Tecnologia em Análise e Desenvolvimento de Sistema

Programação Orientada a Objetos – POOS3

Laboratório 2

Objetivo

- Trabalhar os conceitos de herança e **polimorfismo** em sistemas orientados a objeto.

Atividade

- Implementar um sistema de agenda de contatos.

Início

- Crie um novo projeto chamado ProjLaboratorio2.
- No diretório src crie os pacotes:
  - model
  - data
  - view
- No pacote model crie as classes:
  - Contato
  - Telefone
- No pacote data crie a classe:
  - Agenda
- No pacote view crie a classe:
  - Main

Classe Telefone

- Telefone é composto de ddd, prefixo e número, todos atributos inteiros.
- DDDs possuem três digitos, prefixo e número 4 digitos.

```
package model;

public class Telefone {

    protected int ddd;
    protected int prefixo;
    protected int numero;

    public Telefone(int ddd, int prefixo, int numero) {
        this.ddd = ddd;
        this.prefixo = prefixo;
        this.numero = numero;
    }

    public void setDdd(int ddd){
        if(ddd >= 10 && ddd <= 99){
            this.ddd = ddd;
        }else{
            this.ddd = 0;
        }
    }


    public void setPrefixo(int prefixo) {
        if(prefixo >= 1111 && prefixo <= 9999){
            this.prefixo = prefixo;
        }else{
            this.prefixo = 0;
        }

    }

    public void setNumero(int numero) {
        if(numero >= 0 && numero <= 9999) {
            this.numero = numero;
        }else{
            this.numero = 0;
        }
    }
```

```
    public boolean isValido(){
        boolean retorno = false;
        if(ddd > 0 && prefixo > 0){
            retorno = true;
        }
        return retorno;
    }


    @Override
    public String toString() {
        StringBuffer stringBuffer = new StringBuffer();
        int contaCasas, fone;
        if(!isValido()){
            stringBuffer.append("(xxx) xxxx-xxxx");
        }else {
            stringBuffer.append("(0" + ddd + ") ");
            stringBuffer.append(prefixo + "-");
            contaCasas = 0;
            fone = numero;
            while((fone / (int)(Math.pow(10, contaCasas))) > 0){
                contaCasas++;
            }
            while (contaCasas < 4){
                stringBuffer.append("0");
                contaCasas++;
            }
            if(numero != 0)
                stringBuffer.append(this.numero);
        }
        return stringBuffer.toString();
    }
}
```

Classe Contato

- Possui nome e um telefone.

```
package model;

public class Contato {

    protected String nome;
    protected Telefone telefone;

    public Contato(String nome, int ddd, int prefixo, int numero) {
        this.nome = nome.toUpperCase();
        telefone = new Telefone(ddd, prefixo, numero);
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome.toUpperCase();
    }

    public String getTelefone() {
        return telefone.toString();
    }

    public void setTelefone(int ddd, int prefixo, int numero) {
        this.telefone.setDdd(ddd);
        this.telefone.setNumero(numero);
        this.telefone.setPrefixo(prefixo);
    }

    @Override
    public String toString() {
        return "Contato: " + nome + "  \t Telefone: " + telefone.toString();
    }
}
```

Classe Agenda

- Organiza os contatos em um array de Contatos.

```
package data;
import model.Contato;

public class Agenda {
    protected Contato[] listaContatos;
    protected int qtdMaxima;
    protected int qtdCadastrados;

    public Agenda(int tamanho) {
        if(tamanho < 1){
            tamanho = 1;
        }
        this.qtdMaxima = tamanho;
        this.qtdCadastrados = 0;
        listaContatos = new Contato[this.qtdMaxima];
    }

    public boolean isFull(){      return qtdCadastrados == qtdMaxima;    }

    public boolean isEmpty(){     return qtdCadastrados == 0; }

    public boolean add(Contato contato){
        boolean deuCerto = false;
        if(!isFull() && contato != null){
            listaContatos[qtdCadastrados] = contato;
            qtdCadastrados += 1;
            deuCerto = true;
        }
        return deuCerto;
    }

    public String getAgenda(){
        StringBuffer stringBuffer = new StringBuffer();
        Contato c;
        for(int i=0; i < this.qtdCadastrados; i++){
            c = listaContatos[i];
            stringBuffer.append(c.toString() + "\n");
        }
        return stringBuffer.toString();
    }
}
```

Programa Principal

- Permite a criação e manipulação de uma agenda.

```
package view;

import data.Agenda;
import model.Contato;

import javax.swing.*;

public class Main {

    private static final String APP_NAME = "Agenda de contatos POO";
    private static final String OPCOES = "1 - Inserir contato\n2 - Apresentar agenda";

    public static void main(String[] args) {
        Agenda agenda;
        Contato contato;
        int i;

        i = Integer.valueOf(JOptionPane.showInputDialog(null, "Quantidade máxima de contatos: ", APP_NAME + " | Configuração
agenda", JOptionPane.QUESTION_MESSAGE));
        agenda = new Agenda(i);

        do{
            i = Integer.parseInt(JOptionPane.showInputDialog(null, "Selecione uma opção: \n" + OPCOES,
JOptionPane.QUESTION_MESSAGE));
            switch (i){
                case 1:
                    contato = readContato();
                    if(agenda.add(contato)){
                        JOptionPane.showMessageDialog(null, "Contato inserido com sucesso.", APP_NAME,
JOptionPane.INFORMATION_MESSAGE);
                    }else{
                        JOptionPane.showMessageDialog(null, "Erro ao inserir contato.", APP_NAME, JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                case 2:
                    System.out.println(agenda.getAgenda());
                    break;

                    default:
                        JOptionPane.showMessageDialog(null, "Opção inválida", APP_NAME, JOptionPane.ERROR_MESSAGE);

            }
        }while(JOptionPane.showConfirmDialog(null, "Continuar usando o sistema", APP_NAME, JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION);

    }
```

```
    public static Contato readContato(){
        Contato c;
        String nome;
        int ddd, prefixo, numero;
        nome = JOptionPane.showInputDialog(null, "Nome do contato: ", APP_NAME + " | Novo contato",
JOptionPane.QUESTION_MESSAGE);
        ddd = Integer.parseInt(JOptionPane.showInputDialog(null, "DDD: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));
        prefixo = Integer.parseInt(JOptionPane.showInputDialog(null, "Prefixo: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));
        numero = Integer.parseInt(JOptionPane.showInputDialog(null, "Numero: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));
        c = new Contato(nome, ddd, prefixo, numero);
        return c;
    }
}
```

Executem

Se passaram 10 minutos ....

- Depois da distribuição do software, nosso cliente resolve incluir no sistema um telefone celular para cada agenda.
- Atenção:
  - O usuário da versão 1 não pode deixar de usar a versão dele, apenas os novos usuários (que pagarem) poderão utilizar a nova versão.

O que muda de Telefone para TelefoneCelular?

- Celular tem um digito a mais no prefixo!!!

```
public class TelefoneCelular extends Telefone {

    public TelefoneCelular(int ddd, int prefixo, int numero) {
        super(ddd, prefixo, numero);

    }

    @Override
    public void setPrefixo(int prefixo) {
        //super.setPrefixo(prefixo);
        if(prefixo >= 8111 && prefixo <= 99999){
            this.prefixo = prefixo;
        }else{
            this.prefixo = 0;
        }
    }
}
```

Como vamos desfazer a ação de configuração do prefixo não precisamos executá-la

```
package model;

public class Contato2Fones extends Contato {
    public static final int TIPO_RESIDENCIAL = 1;
    public static final int TIPO_CELULAR = 2;

    protected TelefoneCelular celular;

    public Contato2Fones(String nome, int ddd, int prefixo, int numero, int dddCelular, int prefixoCelular, int
numeroCelular) {
        super(nome, ddd, prefixo, numero);
        this.celular = new TelefoneCelular(dddCelular, prefixoCelular, numeroCelular);
    }

    public Contato2Fones(String nome, int ddd, int prefixo, int numero) {
        super(nome, ddd, prefixo, numero);
    }

    public Contato2Fones(String nome, int ddd, int prefixo, int numero, int tipo) {
        super(nome, ddd, prefixo, numero);
        switch (tipo){
            case TIPO_RESIDENCIAL:
                celular = (TelefoneCelular) doInvado();
                break;
            case TIPO_CELULAR:
                this.telefone = doInvado();
                this.celular = new TelefoneCelular(ddd, prefixo, numero);
                break;
            default:
                this.celular = (TelefoneCelular) doInvado();
                this.telefone = doInvado();
        }

    }

    private Telefone doInvado(){
        Telefone fone = new Telefone(0, 0, 0);
        return fone;
    }

    @Override
    public String toString() {
        String pai = super.toString();
        pai = pai + "  \t Celular: " + celular.toString();
        return pai;
    }

    public void setCelular(TelefoneCelular celular) {
        this.celular = celular;
    }
}
```

Agora o Main2

Dois Main no mesmo sistema.

```
package view;

import data.Agenda;
import model.Contato2Fones;
import model.TelefoneCelular;

import javax.swing.*;

public class Main2 {

    private static final String APP_NAME = "Agenda de contatos POO";
    private static final String OPCOES = "1 - Inserir contato\n2 - Apresentar agenda";

    public static void main(String[] args) {
        Agenda agenda;
        Contato2Fones contato;
        int i;

        i = Integer.valueOf(JOptionPane.showInputDialog(null, "Quantidade máxima de contatos: ", APP_NAME + " | Configuração
agenda", JOptionPane.QUESTION_MESSAGE));
        agenda = new Agenda(i);

        do{
            i = Integer.parseInt(JOptionPane.showInputDialog(null, "Selecione uma opção: \n" + OPCOES,
JOptionPane.QUESTION_MESSAGE));
            switch (i){
                case 1:
                    contato = readContato();
                    if(agenda.add(contato)){
                        JOptionPane.showMessageDialog(null, "Contato inserido com sucesso.", APP_NAME,
JOptionPane.INFORMATION_MESSAGE);
                    }else{
                        JOptionPane.showMessageDialog(null, "Erro ao inserir contato.", APP_NAME, JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                case 2:
                    System.out.println(agenda.getAgenda());
                    break;

                default:
                    JOptionPane.showMessageDialog(null, "Opção inválida", APP_NAME, JOptionPane.ERROR_MESSAGE);

            }
        }while(JOptionPane.showConfirmDialog(null, "Continuar usando o sistema", APP_NAME, JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION);

    }
```

```
    public static Contato2Fones readContato(){
        Contato2Fones c;
        String nome;
        int ddd, prefixo, numero;
        nome = JOptionPane.showInputDialog(null, "Nome do contato: ", APP_NAME + " | Novo contato", JOptionPane.QUESTION_MESSAGE);
        ddd = Integer.parseInt(JOptionPane.showInputDialog(null, "DDD: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));
        prefixo = Integer.parseInt(JOptionPane.showInputDialog(null, "Prefixo: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));
        numero = Integer.parseInt(JOptionPane.showInputDialog(null, "Numero: ", APP_NAME + " | Novo contato | Telefone",
JOptionPane.QUESTION_MESSAGE));

        c = new Contato2Fones(nome, ddd, prefixo, numero);
        ddd = Integer.parseInt(JOptionPane.showInputDialog(null, "DDD: ", APP_NAME + " | Novo contato | Telefone Celular",
JOptionPane.QUESTION_MESSAGE));
        prefixo = Integer.parseInt(JOptionPane.showInputDialog(null, "Prefixo: ", APP_NAME + " | Novo contato | Telefone Celular",
JOptionPane.QUESTION_MESSAGE));
        numero = Integer.parseInt(JOptionPane.showInputDialog(null, "Numero: ", APP_NAME + " | Novo contato | Telefone Celular",
JOptionPane.QUESTION_MESSAGE));

        c.setCelular(new TelefoneCelular(ddd, prefixo, numero));

        return c;
    }

}
```

Se passaram outros 10 minutos ....

- Agora o cliente resolveu que o Contato pode possuir até 10 telefones, e estes podem ser do tipo FIXO ou CELULAR.
- O que fazer?

Contato10Fones

- Temos um array com 1º posições.

```java
package model;

public class Contato10Fones extends Contato{
    private final int MAXIMO_TELEFONES = 10;
    public static final int TIPO_FIXO = 1;
    public static final int TIPO_CELULAR = 2;

    protected Telefone[] listaTelefones;
    protected int telefonesCadastrados;

    public Contato10Fones(String nome) {
        super(nome, 0, 0, 0);
        listaTelefones = new Telefone[MAXIMO_TELEFONES];
        telefonesCadastrados = 0;
    }

    public boolean addTelefone(Telefone fone){
        boolean deuCerto = false;
        if(fone != null && !isFullListaTelefones()){
            if(telefonesCadastrados == 0){
                telefone = fone;
            }
            listaTelefones[telefonesCadastrados] = fone;
            telefonesCadastrados+=1;
            deuCerto = true;
        }
        return deuCerto;
    }

    public boolean addTelefone(int ddd, int prefixo, int numero, int tipo){
        boolean deuCerto = false;
        Telefone fone = null;
        if(!isFullListaTelefones()) {
            switch (tipo) {
                case TIPO_FIXO:
                    fone = new Telefone(ddd, prefixo, numero);
                    break;
                case TIPO_CELULAR:
                    fone = new TelefoneCelular(ddd, prefixo, numero);
                    break;
            }
            if(telefonesCadastrados == 0){
                telefone = fone;
            }
            listaTelefones[telefonesCadastrados] = fone;
            telefonesCadastrados+=1;
            deuCerto = true;
        }

        return deuCerto;
    }
```

```
    public Telefone getTelefone(int posicao){
        Telefone fone = null;
        if(posicao >= 0 && posicao < telefonesCadastrados){
            fone = listaTelefones[posicao];
        }
        return fone;
    }

    private boolean isFullListaTelefones(){
        return telefonesCadastrados == MAXIMO_TELEFONES;
    }

    @Override
    public String toString() {
        Telefone fone;
        StringBuffer stringBuffer = new StringBuffer();

        stringBuffer.append("Contato: " + getNome());
        stringBuffer.append("\nTelefones: ");
        for(int i=0; i < telefonesCadastrados; i++){
            fone = listaTelefones[i];
            if(fone instanceof TelefoneCelular){
                stringBuffer.append("\n\t Telefone Celular: " + fone.toString());
            }else {
                if (fone instanceof Telefone) {
                    stringBuffer.append("\n\t Telefone Fixo...: " + fone.toString());
                }
            }

        }
        return stringBuffer.toString();
    }
}
```