Exercício Prático Aula Prática 6 - 27/04/2017
Programação de Computadores 2 - Programação Orientada a Objetos
Coleções
Prof. Daniel Hasan Dalip – hasan@decom.cefetmg.br

Apos finalizado o exercício, salve os projetos em um zip e envie-o pelo Moodle . Não será aceito o recebimento via e-mail. Plágio não será tolerado e tanto a pessoa que forneceu quanto a que utilizou perderão os pontos da prática.

Entrega:

Exercicio 1 e método escolheCidades do Exercicio 2: até o fim da aula

Finalizar o Exercicio 2: até 03/05/2017

a) Faça uma classe TelaTime, no mesmo pacote da classe atleta e do Time. Mude a visibilidade do atributo do arranjo de atletas para **protected**.

b) Utilizando o atributo (diretamente, e não por meio de metodos) crie o método "imprimeQuantidadeDePessoaPorIdade" este método deverá imprimir na tela a quantidade de pessoas por idade. Por exemplo, supondo que tenha o seguinte conjunto de atletas:

Afonso, 23 anos
Barbara, 40 anos
Carol, 23 anos
Danielle, 40 anos

O sistema deverá imprimir:
Idade: 23 quantidade: 2
Idade: 40 quantidade: 2

Para fazer isso, utilize um HashMap onde as idades serão as chaves e, os valores, a quantidade de pessoas com essa idade. Assim, você irá navegar pelo arranjo de atletas e povoar o hashmap contabilizando os atletas.

Como estamos testando a importância do encapsulamento, para navegar nos atletas, **não use o metodo get**, use o atributo diretamente. Além disso use um for simples: for(int i = 0; i<arrAtletas.length ; i++) {. ... }. Faça isso e imprima o resultado.

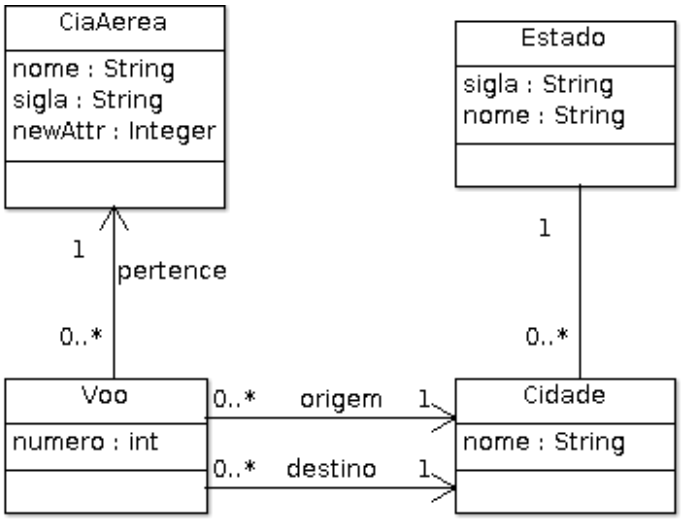
c) Agora, replique essa classe (dando ctrl+c ctrl+v no arquivo, por exemplo) deixe com nome TelaTimeEncapsulamento. Faça apenas uma alteração: ao invés de usar os atributos da classe diretamente, use o metodo getArrAtletas() para extrair os atletas.

d) Na classe Time, agora você irá fazer uma modificação na estrutura e verá quais classes serão as mais afetadas. Ao invés de usar um arranjo para armazenar os atletas, agora use uma lista. No método que retorna o arranjo, usado na letra (c), você deverá transformar a lista em um arranjo. Para transformá-la, dentro do método, crie um arranjo com o tamanho da lista, navegue na lista adicionando cada elemento no arranjo e, depois, o retorne-o.

Perceba que não foi necessário alterar nada em TelaTimeEncapsulamento, porém, na tela TelaTime, você precisou alterar o for. A ideia do encapsulamento é esta: caso uma classe mude sua estrutura interna, as demais classes não irão ser afetadas.

Reflita: Imagine se existem dois sistemas: um com centenas de classes como a **TelaTime** e, outro, com centenas de classes como a **TelaTimeEncapsulamento**. O sistema de mais fácil manutenção seria o que utiliza a classe **TelaTimaEncapsulamento**. Por isso, o encapsulamento é importante.

Exercício 2: Seja o seguinte diagrama (que foi desenhado na aula de UML):
Para o exercício abaixo, baixe o projeto aula_pratica_06_VoosPratica.zip (disponível no Moodle). Neste projeto existem 3 pacotes: o pacote de util com a classe Console, para ser usado ao solicitar algo do usuário, o pacote geo com as classes Cidade e Estado e o pacote ciasAereas com as classes CiaAerea, Voo e TelaGerenciarVoo. Este último pacote é o que vocês deverão utilizar para fazer o exercício, com o auxílio dos demais.



Similar à aula de UML, esta é a modelagem de parte do sistema de reserva de passagens. Onde voos são disponibilizados pela companhia aérea e, cada voo, possui sua cidade origem e destino. Foi feito uma simplificação considerando apenas voos nacionais. Assim, neste modelo, não foi utilizado a classe **Pais**, apenas **Cidade** e **Estado**.

Colocamos também a navegabilidade de cada associação: um voo deverá ter acesso à qual companhia aérea ele pertence e suas cidades origem e destino. A cidade não precisará de visualizar quais são os voos destino nem voos origem. Pode-se decidir na implementação se a cidade visualizará o estado no qual ele pertence ou se o estado visualizará a lista de cidades. Também poderá ser feito em ambas as classes.

Nesta prática você deverá utilizar a implementação de tais classes no arquivo “aula_pratica_06_VoosPratica.zip” no Moodle.

Para fazer com que a cidade tome conhecimento do estado e o estado conheça suas cidades, você poderá ver que a classe Estado possui uma lista de cidades e a classe Cidade possui o objeto que representa seu estado (passado como parâmetro no contrutor). **Veja essas classes e entenda o seu funcionamento.**

Além disso, veja que nas classes Voo, Cidade e CiaAerea foram implementados seus métodos toString para auxiliar na prática de hoje.

Para gerenciarmos as informações referente aos voos, criou-se a classe TelaGerenciaVoos. Ela é responsável por adicionar voos. Para isso, nesta tela assumi-se que as cidades e companhia aéreas já foram cadastradas. Em nosso protótipo, criamos as cidades e companhias aéreas através dos métodos **criaCidades** e **criaCiasAereas.** O método exibeTela exibe os voos e faz o

cadastro de voos adicionais. Para isso, tal método utiliza imprimeVoos e criaVoo.

Nem todos os métodos foram implementados. Primeiramente, leia e entenda todo o código. Logo após, você deverá implementar tais métodos conforme especificado nos comentários de cada um. Segue uma breve descrição de cada método que deverá ser implementado:

- **escolheCidade**: Este método deve listar todas as cidades e solicitar para que o usuário escolha uma. Tal cidade será o retorno deste método.
- **escolheCiaAerea**: Exibe todas as companhias aéreas e solicita ao usuário a escolha de uma delas. Note que a companhia aérea foi implementada através de um Map (dicionário) e o usuário deverá entrar com a sigla da companhia para escolhê-la.
- **criaVoo**: Este método irá usar os dois métodos implementados (escolheCidade e escolheCiaAerea) para criar um Voo.
- **imprimeVoos**: imprime todos os voos.

Para mais informações dos métodos, veja o próprio código. Dica: **teste um método de cada vez.** Por exemplo, dentro do método exibeTela, comente o bloco completo do while e, logo após, faça o teste para escolha da Cidade:

```
TelaGerenciaVoos tela = new TelaGerenciaVoos(mapCiasAereas, lstCidades);
Cidade c =  tela.escolheCidade();
System.out.println(c);
/*String resp = "S";
//resp.equalsIgnoreCase("S") == Se a resposta foi "S" ou "s".
while(resp.equalsIgnoreCase("S"))
{
      tela.imprimeVoos();
      tela.criaVoo();
      tela.imprimeVoos();
      resp = Console.readString("Deseja criar mais um voo (S/N)? ");
      while(!resp.equalsIgnoreCase("S") && !resp.equalsIgnoreCase("N")){
         System.out.println("Resposta inválida!");
         resp = Console.readString("Deseja criar mais um voo (S/N)? ");
      }
}*/
```