FCT/Unesp – Presidente Prudente

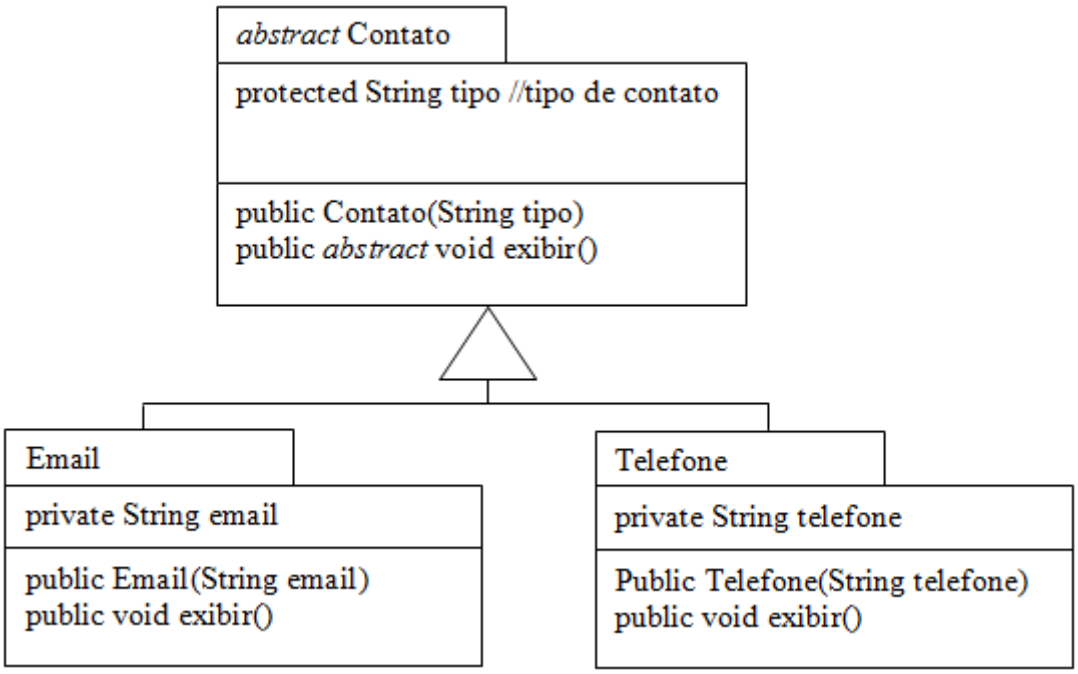
Programação Orientada a Objetos

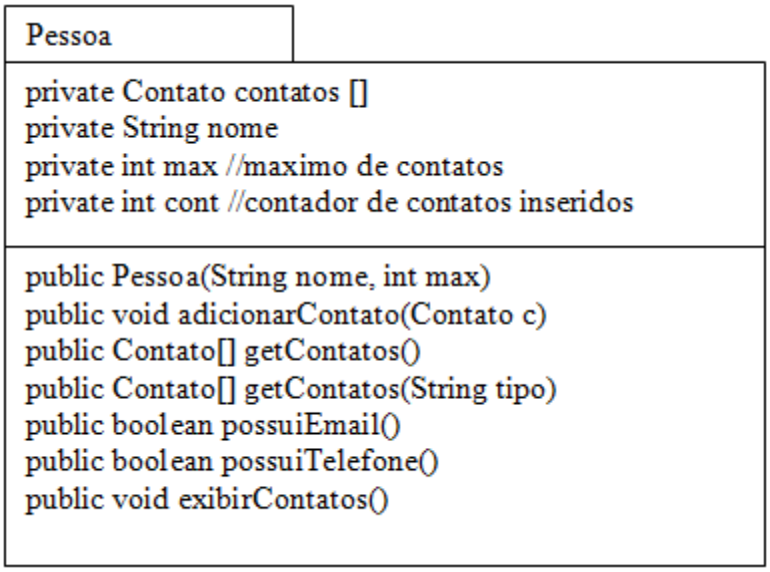
Prof. Danilo Medeiros Eler

Classe Abstrata, Herança e Polimorfismo

Exercícios da Aula 06

1) Os diagramas abaixo representam classes de parte de um programa de uma agenda que armazena pessoas. Uma pessoa pode ter vários contatos, ou seja, pode ter vários números de telefone ou de email.

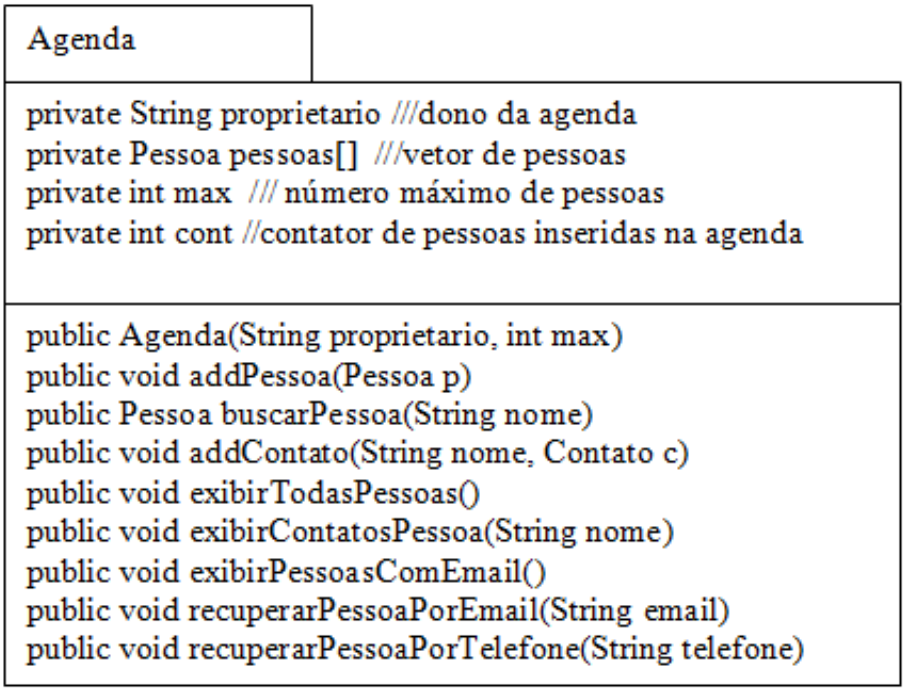




Breve descrição dos métodos da classe Pessoa:

- O método **Contato[] getContatos()** retorna o vetor de contatos por completo.
- O método **Contato[] getContatos(String tipo**) retorna um vetor com os contatos do tipo especificado.
- O método **boolean possuiEmail() –** verifica se a pessoa tem algum contato do tipo email, retornando verdadeiro ou falso
- O método **boolean possuiTelefone() –** verifica se a pessoa tem algum contato do tipo telefone, retornando verdadeiro ou falso
- O método **void exibirContatos() –** exibe o nome da pessoa e os seus contatos, ou seja, todos os números de telefone e endereços de email

O programa também deve armazenar uma lista de pessoas em uma agenda. Para isso, implemente uma classe Agenda que é uma abstração de uma agenda de pessoas, as quais são representadas pela classe Pessoa. O diagrama da classe Agenda e algumas de suas especificações estão descritos a seguir.



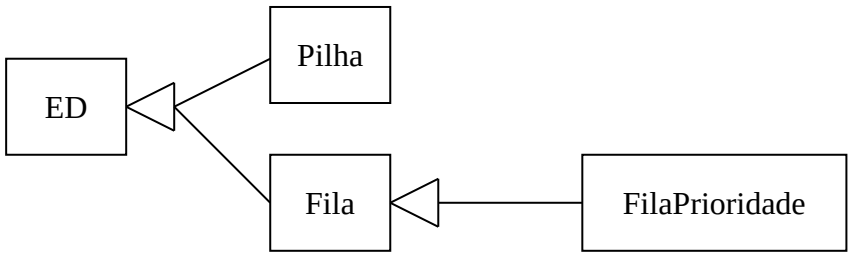
Breve descrição dos métodos da classe Agenda:

- O método **buscarPessoa(String nome)** retorna a pessoa que possui o nome passado como parâmetro. Se não encontrar, deve retornar *null*. Esse método deve ser utilizado sempre que for necessário realizar uma busca no programa.
- O método **addContato(String nome, Contato c)** faz uma busca para encontrar a pessoa que tem o nome passado por parâmetro, se encontrar, adiciona o contato **c** à pessoa que possui o **nome** passado como

parâmetro; caso contrário, se a pessoa não estiver na agenda, não se faz nada. **Utilize** o método **buscarPessoa(String nome)** já implementado para encontrar a pessoa.

- O método **exibirTodasPessoas()** exibe os dados das pessoas cadastreadas na agenda e seus contatos.
- O método **exibirContatosPessoa(String nome)** exibe os contatos da pessoa cujo nome é aquele passado no parâmetro **nome**.
- O método **exibirPessoasComEmail()** exibe somente o nome das pessoas que possuem email.
- Os métodos **recuperarPessoaPorEmail** e **recuperarPessoaPorTelefone** passam, respectivamente, um email ou um telefone por parâmetro. Esse método busca tal informação em cada uma das pessoas cadastradas (inseridas) na agenda. O método deve exibir o nome das pessoas que possuírem o email (no caso do método **recuperarPessoaPorEmail**) ou o telefone (no caso do método **recuperarPessoaPorTelefone**).

2) Utilizando classe abstrata, herança e polimorfismo, faça um programa para representar três tipos de estruturas de dados: Pilha, Fila e Fila com Prioridade. Faça uma classe abstrata chamada ED que possui os métodos abstratos **adicionar** e **remover**. Esses métodos deverão ser implementados pelas subclasses conforme o comportamento de cada uma. Note que a Fila com Prioridade é subclasse da Fila, portanto somente o método **adicionar** deverá ser reimplementado (sobrescrito). Para simplificar, as estruturas armazenarão elementos do tipo inteiro em um vetor estático.



Para testar esse programa você deve criar uma variável do tipo ED e adicionar elementos ao objeto referenciado por ela. Troque o tipo de objeto instanciado e verifique o comportamento das diferentes estruturas. Por exemplo:

```
ED estrutura = new Pilha(10); //uma pilha com 10 posições
//ED estrutura = new Fila(10); //uma fila com 10 posições
//ED estrutura = new FilaPrioridade(10); //uma fila com prioridade com 10 posições
estrutura.adicionar(10);
estrutura.adicionar(5);
estrutura.adicionar(7);
estrutura.adicionar(2);
estrutura.adicionar(4);
```

```
int elemento = estrutura.remover();
elemento = estrutura.remover();
elemento = estrutura.remover();
elemento = estrutura.remover();
elemento = estrutura.remover();
```

Você deve exibir os elementos removidos para verificar o comportamento da estrutura. Modifique os comentários das instanciações iniciais e verifique o comportamento das diferentes estruturas. Perceba que apesar da mudança do tipo de objeto instanciado, os métodos invocados não são alterados. Entretanto, o comportamento irá variar de acordo com a estrutura de dados utilizada.