Tecnologia em Análise e Desenvolvimento de Sistemas

Programação Orientada a Objetos

Revisão, Generics e Interfaces

1º/2018

Situação problema - revisão

- Cliente deseja implementar um sistema para controle de uma fila bancária para atendimento dos clientes.
  - O sistema de controle de fila de um banco opera da seguinte forma:
    - O cliente que chega a agência retira uma senha que pode ter prioridade preferencial (para gestantes, portadores de necessidades especiais e maiores de 65 anos) ou não preferencial.
    - A agência dispõe de um caixa preferencial e vários caixas não preferenciais. Ao decidir qual o próximo cliente a ser atendido, são consideradas duas políticas:
      - O caixa preferencial seleciona o primeiro cliente preferencial da fila. Não havendo clientes preferenciais, é selecionado o primeiro cliente da fila;
      - O caixa não preferencial respeita a ordem de chegada e seleciona o primeiro cliente que está aguardando na fila, preferencial ou não.

Solução

Para começar a resolver o problema é preciso entender o que é uma fila, então, lembremos:

Fila é uma estrutura de dados ordenada pela entrada na qual o primeiro elemento a entrar é o primeiro elemento a sair.

Existem várias formas de implementar fila, mais adiante escolheremos uma. Por hora, vamos pensar em uma solução genérica para fila, ou seja, uma classe abstrata.

Implementado no projeto Exemplo12

```
package model;
public class Cliente {
    private int senha;
    private boolean preferencial;
    public Cliente(int senha, boolean preferencial) {
        this.senha = senha;
        this.preferencial = preferencial;
    }
    public int getSenha() {
        return senha;
    }
    public void setSenha(int senha) {
        this.senha = senha;
    }
    public boolean isPreferencial() {
        return preferencial;
    }
    public void setPreferencial(boolean preferencial) {
        this.preferencial = preferencial;
    }
    @Override
    public String toString() {
        return "Cliente: " + getSenha();
    }
}
```

Inicialmente a classe Cliente

```
package model;
public abstract class Fila {
    public abstract boolean enqueue(Cliente c);
    public abstract Cliente dequeue();
    public abstract boolean isFull();
    public abstract boolean isEmpty();
}
```

O que é uma classe abstrata mesmo? Para que serve?

Pode-se definir classe abstrata como sendo uma classe não instanciável, que apresenta pelo menos um método para o qual não existe implementação.
O objetivo das classes abstratas é apenas organizar características comuns a diversas classes, visando facilitar a manipulação da complexidade e a reutilização de código.

Implementamos a classe abstrata Fila.

Uma implementação válida é a implementação de uma fila estática circular.

No caso, definiu-se uma fila de até 300 Clientes.

Se eu mudar o Cliente para outra classe a fila funciona para outros objetos?

Será que aquele tal de polimorfismo que complicou na prova ajudaria?

```java
public class FilaCircular extends Fila {
    private final int MAXIMO = 300;
    private Cliente[] elementos;
    private int inicio, fim, tamanho;
    public FilaCircular() {
        elementos = new Cliente[MAXIMO];
        inicio = 0;
        fim = -1;
        tamanho = 0;
    }
    @Override
    public boolean enqueue(Cliente c) {
        boolean deuCerto = false;
        if(!isFull() && c != null){
            if(fim == MAXIMO-1){
                fim = 0;
            }else{
                fim = fim + 1;            }
            elementos[fim] = c;
            tamanho = tamanho + 1;
            deuCerto = true;
        }
        return deuCerto;
    }
    @Override
    public Cliente dequeue() {
        Cliente c = null;
        if(!isEmpty()){
            c = elementos[inicio];
            tamanho = tamanho - 1;
            if(inicio == MAXIMO-1){
                inicio = 0;
            }else{
                inicio = inicio + 1;            }
        }
        return c;
    }
    @Override
    public boolean isFull() {
        return tamanho == MAXIMO;
    }
    @Override
    public boolean isEmpty() {
        return tamanho == 0;
    }
}
```

```java
public class FilaGenericaCircular<T> implements FilaGenerica<T> {
    private final int MAXIMO = 300;
    private Object[] elementos;
    private int inicio, fim, tamanho;
    public FilaGenericaCircular() {
        elementos = new Object[MAXIMO];
        inicio = 0;
        fim = -1;
        tamanho = 0;    }
    @Override
    public boolean enqueue(T c) {
        boolean deuCerto = false;
        if(!isFull() && c != null){
            if(fim == MAXIMO-1){
                fim = 0;
            }else{
                fim = fim + 1;
            }
            elementos[fim] = c;
            tamanho = tamanho + 1;
            deuCerto = true;
        }
        return deuCerto;
    }
    @Override
    public T dequeue() {
        T c = null;
        if(!isEmpty()){
            c = (T)elementos[inicio];
            tamanho = tamanho - 1;
            if(inicio == MAXIMO-1){
                inicio = 0;
            }else{
                inicio = inicio + 1;
            }
        }
        return c;
    }
    @Override
    public boolean isFull() { return tamanho == MAXIMO;  }
    @Override
    public boolean isEmpty() { return tamanho == 0; }
}
```

```java
package model;
public interface FilaGenerica<T> {

    public boolean enqueue(T c);

    public T dequeue();

    public boolean isFull();

    public boolean isEmpty();
}
```

```java
package view;
import model.Cliente;
import model.FilaGenericaCircular;
public class TesteFilaGenericaCircular {
    public static void main(String[] args) {
        FilaGenericaCircular<Cliente> fila =
                new FilaGenericaCircular<>();
        fila.enqueue(new Cliente(1, false));
        fila.enqueue(new Cliente(2, false));
        fila.enqueue(new Cliente(3, false));
        fila.enqueue(new Cliente(4, false));
        while(!fila.isEmpty()){
            System.out.println(fila.dequeue().toString());
        }
    }
}
```

O que é esse <T>?

O que é interface e implements?

Classes Genéricas

Classe Genérica

- Analisando o exemplo da Fila Circular:
  - observa-se que o código da classe não mudaria se os elementos da Fila fossem Strings ou Pets ou qualquer outro tipo de objeto.
  - o que mudaria seria o tipo de entrada do enqueue() e o tipo de retorno do dequeue().
- Para melhorar o **reuso** de software, a partir do Java 5, foi disponibilizado o uso de Classe Genérica, que apoiada pelo polimorfismo (compatibilidade de tipos) permite maravilhas na implementação, reduzindo a quantidade de **cast** que deve ser realizada.

Material Adicional

- **Leitura Obrigatória**
  - Winder, R.; Roberts, G. Desenvolvendo Software em Java. 3 ed. Rio de Janeiro: LTC, 2009.
    - Capítulos 6.3
  - https://www.devmedia.com.br/usando-generics-em-java/28981
  - http://argonavis.com.br/cursos/java/j100/java-5-generics.pdf
  - https://javaprogramando.blogspot.com.br/2009/02/artigo-classes-genericas.html

Interfaces

Interfaces

- Uma interface é sintaticamente semelhante a uma classe abstrata no fato de podermos especificar um ou mais métodos sem corpo.
- Esses métodos devem ser implementados por uma classe para que suas ações sejam definidas.
- Logo uma interface **especifica o que deve ser feito, mas não como deve ser feito**.
- Quando uma interface é definida, não existe um número de classes que podem implementá-la (implements), além disso, uma classe pode implementar qualquer número de interfaces.

Interface x Classe Abstrata

- **Interfaces:**
  - Uma interface não é considerada uma Classe e sim uma Entidade.
  - Não possui implementação, apenas assinatura, ou seja, apenas a definição dos seus métodos sem o corpo.
  - Todos os métodos são abstratos.
  - Seus métodos são implicitamente Públicos e Abstratos.
  - Não há como fazer uma instância de uma Interface e nem como criar um Construtor.
  - Funcionam como um tipo de "contrato", onde são especificados os atributos, métodos e funções que as classes que implementem essa interface são obrigadas a implementar.
  - Já que Java não suporta Heranças Múltiplas, as Interfaces são usadas para implementá-las.

- **Classes Abstratas:**
  - As classes abstratas devem conter pelo menos um método abstrato, que não tem corpo.
  - É um tipo especial de classe que não há como criar instâncias dela.
  - É usada apenas para ser herdada, funciona como uma super classe.
  - Uma grande vantagem é que força a hierarquia para todas as subclasses.
  - É um tipo de contrato que faz com que as subclasses contemplem as mesmas hierarquias e/ou padrões.

Material Adicional

- **Leitura Obrigatória**
  - Winder, R.; Roberts, G. Desenvolvendo Software em Java. 3 ed. Rio de Janeiro: LTC, 2009.
    - Capítulos 7.6
  - https://www.devmedia.com.br/interfaces-x-classes-abstratas/13337
  - https://www.devmedia.com.br/forum/duvidas-simples-de-certificacao-java-diferenca-entre-classe-abstrata-e-interface/566376
  - https://www.caelum.com.br/apostila-java-orientacao-objetos/classes-abstratas/#null
  - https://www.caelum.com.br/apostila-java-orientacao-objetos/interfaces/#interfaces

```
package model;
public interface FilaComPrioridade {
    public Cliente dequeuePrioritario();
}
```

Nova interface com uma característica muito particular.

Como vamos terminar o sistema da fila do banco?

```java
package model;
public class FilaAtendimentoBancario<T>
        extends FilaGenericaCircular<T>
        implements FilaComPrioridade {
    @Override
    public Cliente dequeuePrioritario() {
        Cliente c = null;
        Cliente retorno =  null;
        int contador, diff;
        boolean achou;
        contador = 1;
        achou = false;
        while (contador <= getTamanho() && !achou){
            c = (Cliente) dequeue();
            contador++;
            if(c.isPreferencial()){
                retorno = c;
                achou = true;
            }else{
                enqueue((T) c);
            }
        }
        diff = (getTamanho() - contador)+1;
        while(diff >= 0){
            enqueue(dequeue());
            diff--;
        }
        return retorno;
    }
}
```

Faltava apenas a implementação de do método da interface.

Se mudarmos a tecnologia da implementação a interface continua sendo necessária, assim todos os métodos deverão ser implementados.

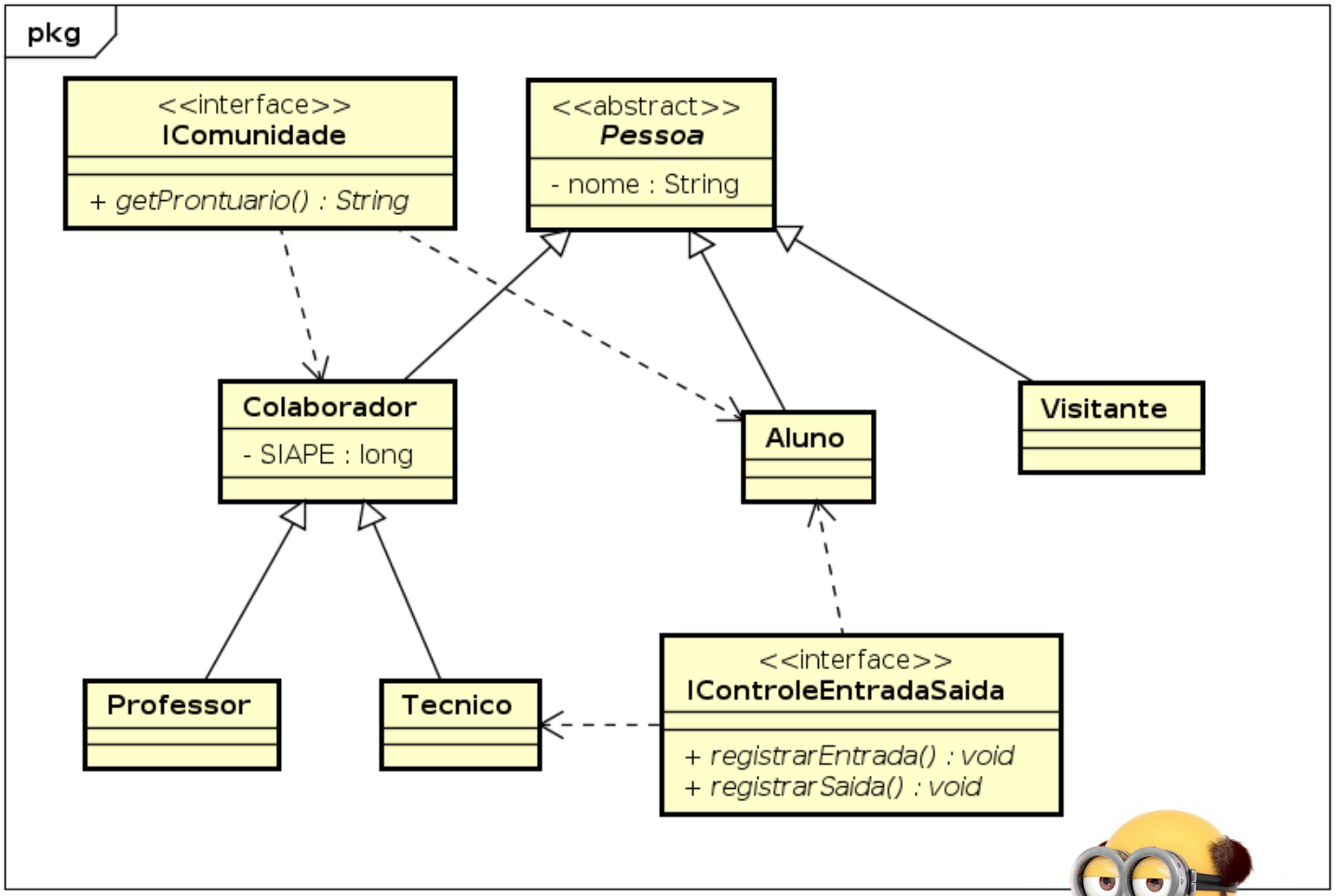
Esse foi o Exemplo12.

Vamos analisar outro exemplo usando fila.

Agora o Exemplo13

Vamos pensar em um sistema para controle e registro de acesso para implantação na portaria do câmpus.

Vamos observar o diagrama, ele é incompleto, mas podemos dar início ao nosso projeto.



Acho que vamos implementar esse!

Material Adicional

- **Leitura Obrigatória**
  - Winder, R.; Roberts, G. Desenvolvendo Software em Java. 3 ed. Rio de Janeiro: LTC, 2009.
    - Capítulo 22

Trabalhando

- **Exercício de Fixação**
  - 10
  - Controle de Acesso