|  |
| --- |
|  |
| Autoatendimento Bancário |
| Trabalho final da matéria:  ***Introdução ao Java para Android*** |

**Flávio Mitsuyoshi Tamanaha Ota - RM 47097**

**Leandro de Freitas Santos - RM47052**

**Rodrigo Tetsuya Yamashiro Ota - RM 47100**

**Wellington Sérgio Martiniano Santos - RM47086**

**10 de Agosto de 2014**

Este documento tem o objetivo descrever o projeto desenvolvido como trabalho final da matéria de Introdução ao Java para Android, ministrado por Michel Fernandes para o curso de MBA em desenvolvimento móvel e games da FIAP (5MOB).

Sumário

[Introdução 4](#_Toc395361522)

[Autoatendimento Bancário 5](#_Toc395361523)

[Logging 5](#_Toc395361524)

[Mensagens 5](#_Toc395361525)

[Configuração 5](#_Toc395361526)

[Funções e classes utilizadas: 5](#_Toc395361527)

[Design e Interface com Usuário 6](#_Toc395361528)

[Documentação 6](#_Toc395361529)

[Documentação do Código 6](#_Toc395361530)

[Documentação técnica 6](#_Toc395361531)

[Dependências 6](#_Toc395361532)

[Utilização da aplicação 6](#_Toc395361533)

[Empacotamento 7](#_Toc395361534)

[Diagrama de Classes 7](#_Toc395361535)

[Código Fonte Java 8](#_Toc395361536)

[Código Fonte Properties 20](#_Toc395361537)

[Casos de Uso 22](#_Toc395361538)

[1. Criando a conta 22](#_Toc395361539)

[2. Fazendo um Saque 22](#_Toc395361540)

[3. Fazendo um Depósito 23](#_Toc395361541)

[4. Consulta saldo 23](#_Toc395361542)

[5. Consulta extrato 24](#_Toc395361543)

[6. Cálculo de Imposto 26](#_Toc395361544)

[7. Sair 26](#_Toc395361545)

[8. Erro de digitação na conta 27](#_Toc395361546)

[9. Saldo Insuficiente 27](#_Toc395361547)

[Conclusão 29](#_Toc395361548)

## Introdução

Através deste documento que será apresentado junto com o código fonte, buscamos mostrar a nossa solução para o projeto apresentado, justificando cada escolha baseando em conceitos técnicos e experiência que serão avaliados.

Em relação às tecnologias aplicadas, procuramos utilizar a maior parte de conhecimentos ministradas em aula e que seriam viáveis para a solução do projeto proposto.

Ponto a ser considerar foi pensado de forma a ter uma usabilidade para que não seja tão complexa a sua utilização.

Mesmo que o Autoatendimento Bancário pareça ser um dos mais difíceis vimos que podemos aprender muito quando aceitamos alguns desafios.

## Autoatendimento Bancário

Dentre os projetos apresentados, escolhemos o autoatendimento bancário.

Dado que não foram colocados muitos requisitos para seu desenvolvimento, tentamos buscar um modo de fácil entendimento e que utilize as formas mais simples para que usuários não tenham problemas ao utilizar e nem se confundam.

Tivemos como inspiração as operações de banco utilizadas em celulares apenas para algumas modificações.

Dadas limitações técnicas colocadas no projeto, decidimos pela utilização do próprio console para a iteração com o usuário.

Embora nosso autoatendimento não tenha acesso ao banco de dados assim ficamos limitados ao programa com um saldo de R$ 10.000,00.

Não poderíamos nem tivemos tempo de inserir todas as funcionalidades bancárias colocamos apenas as mais simples.

## Logging

FALTA ADICIONAR UM BREVE COMENTÁRIO SOBRE LOGGING.

## Mensagens

FALTA ADICIONAR OS PRINTS COM LOGGING GERADOS.

## Configuração

## Funções e classes utilizadas:

Agora abaixo teremos uma lista do que foi utilizado para compor o nosso sistema de autoatendimento:

* java.util.InputMismatchException;
* java.util.Scanner;
* java.io.IOException;
* java.io.FileWriter;
* java.io.IOException;
* java.io.PrintWriter;
* java.text.SimpleDateFormat;
* java.util.ArrayList;
* java.util.Date.

A seguir estão as classes criadas por nossa aplicação:

* AutoAtendimento;
* ContaCorrente;
* Conta;
* Movimentacao;
* Movimento;
* SaldoInsuficienteException;
* ValorInvalidoException;
* SaldoInsuficienteException;
* Tributavel;
* UtilProperties.

## Design e Interface com Usuário

O design foi baseado em solicitações e respostas em textos que são bem explicativos com o efeito de não atrapalhar o usuário e sim auxiliado, não focamos em deixa-lo agradável aos olhos sendo que por premissa seria utilizado via console então nossos esforços foram em sua funcionalidade.

Logo dessa forma a interface com o usuário é escrita forçando o usuário a ler sendo o máximo de usabilidade possível para uma aplicação de nível console.

## Documentação

### Documentação do Código

A documentação do código foi elaborada baseada em Javadoc e pode ser acessada diretamente da pasta doc/javadoc contida dentro do projeto de Autoatendimento Báncario.

### Documentação técnica

## Dependências

Para construir o aplicativo utilizamos o JDK versão 7 (Java Development Kit) para desenvolvê-lo então para compila-lo e testa-lo aconselhamos que utilizem o JDK versão 7.

## Utilização da aplicação

A aplicação recebe a inserção de valores como seu tratamento está sendo feito de maneira bem simples qualquer digitação fora do padrão causará erro e reiniciará a aplicação abaixo algumas anotações sobre a aplicação:

* Inserção do nome será apenas o primeiro.
* Os dados conta, agencia, senha, valor deve ser numéricos.
* O imposto da conta corresponde a 1% do seu valor corrente.

## Empacotamento

Utilizamos os pacotes com uma nomenclatura de fácil entendimento:

* ***br.com.fiap.sistema****:* possui a classe main da execução;
* ***br.com.fiap.sistema.conta****:* possui as classes que criam a conta e fazem sua movimentação;
* ***br.com.fiap.sistema.exceptions****:*  possui as classes que tratam os erros possíveis no Autoatendimento;
* ***br.com.fiap.sistema.intefaces****:* possui a Interface do calculo do Imposto;
* ***br.com.fiap.sistema.util****:*possui a classe UtilProperties.java , que acessa o arquivo “autoatendimentoBancario.properties”, nas pasta properties.

## Diagrama de Classes

Abaixo temos uma representação da estrutura e relações das classes que servem de modelo para objetos.

|  |
| --- |
| C:\DESENVOLVIMENTO\METADADOS\JAVA\PROJETO_JAVA\Documentos - Java\PRINT_DIAGRAMA_ATIVIDADE_FINAL.png |

## Código Fonte Java

Código fonte do projeto, apresentado por classes organizadas conforme indicado no item “Empacotamento”.

|  |
| --- |
| Classe: AutoAtendimento.java |
| package br**.**com**.**fiap**.**sistema**;**  **import** java**.**io**.**IOException**;**  **import** java**.**util**.**InputMismatchException**;**  **import** java**.**util**.**Properties**;**  **import** java**.**util**.**Scanner**;**  **import** br**.**com**.**fiap**.**sistema**.**conta**.**ContaCorrente**;**  **import** br**.**com**.**fiap**.**sistema**.**exceptions**.**SaldoInsuficienteException**;**  **import** br**.**com**.**fiap**.**sistema**.**exceptions**.**ValorInvalidoException**;**  **import** br**.**com**.**fiap**.**sistema**.**util**.**UtilProperties**;**  public class AutoAtendimento **{**  /\*\*  \* Método main é responsável por inicializar a aplicação  \*  \* **@param** args  \* **@throws** IOException  \*/  public static void main**(**String**[]** args**)** **throws** IOException **{**  /\*  \* Chamada do método estático UtilProperties.getProp() para implementar  \* mensagens parametrizadas em arquivo ".properties"  \*/  Properties prop **=** UtilProperties**.**getProp**();**  boolean continuar **=** **true;**  double valor **=** 0**;**  **try** **{**  /\*  \* Implementação da classe Scanner, que converte textos para tipos primitivos,  \* sendo que esses textos podem ser considerados como objetos do tipo String,  \* InputStream e arquivos.  \*/  Scanner s **=** **new** Scanner**(**System**.**in**);**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.bemVindo"**));**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.criandoNovaConta"**));**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.informeSeuNome"**));**  String nome **=** s**.**next**();**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.agencia"**));**  int agencia **=** s**.**nextInt**();**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.conta"**));**  int numero **=** s**.**nextInt**();**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.senha"**));**  int senha **=** s**.**nextInt**();**  /\*  \* Chamada da Classe ContaCorrente, que extende a classe Conta  \* que implementa a interface Tributavel  \*/  ContaCorrente conta **=** **new** ContaCorrente**(**numero**,** agencia**,** senha**,** nome**);**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.ola"**)** **+** conta**.**getNome**()+**  prop**.**getProperty**(**"prop.atendimento.premiadoSaldoConta"**));**  **do** **{**  System**.**out**.**println**(**  prop**.**getProperty**(**"prop.atendimento.selecioneOperacao"**)** **+**  prop**.**getProperty**(**"prop.atendimento.saque"**)** **+**  prop**.**getProperty**(**"prop.atendimento.deposito"**)** **+**  prop**.**getProperty**(**"prop.atendimento.consultaSaldo"**)** **+**  prop**.**getProperty**(**"prop.atendimento.consultaExtrato"**)** **+**  prop**.**getProperty**(**"prop.atendimento.calculoImposto"**)** **+**  prop**.**getProperty**(**"prop.atendimento.sair"**)**  **);**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.operacao"**));**  int operacao **=** s**.**nextInt**();**  /\*  \* Utilizado a função switch/case, que verifica a variável  \* enviada, e direciona para a opção correspondente.  \*  \* Para esta operação foi utilizada as opções de 1 á 6 (  \* 1: Saque, 2: Depósito, 3: Consulta de Saldo,  \* 4: Consulta de Extrato, 5: Cálculo de Imposto, 0: Sair),  \* sendo a opção default, a de tratamento de erro (opção inválida).  \*/  **try** **{**  **switch(**operacao**){**  **case** 1**:**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.informeValorSaque"**));**  valor **=** s**.**nextDouble**();**  conta**.**saca**(**valor**);**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.saqueEfetuado"**));**  **break;**  **case** 2**:**  System**.**out**.**print**((**prop**.**getProperty**(**"prop.atendimento.informeValorDeposito"**)));**  valor **=** s**.**nextDouble**();**  conta**.**deposita**(**valor**);**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.depositoEfetuado"**));**  **break;**  **case** 3**:**  System**.**out**.**printf**(**prop**.**getProperty**(**"prop.atendimento.saldoReais"**),** conta**.**getSaldo**());**  **break;**  **case** 4**:**  boolean continuar3 **=** **true;**  **do** **{**  System**.**out**.**println**(**  prop**.**getProperty**(**"prop.atendimento.escolhaFormaImpressao"**)** **+**  prop**.**getProperty**(**"prop.atendimento.tela"**)** **+**  prop**.**getProperty**(**"prop.atendimento.arquivo"**)** **+**  prop**.**getProperty**(**"prop.atendimento.cancelar"**)**  **);**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.opcao"**));**  operacao **=** s**.**nextInt**();**  /\*  \* Utilizado a função switch/case, para selecionar o  \* modo de visualização do extrato.  \*  \* Para esta operação foi utilizada as opções de 1 á 3 (  \* 1: Tela, 2: Arquivo, 3: Cancelar)  \* sendo a opção default, a de tratamento de erro (opção inválida).  \*/  **switch** **(**operacao**)** **{**  **case** 1**:**  continuar3 **=** **false;**  conta**.**emiteExtratoTela**();**  **break;**  **case** 2**:**  continuar3 **=** **false;**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.gerandoArquivo"**));**  **try** **{**  conta**.**emiteExtratoArquivo**();**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.geradoSucesso"**));**  **}** **catch** **(**IOException e**)** **{**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.erroGeracaoArquivo"**));**  **}**  **break;**  **case** 3**:**  continuar3 **=** **false;**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.consultaCancelada"**));**  **break;**  **default:**  // Mensagem de erro para valor de opção inválido.  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.opcaoInvalida"**));**  **break;**  **}**  **}** **while** **(**continuar3**);**  **break;**  **case** 5**:**  System**.**out**.**printf**(**prop**.**getProperty**(**"prop.atendimento.impostoConta"**),** conta**.**calculaTributos**());**  **break;**  **case** 0**:**  continuar **=** **false;**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.sessaoFinalizada"**));**  **break;**  **default:**  // Mensagem de erro para valor de opção inválido.  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.operacaoInvalida"**));**  **break;**  **}**  **}** **catch** **(**ValorInvalidoException e**)** **{**  // Mensagem de erro para envio de valores inválidos.  System**.**out**.**println**(**e**.**getMessage**());**  **}** **catch** **(**SaldoInsuficienteException e**)** **{**  // Mensagem de erro para saldo insuficiente na conta.  System**.**out**.**println**(**e**.**getMessage**());**  **}**  **if** **(**continuar**){**  boolean continuar2 **=** **true;**  **do** **{**  System**.**out**.**println**(**  prop**.**getProperty**(**"prop.atendimento.desejaRealizaOutraOperacao"**)** **+**  prop**.**getProperty**(**"prop.atendimento.sim"**)** **+**  prop**.**getProperty**(**"prop.atendimento.nao"**)**  **);**  System**.**out**.**print**(**prop**.**getProperty**(**"prop.atendimento.opcao"**));**  operacao **=** s**.**nextInt**();**  /\*  \* Utilizado a função switch/case, para selecionar  \* o cancelamento da visualização do extrato.  \*  \* Para esta operação foi utilizada as opções de 1 á 3 (  \* 1: Tela, 2: Arquivo, 3: Cancelar)  \* sendo a opção default, a de tratamento de erro (opção inválida).  \*/  **switch** **(**operacao**)** **{**  **case** 1**:**  continuar2 **=** **false;**  **break;**  **case** 2**:**  continuar **=** **false;**  continuar2 **=** **false;**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.sessaoFinalizada"**));**  **break;**  **default:**  // Mensagem de erro para valor de opção inválido.  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.operacaoInvalida"**));**  **break;**  **}**  **}** **while** **(**continuar2**);**  **}**  **}** **while** **(**continuar**);**  s**.**close**();**  **}** **catch** **(**InputMismatchException e**)** **{**  // Mensagem de erro para digitação de valores inválidos.  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.erroDigitacao"**));**  main**(**args**);**  **}** **catch** **(**Exception e**)** **{**  e**.**printStackTrace**();**  // Mensagem de erro de sistema.  System**.**out**.**println**(**prop**.**getProperty**(**"prop.atendimento.erroSistema"**));**  main**(**args**);**  **}**  **}**  **}** |

|  |
| --- |
| Classe: Conta.java |
| package br**.**com**.**fiap**.**sistema**.**conta**;**  **import** java**.**io**.**FileWriter**;**  **import** java**.**io**.**IOException**;**  **import** java**.**io**.**PrintWriter**;**  **import** java**.**text**.**SimpleDateFormat**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Date**;**  **import** java**.**util**.**Properties**;**  **import** br**.**com**.**fiap**.**sistema**.**exceptions**.**SaldoInsuficienteException**;**  **import** br**.**com**.**fiap**.**sistema**.**exceptions**.**ValorInvalidoException**;**  **import** br**.**com**.**fiap**.**sistema**.**util**.**UtilProperties**;**  public abstract class Conta**{**  // Saldo incial da conta no valor de R$ 10.000,00.  protected double saldo **=** 10000**;**  private int numero**;**  private int agencia**;**  private int senha**;**  private String nome**;**  private ArrayList**<**Movimentacao**>** movimentacoes **=** **new** ArrayList**<**Movimentacao**>();**  /\*\*  \* Construtor da classe Conta.  \*  \* **@param** numero  \* **@param** agencia  \* **@param** senha  \* **@param** nome  \*/  public Conta**(**int numero**,** int agencia**,** int senha**,** String nome**)** **{**  **super();**  **this.**numero **=** numero**;**  **this.**agencia **=** agencia**;**  **this.**senha **=** senha**;**  **this.**nome **=** nome**;**  **}**  /\*\*  \* Método modificador do atributo nome.  \*  \* **@param** nome  \*/  public void setNome**(**String nome**)** **{**  **this.**nome **=** nome**;**  **}**  /\*\*  \* Método de consulta do atributo nome.  \*  \* **@return** nome  \*/  public String getNome**()** **{**  **return** nome**;**  **}**  /\*\*  \* Método de consulta do atributo numero.  \*  \* **@return** numero  \*/  public int getNumero**()** **{**  **return** numero**;**  **}**  /\*\*  \* Método para alterar o atributo numero.  \*  \* **@param** numero  \*/  public void setNumero**(**int numero**)** **{**  **this.**numero **=** numero**;**  **}**  /\*\*  \* Método de consulta do atributo agencia.  \*  \* **@return** agencia  \*/  public int getAgencia**()** **{**  **return** agencia**;**  **}**  /\*\*  \* Método para alterar o atributo agencia.  \*  \* **@param** agencia  \*/  public void setAgencia**(**int agencia**)** **{**  **this.**agencia **=** agencia**;**  **}**  /\*\*  \* Método de consulta do atributo senha.  \*  \* **@return** senha  \*/  public int getSenha**()** **{**  **return** senha**;**  **}**  /\*\*  \* Método para alterar o atributo senha.  \*  \* **@param** senha  \*/  public void setSenha**(**int senha**)** **{**  **this.**senha **=** senha**;**  **}**  /\*\*  \* Método de consulta do atributo saldo.  \*  \* **@return**  \*/  public double getSaldo**()** **{**  **return** saldo**;**  **}**  /\*\*  \* Método para o tratamento do valor de saque.  \*  \* **@param** valor  \*/  public void saca**(**double valor**)** **{**  **if** **(**valor **>** **this.**saldo**)** **{**  **throw** **new** SaldoInsuficienteException**(**valor**);**  **}** **else** **{**  **this.**saldo **-=** valor**;**  Movimentacao m **=** **new** Movimentacao**(new** Date**(),** valor**,** **this.**saldo**,** Movimento**.**DEBITO**);**  addMovimentacao**(**m**);**  **}**  **}**  /\*\*  \* Método para o tratamento do valor de depósito.  \*  \* **@param** valor  \* **@throws** ValorInvalidoException  \*/  public void deposita**(**double valor**)** **throws** ValorInvalidoException **{**  **if** **(**valor **<** 0**)** **{**  **throw** **new** ValorInvalidoException**(**valor**);**  **}** **else** **{**  **this.**saldo **+=** valor**;**  Movimentacao m **=** **new** Movimentacao**(new** Date**(),** valor**,** **this.**saldo**,** Movimento**.**CREDITO**);**  addMovimentacao**(**m**);**  **}**  **}**  /\*\*  \* Método para o adiciona uma nova  \* movimentação no fluxo de visualização do extrato.  \*  \* **@param** m  \*/  public void addMovimentacao**(**Movimentacao m**){**  **this.**movimentacoes**.**add**(**m**);**  **}**  /\*\*  \* Método para visualização de extrato na tela.  \*  \* **@throws** IOException  \*/  public void emiteExtratoTela**()** **throws** IOException**{**  /\*  \* Chamada do método estático UtilProperties.getProp() para implementar  \* mensagens parametrizadas em arquivo ".properties"  \*/  Properties prop **=** UtilProperties**.**getProp**();**  /\*  \* Classe SimpleDateFormat implementada para formatar a data/hora.  \*/  SimpleDateFormat formatador **=** **new** SimpleDateFormat**(**"dd/MM/yyyy"**);**  SimpleDateFormat formatador2 **=** **new** SimpleDateFormat**(**"dd/MM/yyyy - HH:mm:ss"**);**  Date d **=** **new** Date**();**  // cabeçalho  System**.**out**.**printf**(**"\n%45s\n"**,**prop**.**getProperty**(**"prop.conta.cabecalho.titulo"**));**  System**.**out**.**printf**(**"%-39s%21s\n"**,**prop**.**getProperty**(**"prop.conta.cabecalho.extratoContaCorrente"**),**formatador2**.**format**(**d**));**  System**.**out**.**printf**(**  "%-20s%-20s%.20s \n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.agencia"**)** **+** **this.**agencia**,**  prop**.**getProperty**(**"prop.conta.cabecalho.conta"**)** **+** **this.**numero**,**  prop**.**getProperty**(**"prop.conta.cabecalho.cliente"**)** **+** **this.**nome  **);**  System**.**out**.**printf**(**"%-15s%-17s%-14s%14s\n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.data"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.descricao"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.valor"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.saldo"**)**  **);**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.conta.layout.barra"**));**  /\*  \* Implementado a função ArrayList (List), com finalidade de exibir todos arranjo  \* de movimentações realizadas pelo cliente.  \*/  **for** **(**Movimentacao m **:** **this.**movimentacoes**)** **{**  System**.**out**.**printf**(**"%-15s%-17s%-14.2f%14.2f \n"**,**formatador**.**format**(**m**.**getData**()),** m**.**getMovimento**().**getDescricao**(),** m**.**getValor**(),** m**.**getSaldo**());**  **}**  System**.**out**.**println**(**prop**.**getProperty**(**"prop.conta.layout.barra"**));**  System**.**out**.**printf**(**"%-40s%20.2f \n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.saldoFinal"**),this.**saldo**);**  **}**  /\*\*  \* Método para emissão de extrato em arquivo do tipo ".txt"  \*  \* **@throws** IOException  \*/  public void emiteExtratoArquivo**()** **throws** IOException**{**  /\*  \* Chamada do método estático UtilProperties.getProp() para implementar  \* mensagens parametrizadas em arquivo ".properties"  \*/  Properties prop **=** UtilProperties**.**getProp**();**  /\*  \* Classe SimpleDateFormat implementada para formatar a data/hora.  \*/  SimpleDateFormat formatador **=** **new** SimpleDateFormat**(**"dd/MM/yyyy"**);**  SimpleDateFormat formatador2 **=** **new** SimpleDateFormat**(**"dd/MM/yyyy - HH:mm:ss"**);**  Date d **=** **new** Date**();**  /\*\*  \* Classe FileWriter implementada para inserir fluxos de caracteres  \* em arquivos, exemplo ".txt".  \*/    FileWriter fw **=** **new** FileWriter**(**"extrato.txt"**);**  PrintWriter out **=** **new** PrintWriter**(**fw**);**  out**.**printf**(**"%45s \n"**,**"Sistema de Auto-Atendimento"**);**  out**.**printf**(**"%-39s%21s \n"**,**"Extrato de Conta Corrente"**,**formatador2**.**format**(**d**));**  out**.**printf**(**"%-20s%-20s%.20s \n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.agencia"**)** **+** **this.**agencia**,**  prop**.**getProperty**(**"prop.conta.cabecalho.conta"**)** **+** **this.**numero**,**  prop**.**getProperty**(**"prop.conta.cabecalho.cliente"**)** **+** **this.**nome  **);**  out**.**printf**(**"%-15s%-17s%-14s%14s\n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.data"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.descricao"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.valor"**),**  prop**.**getProperty**(**"prop.conta.cabecalho.saldo"**)**  **);**  out**.**println**(**prop**.**getProperty**(**"prop.conta.layout.barra"**));**    /\*  \* Implementado a função ArrayList (List), com finalidade de exibir todos arranjo  \* de movimentações realizadas pelo cliente.  \*/  **for** **(**Movimentacao m **:** **this.**movimentacoes**)** **{**  out**.**printf**(**"%-15s%-17s%-14.2f%14.2f \n"**,**formatador**.**format**(**m**.**getData**()),** m**.**getMovimento**().**getDescricao**(),** m**.**getValor**(),** m**.**getSaldo**());**  **}**  out**.**println**(**prop**.**getProperty**(**"prop.conta.layout.barra"**));**  out**.**printf**(**"%-40s%20.2f \n\n"**,**  prop**.**getProperty**(**"prop.conta.cabecalho.saldoFinal"**),**1000.004**);**  out**.**close**();**  **}**  **}** |

|  |
| --- |
| Classe: ContaCorrente.java |
| package br**.**com**.**fiap**.**sistema**.**conta**;**  **import** br**.**com**.**fiap**.**sistema**.**interfaces**.**Tributavel**;**  public class ContaCorrente **extends** Conta **implements** Tributavel **{**  /\*\*  \* Construtor da classe Conta Corrente.  \*  \* **@param** numero  \* **@param** agencia  \* **@param** senha  \* **@param** nome  \*/  public ContaCorrente**(**int numero**,** int agencia**,** int senha**,** String nome**)** **{**  **super(**numero**,** agencia**,** senha**,** nome**);**  **}**  /\*\*  \* Calcula o valor de 1% de tributo.  \*/  @Override  public double calculaTributos**()** **{**  **return** **this.**getSaldo**()** **\*** 0.01**;**  **}**  **}** |

|  |
| --- |
| Classe: Movimentacao.java |
| package br**.**com**.**fiap**.**sistema**.**conta**;**  **import** java**.**util**.**Date**;**  public class Movimentacao **{**  private Date data**;**  private double valor**;**  private double saldo**;**  private Movimento movimento**;**  /\*\*  \* Construtor da classe Movimentacao.  \*  \* **@param** data  \* **@param** valor  \* **@param** saldo  \* **@param** movimento  \*/  public Movimentacao**(**Date data**,** double valor**,** double saldo**,**  Movimento movimento**)** **{**  **this.**data **=** data**;**  **this.**valor **=** valor**;**  **this.**saldo **=** saldo**;**  **this.**movimento **=** movimento**;**  **}**  /\*\*  \* Método de consulta do atributo data.  \*  \* **@return**  \*/  public Date getData**()** **{**  **return** data**;**  **}**  /\*\*  \* Método para alterar o atributo data.  \*  \* **@param** data  \*/  public void setData**(**Date data**)** **{**  **this.**data **=** data**;**  **}**  /\*\*  \* Método de consulta do atributo valor.  \*  \* **@return** valor  \*/  public double getValor**()** **{**  **return** valor**;**  **}**  /\*\*  \* Método para alterar o atributo valor.  \*  \* **@param** valor  \*/  public void setValor**(**double valor**)** **{**  **this.**valor **=** valor**;**  **}**  /\*\*  \* Método de consulta do atributo saldo.  \*  \* **@return** saldo  \*/  public double getSaldo**()** **{**  **return** saldo**;**  **}**  /\*\*  \* Método para alterar o atributo saldo.  \*  \* **@param** saldo  \*/  public void setSaldo**(**double saldo**)** **{**  **this.**saldo **=** saldo**;**  **}**  /\*\*  \* Método de consulta da classe Movimento.  \*  \* **@return**  \*/  public Movimento getMovimento**()** **{**  **return** movimento**;**  **}**  /\*\*  \* Método para alterar o atributo movimento.  \*  \* **@param** movimento  \*/  public void setMovimento**(**Movimento movimento**)** **{**  **this.**movimento **=** movimento**;**  **}**  **}** |

|  |
| --- |
| Classe: Movimento.java |
| package br**.**com**.**fiap**.**sistema**.**conta**;**  /\*  \* Instância do tipo enum.  \* Utilizada para uma lista constante dos itens CREDITO  \* e DEBITO, com o seus respectivos códigose e descrições.  \*  \*/  public enum Movimento **{**  CREDITO**(**"Depósito"**,** 1**),** DEBITO**(**"Saque "**,** 2**);**  private String descricao**;**  private int codigo**;**  /\*\*  \* Método Movimento, que altera os atributos  \* descrição e codigo  \*  \* **@param** descricao  \* **@param** codigo  \*/  Movimento**(**String descricao**,** int codigo**)** **{**  **this.**descricao **=** descricao**;**  **this.**codigo **=** codigo**;**  **}**  /\*\*  \* Método de consulta do atributo codigo.  \*  \* **@return** codigo  \*/  public int getCodigo**()** **{**  **return** codigo**;**  **}**  /\*\*  \* Método de consulta do atributo descricao.  \*  \* **@return** descricao  \*/  public String getDescricao**()** **{**  **return** descricao**;**  **}**  **}** |

|  |
| --- |
| Classe: SaldoInsuficienteException.java |
| package br**.**com**.**fiap**.**sistema**.**exceptions**;**  public class SaldoInsuficienteException **extends** RuntimeException**{**  /\*\*  \*  \*/  private static final long serialVersionUID **=** **-**4968563189450809791L**;**  /\*\*  \*  \* **@param** valor  \*/  public SaldoInsuficienteException**(**double valor**)** **{**  **super(**"Saldo insuficiente na conta. Operação cancelada. \n"**);**  **}**  **}** |

|  |
| --- |
| Classe: ValorInvalidoException.java |
| package br**.**com**.**fiap**.**sistema**.**exceptions**;**  public class ValorInvalidoException **extends** RuntimeException**{**  /\*\*  \*  \* **@param** valor  \*/  public ValorInvalidoException**(**double valor**)** **{**  **super(**"Valor inválido: " **+** valor **+** ". Operação cancelada. \n"**);**  **}**  private static final long serialVersionUID **=** 1L**;**  **}** |

|  |
| --- |
| Classe: Tributavel.java |
| package br**.**com**.**fiap**.**sistema**.**interfaces**;**  /\*  \* Implementação da interface para o cálculo de tributo  \*/  public interface Tributavel **{**  double calculaTributos**();**  **}** |

|  |
| --- |
| Classe: UtilProperties.java |
| package br**.**com**.**fiap**.**sistema**.**util**;**  **import** java**.**io**.**FileInputStream**;**  **import** java**.**io**.**IOException**;**  **import** java**.**util**.**Properties**;**  public class UtilProperties **{**    /\*\*  \* Método getProp() é resposável por acessar o arquivo de extensão  \* ".properties"  \*  \* **@return**  \* **@throws** IOException  \*/  public static Properties getProp**()** **throws** IOException **{**  Properties props **=** **new** Properties**();**  FileInputStream file **=** **new** FileInputStream**(**  "./properties/autoatendimentoBancario.properties"**);**  props**.**load**(**file**);**  **return** props**;**  **}**  **}** |

## Código Fonte Properties

Arquivo com a extensão “.properties”, com a lista de todas as mensagens da aplicação, sendo representados com a estrutura de “chave = valor”, para serem acessadas em tempo de execução.

|  |
| --- |
| Properties: autoatendimentoBancario.properties |
| #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#  #Arquivo com todas as mensagens parametrizada  prop.conta.cabecalho.titulo = Sistema de Auto-Atendimento  prop.conta.cabecalho.extratoContaCorrente = Extrato de Conta Corrente  prop.conta.cabecalho.agencia = Agência:  prop.conta.cabecalho.conta = Conta:  prop.conta.cabecalho.cliente = Cliente:  prop.conta.cabecalho.data = Data  prop.conta.cabecalho.descricao = Descrição  prop.conta.cabecalho.valor = Valor  prop.conta.cabecalho.saldo = Saldo  prop.conta.cabecalho.saldoFinal = Saldo Final  prop.conta.layout.barra = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  prop.atendimento.bemVindo = Bem vindo ao sistema de Auto-Atendimento  prop.atendimento.criandoNovaConta = Criando uma nova conta..  prop.atendimento.informeSeuNome = \nInforme seu nome:  prop.atendimento.agencia = Agência:  prop.atendimento.conta = Conta:  prop.atendimento.senha = Senha:  prop.atendimento.ola = \nOlá,  prop.atendimento.premiadoSaldoConta = . Você foi premiado com um saldo de R$ 10.000,00 na sua conta!  prop.atendimento.senha = Senha:  prop.atendimento.selecioneOperacao = \nSelecione a operação desejada:  prop.atendimento.saque = \n 1: Saque  prop.atendimento.deposito = \n 2: Depósito  prop.atendimento.consultaSaldo = \n 3: Consulta de Saldo  prop.atendimento.consultaExtrato = \n 4: Consulta de Extrato  prop.atendimento.calculoImposto = \n 5: Cálculo de Imposto  prop.atendimento.sair = \n 0: Sair  prop.atendimento.operacao = Operação:  prop.atendimento.informeValorSaque = \nInforme o valor do saque:  prop.atendimento.saqueEfetuado = Saque efetuado com sucesso!  prop.atendimento.informeValorDeposito = \nInforme o valor do depósito:  prop.atendimento.depositoEfetuado = Depósito efetuado com sucesso!  prop.atendimento.saldoReais = \nSaldo: R$ %.2f \n  prop.atendimento.escolhaFormaImpressao = \nEscolha a forma de impressão:  prop.atendimento.tela = \n 1: Tela  prop.atendimento.arquivo = \n 2: Arquivo  prop.atendimento.cancelar = \n 3: Cancelar  prop.atendimento.opcao = Opção:  prop.atendimento.gerandoArquivo = \nGerando arquivo...  prop.atendimento.geradoSucesso = \nArquivo gerado com sucesso.  prop.atendimento.erroGeracaoArquivo = \nErro na geração do arquivo  prop.atendimento.consultaCancelada = \nConsulta cancelada.  prop.atendimento.opcaoInvalida = \nOpção inválida. Tente Novamente.  prop.atendimento.impostoConta = \nImposto da conta: R$ %.2f \n  prop.atendimento.sessaoFinalizada = \nSessão finalizada.  prop.atendimento.operacaoInvalida = \nOperação inválida.  prop.atendimento.desejaRealizaOutraOperacao = \nDeseja realizar outra operação?  prop.atendimento.sim = \n 1: Sim  prop.atendimento.nao = \n 2: Não  prop.atendimento.erroDigitacao = \nErro de digitação. Tente novamente.  prop.atendimento.erroSistema = \nErro no sistema. Tente novamente.  #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# |

## Casos de Uso

Segue abaixo a descrição dos fluxos das funções do projeto, sendo representadas por narrativas textuais.

|  |
| --- |
| Criando a conta |
| Pré-requisitos: iniciar a aplicação |
| * Usuário informa seu nome; * Informa o número da agência; * Informa o número da conta; * Informa sua senha; * Sistema Informa que usuário foi premiado e possui um saldo de 10.000 reais; * E pede para que se insira a operação; |
|  |

|  |
| --- |
| Fazendo um Saque |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| * Usuário informa a operação 1; * Informa o valor do saque; * Sistema informa que o saque foi feito com sucesso; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
|  |

|  |
| --- |
| Fazendo um Depósito |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| * Usuário informa a operação 2; * Informa o valor do deposito; * Sistema informa que o deposito foi feito com sucesso; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
|  |

|  |
| --- |
| Consulta saldo |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| * Usuário informa a operação 3; * Sistema informa que o saldo da conta; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
|  |

|  |
| --- |
| Consulta extrato |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| * Usuário informa a operação 4; * Informa o tipo de impressão 1 na tela 2 arquivo e 3 cancela; * Caso o usuário digite 1 o extrato aparecerá na tela; * Caso o usuário digite 2 será gerado um extrato em arquivo; * Caso o usuário digite 3 sera cancelada a consulta; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
| Em tela |
| Em arquivo |
|  |
| Cancelado |

|  |
| --- |
| Cálculo de Imposto |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| * Usuário informa a operação 5; * Sistema informa que o imposto da conta; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
|  |

|  |
| --- |
| Sair |
| Pré-requisitos: Ter executado pelo menos o Caso de uso 1 |
| Usuário informa a operação 0;  Sistema encerra a aplicação; |
|  |

|  |
| --- |
| Erro de digitação na conta |
| Pré-requisitos: Nenhum |
| * Usuário informa seu nome; * Informa o número da agência; * Informa o número da conta; * Informa sua senha; * Sistema Informa que ouve erro na digitação e reinicia o sistema. |
|  |

|  |
| --- |
| Saldo Insuficiente |
| Pré-requisitos: Nenhum |
| * Usuário informa a operação 1; * Informa o valor do saque acima do que possui em conta; * Sistema informa que o saque foi feito com sucesso; * Sistema pergunta se o usuário quer fazer mais alguma operação; * Usuário pode inserir 1 para fazer mais operações ou inserir 2 para sair; |
|  |

## Conclusão

Concluímos que se aprende realmente quando começamos a exercitar, esse trabalho nos fez utilizar conceitos que em grande maioria já utilizaríamos em java trabalhando, mas que tirando como uma base para o android será muito útil para nossas futuras aplicações.

O autoatendimento bancário foi um desafio a ser superado que em equipe e com o conhecimento adquirido e relembrado pelas aulas se tornou menos complicado.