5. Antes de adicionar o objeto na lista é feita uma verificação da capacidade da turma, isso pode ser visto na linha 42 usando o método `size()` já que `estudantes` é uma lista do tipo `ArrayList`. Caso seja não haja restrições, o estudante é adicionado na lista usando o método `add()` na linha 46.

```
37      /**
38       * Adiciona um estudante na turma. Se a turma estiver cheia não
39       * aceita mais adições
40       */
41      public void inscreverEstudante(Estudante novoEstudante) {
42          if(estudantes.size() == capacidade) {
43              System.out.println("A turma está cheia, inscrição não efetuada.");
44          }
45          else {
46              estudantes.add(novoEstudante);
47          }
48      }
```

Figura 5: Método para inscrever estudantes na turma.

O método `imprimeLista()`, na Figura 6, lista todo os detalhes da turma, incluindo a lista de estudantes. Na linha 86 é usado o for-each para percorrer toda a lista e o método `print()` da classe `Estudante` é usado para listar o nome e id do estudante, este método já está bem definido na classe.

```
82      public void imprimeLista() {
83          System.out.println("Turma " + diaHorario);
84          System.out.println("Instrutor: " + instrutor + "   sala: " + sala);
85          System.out.println("Lista da turma:");
86          for(Estudante estudante : estudantes) {
87              estudante.print();
88          }
89          System.out.println("Number of students: " + numeroDeEstudantes());
90      }
91  }
```

Figura 6: Método para imprimir os detalhes da turma.

Os demais métodos da classe são métodos modificadores do tipo *set* para atualizar os valores dos demais atributos.

Prática

Implemente este projeto e complete a classe `Turma` com os demais métodos e faça testes, crie uma turma, adicione alunos e mostre os detalhes dela. A classe Turma será disponibilizada no SIGAA, mas é interessante que você antes tente implementar para entender bem como funciona a construção e uso das classes.