
Universidade Federal de Minas Gerais

**Programação Orientado a Objetos**

Trabalho Prático 2

Documentação

Davi de Lima Rolim
Lucas Mol Holmquist

3 de julho de 2018

No arquivo estão anexados dentro da pasta Operadora em src, os pacotes **excecoes**, **Interface**, **main** e **Operadora** que contém os arquivos .java referentes às classes criadas para realizar o trabalho. Nesse trabalho as informações da operadora são salvas no arquivo $DadosOperadora.txt$. Os códigos foram feitos utilizando a IDE Eclipse, na ausência dessa, para executar o programa basta abrir o arquivo **Operadora.jar** pela linha de comando, usando o comando **"java -jar Operadora.jar"**.

No código abaixo (Código 1: Função main.) pode-se observar o código no método main. Em primeiro lugar o arquivo $DadosOperadora.txt$ é lido. Caso este arquivo não exista, cria-se o arquivo e uma operadora padrão é inicializado e retornado para $op$. A interface então recebe essa operadora através do seu construtor e em seguida inicializa-se o menu, método contido na interface que interage com o usuário através do console na linha de comando. Um exemplo do menu aberto pode ser visualizado na figura 1.

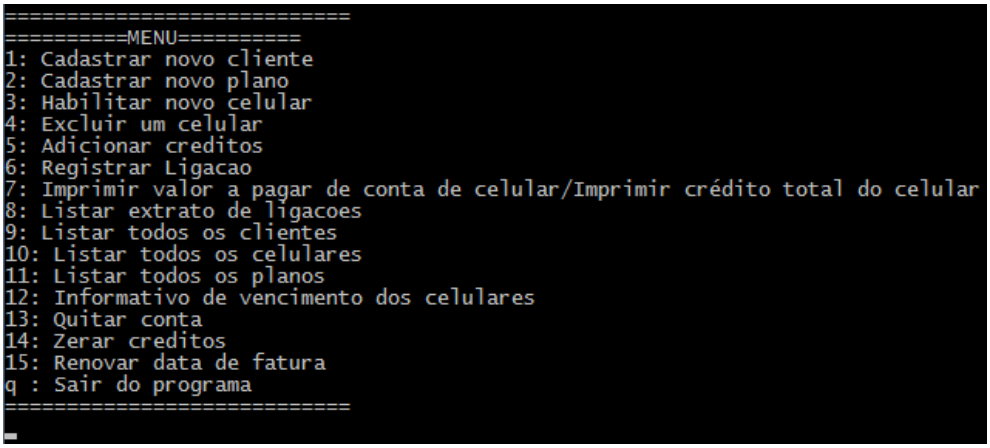


Figura 1: Menu do programa

A classe Celular contém o número do celular, o cliente que a possui e uma, e apenas uma, clase ContaCelular. A classe ContaCelular guarda todos os dados que são comuns para os dois tipos de conta: conta de assinatura ou conta de cartão. O vencimento por exemplo, apesar de ter diferentes funções, é um membro que os dois tipos de conta de celular possuem, um para o vencimento da fatura e outro para o vencimento dos créditos. Aproveitou-se dessas múltiplas características compartilhadas pelas duas contas para usar de polimorfismo, ou seja, reescrita de um mesmo método para tratamento das especificidades de cada conta, de modo que não é necessário alterar nada no código das classes filhas(Cartao e Assinatura) para retirar ou adicionar um outro tipo de conta de celular.

Na linguagem java as exceções para tratamento em tempo de execução devem ser definidas como filhas da classe Exception. A classe Exception é filha da classe Throwable e lida com tratamento de erro em tempo de execução. Ao contrário da classe Error, também filha da classe Throwable, que lida com erros no programa que não devem passar por tratamento. As classes de exceções definidas foram: ExcecaoCelular, ExcecaoPlano e ExcecaoCliente, que possuem como membro, respectivamente, um Celular, um Cliente e um Plano. De forma que quando ocorre alguma exceção relacionada, por exemplo, com uma operação envolvendo dados de cliente, joga-se ("throw") a exceção da classe ExcecaoCliente, que pode ou não conter dados do cliente que está relacionado com o problema, que é mostrado para o usuário juntamente com uma mensagem

descrevendo o problema, mensagem esta que é membra herdada da classe Exception. Abaixo se encontra o diagrama UML do projeto.

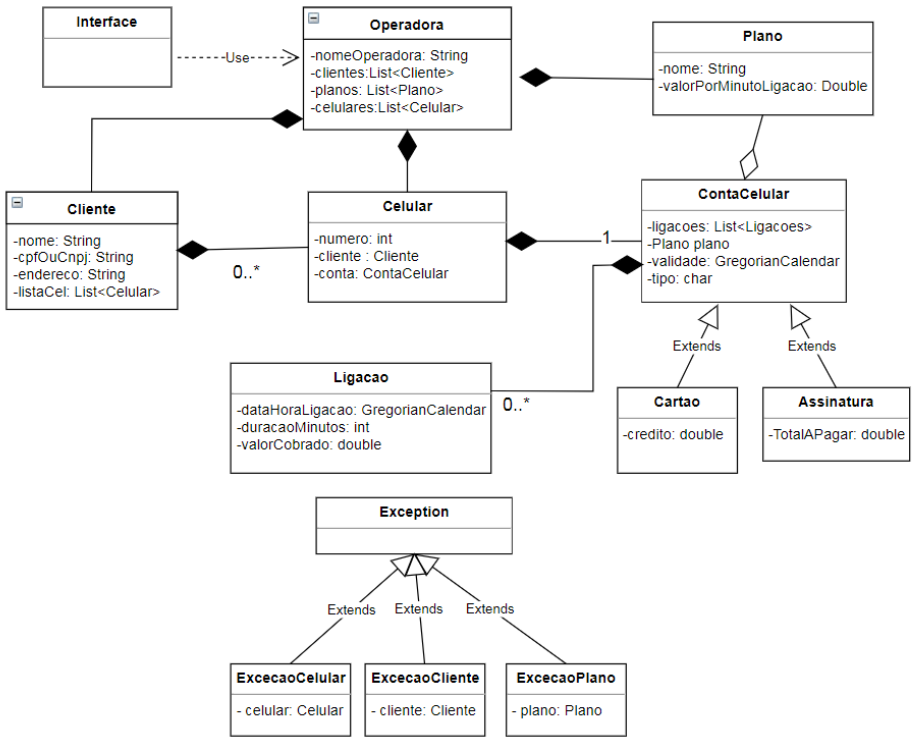


Figura 2: Diagrama UML

Listou-se abaixo os comandos disponíveis para o usuário que foram adicionados além do que foi pedido para este trabalho prático por se tratarem de funções essenciais para que o usuário possa atualizar os valores diante de pagamento dos clientes.

1. Quitar conta - Permite ao usuário quitar totalmente ou parcialmente a conta de um celular com conta do tipo assinatura. Dessa forma, caso um cliente venha a pagar a fartura, basta o usuário inserir o número do celular e quitar a fatura adequadamente. Usado apenas para celulares do tipo assinatura.

2. Zerar créditos - Permite ao usuário zerar os créditos de um celular com conta do tipo cartão. O usuário deve inserir o número de celular para indicar a conta que será zerada. Usado apenas para celulares do tipo cartão.

3. Renovar data de fatura - Permite ao usuário avançar o vencimento da fatura em um mês. Basta o usuário inserir o número do celular que se deseja renovar a data de fatura. Usado apenas para celulares do tipo assinatura.

```
1
2   public static void main(String[] args) {
3     Operadora op = Operadora.readFile();
4     if(op != null) {
5       Interface in = new Interface(op);
6       in.menu();
7     }else {
8       System.out.println("Nao foi possivel criar operadora.");
9     }
10  }
```

Código 1: Função main.

**As classes com seus respectivos atributos e métodos são escritos de forma ilustrativa(a implementação é desconsiderada) nos códigos a seguir:**

```
1  package main;
2  import Interface.Interface;
3  import Operadora.*;
4
5  public class Cliente {
6
7    private String nome;
8    private String cpf_cnpj;
9    private String endereco;
10   private ArrayList<Celular> listaCelulares;
11
12   //Construtor
13   public Cliente(String nome, String c, String end, String f)
14
15   public String getNome()
16   public void setNome(String nome)
17   public String getCpfOucnpj()
18   public void setCpfpOucnpj(String cpfOuCnpj)
19   public String getEndereco()
20   public void setEndereco(String endereco)
21   public ArrayList<Celular> getListaCel()
22   public void addCelular(Celular celular)
23   public void removerCelular(Celular celular) throws ExcecaoCelular
24   public String toString()
25 }
```

Código 2: Classe Cliente

```java
package Operadora;

public class Celular implements Serializable{

  static int proximoNumeroCelular = 999000001;

  private int numero;
  private Cliente cliente;
  private ContaCelular conta;


  public Celular(Cliente cliente, Plano plano, GregorianCalendar validade)
  public Celular(Cliente cliente, Plano plano)

  public int getNumero()
  public void setNumero(int numero)
  public Cliente getCliente()
  public void setCliente(Cliente cliente)
  public ContaCelular getConta()
  public void setConta(ContaCelular conta)
  public static void setProximoNumeroCelular(int proximoNumeroCelular)

}
```

Código 3: Classe Celular

```java
package Operadora;

import excecoes.ExcecaoCelular;

public class ContaCelular{ // Associado a um e apenas um celular

  protected ArrayList<Ligacao> ligacoes;
  protected GregorianCalendar validade;
  protected Plano plano;
  protected static char tipo;

  ContaCelular(Plano plano)
  ContaCelular(Plano plano, GregorianCalendar validade)

  public boolean checarPendencia()
  public char getTipo()
  public void registrarLigacao(GregorianCalendar dataLigacao, double duracao)
      throws ExcecaoCelular
  public ArrayList<Ligacao> extratoLigacoes(GregorianCalendar data)
  public ValorData listarValorContaCredito()
  public GregorianCalendar getValidade()
  public void setValidade(GregorianCalendar validade)
  public Plano getPlano()
  public void setPlano(Plano plano)
  public boolean checarVencimento()
}
```

Código 4: Classe ContaCelular

```java
package Operadora;

import excecoes.ExcecaoCelular;

public class ContaCartao extends ContaCelular {

  double credito;

  public ContaCartao(Plano plano)

  public boolean checarPendencia()
  public char getTipo()
  public void adicionarCredito(double creditoAdicionado, GregorianCalendar validade)
  public void registrarLigacao(GregorianCalendar dataLigacao, double duracao)
      throws ExcecaoCelular
  public ValorData listarValorContaCredito()
  public void zerarCredito()
}
```

Código 5: Classe ContaCartao

```java
package Operadora;

import excecoes.ExcecaoCelular;

public class ContaAssinatura extends ContaCelular{

  double TotalAPagar = 0;

  public ContaAssinatura(Plano plano, GregorianCalendar validade)

  public boolean checarPendencia()
  public char getTipo()
  public void registrarLigacao( GregorianCalendar dataLigacao, double duracao)
      throws ExcecaoCelular
  public double getTotalAPagar()
  public void quitarConta()
  public void quitarValorDeConta(double valor)
  public ValorData listarValorContaCredito()
  public void renovarValidadeFatura()
}
```

Código 6: Classe ContaAssinatura

```java
package Operadora;

public class Ligacao {

  private GregorianCalendar dataLigacao;
  private double duracaoMinutos;
  private double valorCobrado;

  public Ligacao()

  public Ligacao(GregorianCalendar dataLigacao, double duracaoMinutos,
      double valorCobrado)
  public GregorianCalendar getDataLigacao()
  public void setDataLigacao(GregorianCalendar dataInicioLigacao)
  public double getDuracaoMinutos()
  public void setDuracaoMinutos(double duracaoMinutos)
  public double getValorCobrado()
  public void setValorCobrado(double valorCobrado)
}
```

Código 7: Classe Ligacao

```java
package Operadora;

import excecoes.ExcecaoPlano;
import excecoes.ExcecaoCelular;
import excecoes.ExcecaoCliente;

public class Operadora implements Serializable{

  private String nomeOperadora;
  private ArrayList<Cliente> clientes;
  private ArrayList<Celular> celulares;
  private ArrayList<Plano> planos;

  public Operadora(ArrayList<Cliente> clientes, ArrayList<Celular> celulares,
      ArrayList<Plano> planos)
  public Operadora(String nomeOperadora)

  public void cadastrarCliente(String nome, String endereco, String cpfOuCnpj)
      throws ExcecaoCliente
  public void cadastrarPlano(String nome, double valorPorMinuto) throws ExcecaoPlano
  public void habilitarCelularPosPago(Cliente cliente, String nomePlano,
      GregorianCalendar vencimentoFartura) throws ExcecaoPlano, ExcecaoCliente
  public void habilitarCelularPrePago(Cliente cliente, String nomePlano)
      throws ExcecaoPlano, ExcecaoCliente
  public void excluirCelular(int numeroCelular) throws ExcecaoCelular
  public void adicionarCredito(int numeroCelular, double valor) throws ExcecaoCelular
  public void registrarLigacao(int numeroCelular, double duracao, GregorianCalendar
      dataLigacao) throws ExcecaoCelular
  public ValorData listarValorContaCredito(int numeroCelular)  throws ExcecaoCelular
  public ArrayList<Ligacao> listarExtratoLigacoes(int numeroCelular, GregorianCalendar
      data) throws ExcecaoCelular
  public ArrayList<Cliente> listarClientes()
  public ArrayList<Plano> listarPlanos()
  public ArrayList<Celular> listarCelulares()
  public ArrayList<Celular> informativoDeVencimento()
  public void quitarConta(int numeroCelular) throws ExcecaoCelular
  public void quitarParteConta(int numeroCelular, double valor) throws ExcecaoCelular
  public void zerarCreditoConta(int numeroCelular) throws ExcecaoCelular
  private Celular buscarCelular(int numeroCelular) throws ExcecaoCelular
  private Plano buscarPlano(String nomePlano) throws ExcecaoPlano
  public Cliente buscarCliente(String cpfOuCnpj) throws ExcecaoCliente
  public void renovarDataFatura(int numeroCelular) throws ExcecaoCelular
  public void escreverArquivo()
  public static Operadora lerArquivo()
  public static Operadora operadoraPreDefinida()
}
```

Código 8: Classe Operadora

```java
package Interface;

import Operadora.*;
import excecoes.*;

public class Interface {

  private Operadora operadora;
  private static boolean menuAberto = false;

  public Interface(Operadora operadora)
  public void menu()
  void cadastrarCliente() throws ExcecaoCliente
  void cadastrarPlano() throws ExcecaoPlano
  void habilitarCelular() throws ExcecaoCliente, ExcecaoPlano
  void excluirCelular() throws ExcecaoCelular
  void adicionarCreditos() throws ExcecaoCelular
  void registrarLigacao() throws ExcecaoCelular
  void listarValorContaCredito() throws ExcecaoCelular
  void extratoLigacoes()throws ExcecaoCelular
  void listarClientes()throws ExcecaoCelular
  void listarCelulares()throws ExcecaoCelular
  void listarPlanos()throws ExcecaoCelular
  void informativoVencimento()throws ExcecaoCelular
  void quitarConta()throws ExcecaoCelular
  void zerarCredito()throws ExcecaoCelular
  void renovarDataFatura() throws ExcecaoCelular
  private String DateToString(GregorianCalendar Data)
  private GregorianCalendar GetStringToGregorianCalData(String caso, String repeticao)
}
```

Código 9: Classe Interface

```java
package Operadora;

public class ValorData {
  double valor;
  GregorianCalendar data;
  char tipo;

  public ValorData()
  public ValorData(double valor, GregorianCalendar data, char tipo)
  public double getValor()
  public void setValor(double valor)
  public GregorianCalendar getData()
  public void setData(GregorianCalendar data)
  public char getTipo()
  public void setTipo(char tipo)
}
```

Código 10: Classe ValorData

```
1     public class ExcecaoCelular extends Exception{
2
3   private Celular celular;
4
5   public ExcecaoCelular() {super("Erro relacionado a dados de celular.");}
6   public ExcecaoCelular(String message) {super(message);}
7   public ExcecaoCelular(String message, Celular celular)
8   public Celular getCelular() {return celular;}
9
10 }
```

Código 11: ExcecaoCelular

```
1  package excecoes;
2
3  import Operadora.*;
4
5  public class ExcecaoCliente extends Exception {
6
7    private Cliente cliente;
8
9    public ExcecaoCliente() {super("Excecao relacionado a dados de cliente.");}
10   public ExcecaoCliente(String message) {super(message);}
11   public ExcecaoCliente(String message, Cliente cliente) {
12     super(message);
13     this.cliente = cliente;
14   }
15   public Cliente getCliente() {return cliente;}
16 }
```

Código 12: ExcecaoCliente

```
1  package excecoes;
2
3  import Operadora.*;
4
5  @SuppressWarnings("serial")
6  public class ExcecaoPlano extends Exception{
7
8    private Plano plano;
9
10   public ExcecaoPlano() { super("Excecao relacionado a dados de plano.");}
11   public ExcecaoPlano(String mensagem){super(mensagem);}
12   public ExcecaoPlano(String mensagem, Plano plano)
13   public Plano getPlano() {return plano;}
14
15 }
```

Código 13: ExcecaoPlano