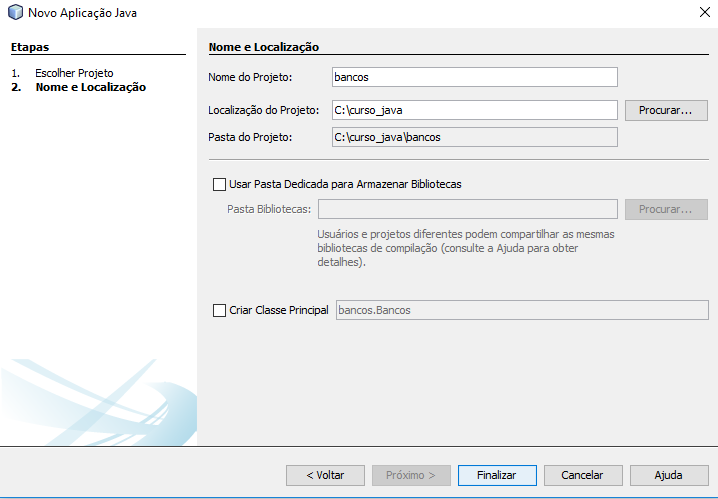
**Curso de Java - POO**

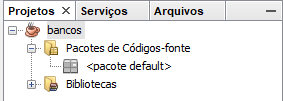
**Carlos Emilio Padilla Severo**

**Vídeos**: https://www.youtube.com/watch?v=oH6rDvPaAew&list=PLqmCwMNmP1Ix0MMUoBHsCY7Fyzk2kotbg

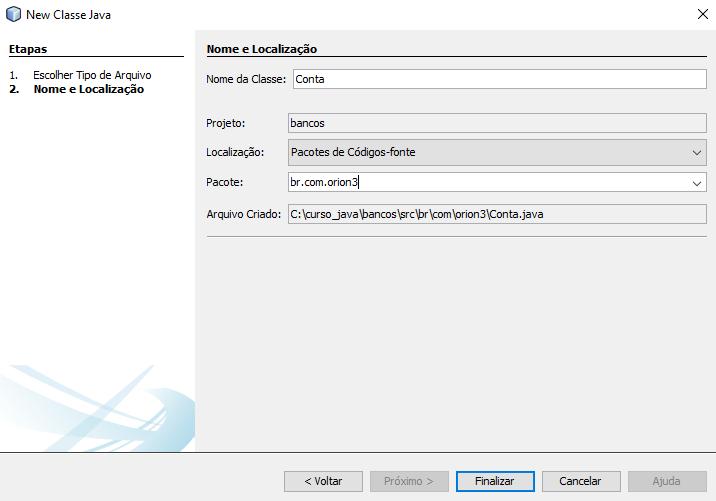
# Aula 01 - Classes, construtores e métodos de acesso



- Desmarcar a caixa Criar Classe Principal



- Crie a classe Conta



**Conta.java**

package br.com.orion3;

public class Conta {

private int numero;

private String correntista;

private float saldo;

public Conta() {

}

public Conta(int numero, String correntista, float valor) {

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public String getCorrentista() {

return correntista;

}

public void setCorrentista(String correntista) {

this.correntista = correntista;

}

public float getSaldo() {

return saldo;

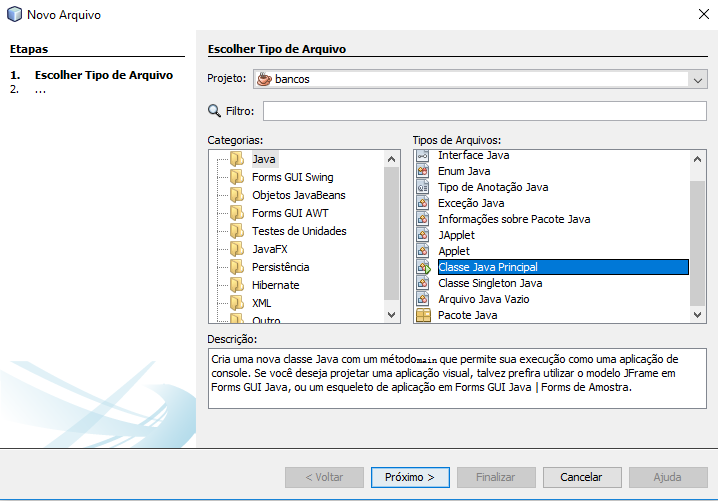
}

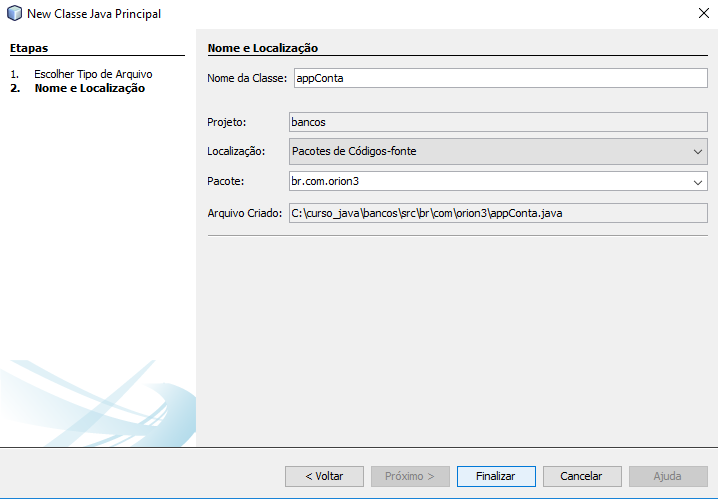
public void setSaldo(float saldo) {

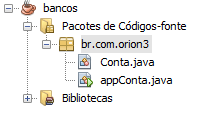
this.saldo = saldo;

}

}







**appConta.java**

package br.com.orion3;

public class appConta {

public static void main(String[] args) {

Conta c1;

c1 = new Conta();

c1.setNumero(123456);

c1.setCorrentista("André Pereira");

c1.setSaldo(150.00f);

System.out.println("=================================================");

System.out.println("Número da conta: ............. " + c1.getNumero());

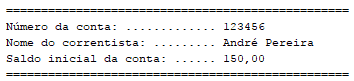
System.out.println("Nome do correntista: ......... " + c1.getCorrentista());

System.out.printf("Saldo inicial da conta: ...... %.2f\n", c1.getSaldo());

System.out.println("=================================================");

}

}



# Aula 02 - Sobrecarga de métodos e associação entre classes

**Pessoa.java**

package br.com.orion3;

public class Pessoa {

private String nome;

private String email;

public Pessoa() {

}

public Pessoa(String nome, String email) {

this.nome = nome;

this.email = email;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**Conta.java**

package br.com.orion3;

public class Conta {

private int numero;

private Pessoa correntista;

private float saldo;

public Conta() {

}

public Conta(int numero, Pessoa correntista, float valor) {

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public Pessoa getCorrentista() {

return correntista;

}

public void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public float getSaldo() {

return saldo;

}

public void setSaldo(float saldo) {

this.saldo = saldo;

}

}

**appConta.java**

package br.com.orion3;

public class appConta {

public static void main(String[] args) {

Conta c1, c2;

Pessoa p1, p2;

c1 = new Conta();

p1 = new Pessoa("Cristiane Barbosa", "cristiane\_barbosa@gmail.com");

c1.setNumero(123457);

c1.setCorrentista(p1);

c1.setSaldo(150.00f);

p2 = new Pessoa();

c2 = new Conta();

p2.setNome("André Pereira");

p2.setEmail("andre\_pereira@gmail.com");

c2.setNumero(123456);

c2.setCorrentista(p2);

c2.setSaldo(750.00f);

System.out.println("=================================================");

System.out.println("Número da conta: ............. " + c1.getNumero());

System.out.println("Nome do correntista: ......... " + c1.getCorrentista().getNome());

System.out.printf("Saldo inicial da conta: ...... %.2f\n", c1.getSaldo());

System.out.println("=================================================");

System.out.println("Número da conta: ............. " + c2.getNumero());

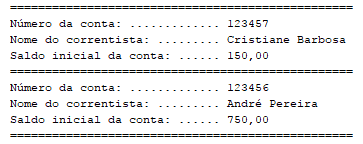
System.out.println("Nome do correntista: ......... " + c2.getCorrentista().getNome());

System.out.printf("Saldo inicial da conta: ...... %.2f\n", c2.getSaldo());

System.out.println("=================================================");

}

}



# Aula 03 - Criando operações para movimentação de contas

**Conta.java**

package br.com.orion3;

public class Conta {

private int numero;

private Pessoa correntista;

private float saldo;

public Conta() {

}

public Conta(int numero, Pessoa correntista, float valor) {

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public Pessoa getCorrentista() {

return correntista;

}

public void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public float getSaldo() {

return saldo;

}

public void setSaldo(float saldo) {

this.saldo = saldo;

}

public void depositar(float valor){

this.setSaldo(this.getSaldo() + valor);

}

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

**appConta.java**

package br.com.orion3;

public class appConta {

public static void main(String[] args) {

Conta c1, c2;

Pessoa p1, p2;

c1 = new Conta();

p1 = new Pessoa("Cristiane Barbosa", "cristiane\_barbosa@gmail.com");

c1.setNumero(123457);

c1.setCorrentista(p1);

c1.setSaldo(150.00f);

p2 = new Pessoa();

c2 = new Conta();

p2.setNome("André Pereira");

p2.setEmail("andre\_pereira@gmail.com");

c2.setNumero(123456);

c2.setCorrentista(p2);

c2.setSaldo(750.00f);

if(c1.sacar(300.00f)){

System.out.println("Saque efetuado com sucesso!");

System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

} else {

System.out.println("Saque insuficiente!\n");

}

if(c1.sacar(100.00f)){

System.out.println("Saque efetuado com sucesso!");

System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

} else {

System.out.println("Saque insuficiente!\n");

}

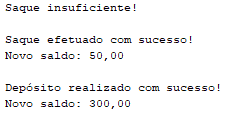
c1.depositar(250.00f);

System.out.println("Depósito realizado com sucesso!");

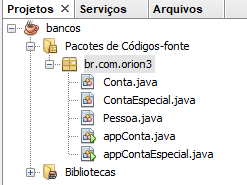
System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

}

}



# Aula 04 - Conceito de especialização e polimorfismo



**ContaEspecial.java**

package br.com.orion3;

public class ContaEspecial extends Conta {

private float limite;

public ContaEspecial(){

}

public ContaEspecial(int numero, Pessoa correntista, float valor, float limite){

super(numero, correntista, valor);

this.setLimite(limite);

}

public float getLimite() {

return limite;

}

public void setLimite(float limite){

this.limite = limite;

}

@Override

public boolean sacar(float valor) {

if(this.getSaldo() - valor >= this.getLimite()) {

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

**appContaEspecial.java**

package br.com.orion3;

public class appContaEspecial {

public static void main(String[] args) {

ContaEspecial e1;

Pessoa p1 = new Pessoa("Bruno Teixeira", "bruno\_teixeira@hotmail.com");

e1 = new ContaEspecial(123458, p1, 0.00f, -1000.00f);

System.out.printf("Saldo inicial da conta: %.2f\n", e1.getSaldo());

System.out.printf("Limite: %.2f\n\n", e1.getLimite());

if(e1.sacar(1200.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

if(e1.sacar(800.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

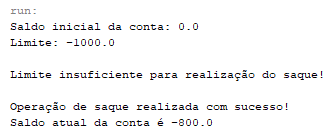
} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

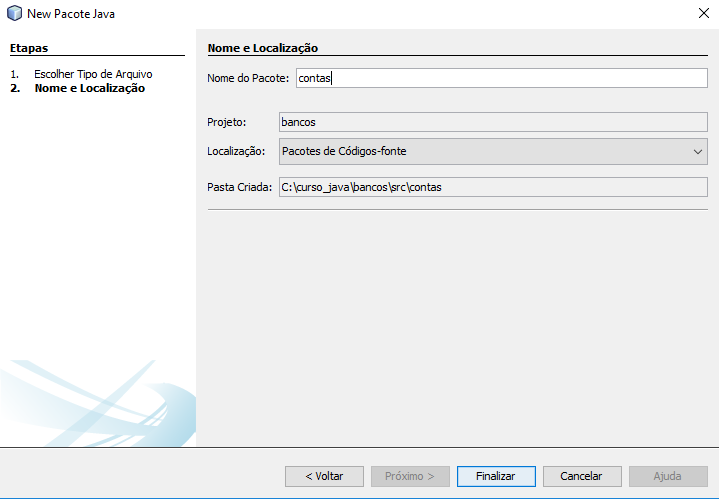
}

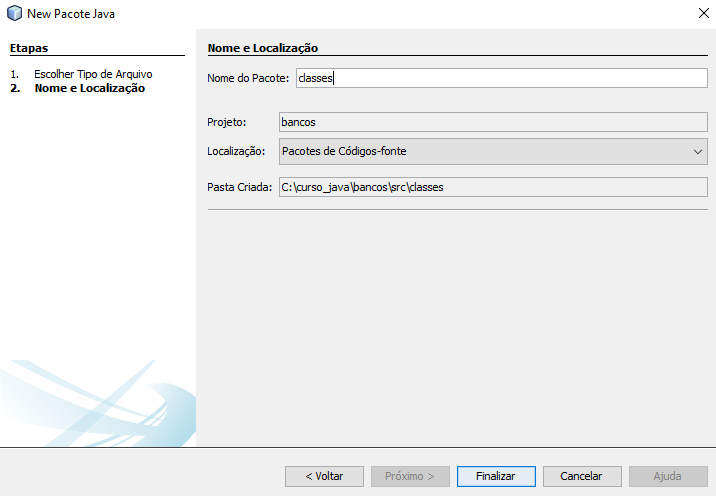
}

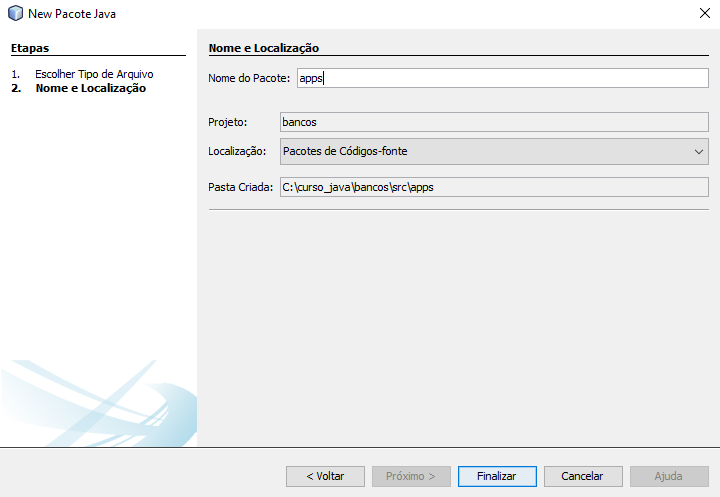


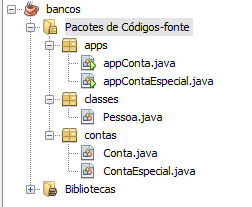
# Aula 05 - Encapsulamento, escopo de atributos e pacotes

## Inserindo pacotes e remanejando classes









**contas\Conta.java**

package contas;

import classes.Pessoa;

public class Conta {

protected int numero;

protected Pessoa correntista;

protected float saldo;

public Conta() { }

public Conta(int numero, Pessoa correntista, float valor) {

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public Pessoa getCorrentista() {

return correntista;

}

public void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public float getSaldo() {

return saldo;

}

public void setSaldo(float saldo) {

this.saldo = saldo;

}

public void depositar(float valor){

this.setSaldo(this.getSaldo() + valor);

}

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

**apps\appConta.java**

package apps;

import classes.Pessoa;

import contas.Conta;

public class appConta {

public static void main(String[] args) {

Conta c1, c2;

Pessoa p1, p2;

c1 = new Conta();

p1 = new Pessoa("Cristiane Barbosa", "cristiane\_barbosa@gmail.com");

c1.setNumero(123457);

c1.setCorrentista(p1);

c1.setSaldo(150.00f);

p2 = new Pessoa();

c2 = new Conta();

p2.setNome("André Pereira");

p2.setEmail("andre\_pereira@gmail.com");

c2.setNumero(123456);

c2.setCorrentista(p2);

c2.setSaldo(750.00f);

if(c1.sacar(300.00f)){

System.out.println("Saque efetuado com sucesso!");

System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

} else {

System.out.println("Saque insuficiente!\n");

}

if(c1.sacar(100.00f)){

System.out.println("Saque efetuado com sucesso!");

System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

} else {

System.out.println("Saque insuficiente!\n");

}

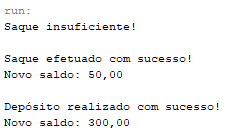
c1.depositar(250.00f);

System.out.println("Depósito realizado com sucesso!");

System.out.printf("Novo saldo: %.2f\n\n", c1.getSaldo());

}

}



**contas\ContaEspecial.java**

package contas;

import classes.Pessoa;

import contas.Conta;

public class ContaEspecial extends Conta {

private float limite;

public ContaEspecial(){

}

public ContaEspecial(int numero, Pessoa correntista, float valor, float limite){

super(numero, correntista, valor);

this.setLimite(limite);

}

public float getLimite() {

return limite;

}

public void setLimite(float limite){

this.limite = limite;

}

@Override

public boolean sacar(float valor) {

if(this.getSaldo() - valor >= this.getLimite()) {

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

**apps\appContaEspecial.java**

package apps;

import classes.Pessoa;

import contas.ContaEspecial;

public class appContaEspecial {

public static void main(String[] args) {

ContaEspecial e1;

Pessoa p1 = new Pessoa("Bruno Teixeira", "bruno\_teixeira@hotmail.com");

e1 = new ContaEspecial(123458, p1, 0.00f, -1000.00f);

System.out.printf("Saldo inicial da conta: %.2f\n", e1.getSaldo());

System.out.printf("Limite: %.2f\n\n", e1.getLimite());

if(e1.sacar(1200.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

if(e1.sacar(800.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

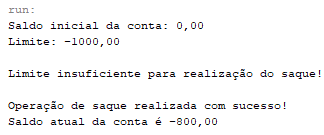
} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

}

}



**classes\Pessoa.java**

package classes;

public class Pessoa {

private String nome;

private String email;

public Pessoa() {

}

public Pessoa(String nome, String email) {

this.nome = nome;

this.email = email;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getEmail() {

return email;

}

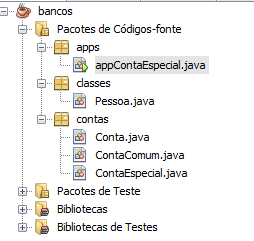
public void setEmail(String email) {

this.email = email;

}

}

# Aula 06 - Uso de abstract e final



**contas\Conta.java**

package contas;

import classes.Pessoa;

public abstract class Conta {

protected int numero;

protected Pessoa correntista;

protected float saldo;

public Conta() {

}

public Conta(int numero, Pessoa correntista, float valor) {

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public final int getNumero() {

return numero;

}

public final void setNumero(int numero) {

this.numero = numero;

}

public final Pessoa getCorrentista() {

return correntista;

}

public final void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public final float getSaldo() {

return saldo;

}

public final void setSaldo(float saldo) {

this.saldo = saldo;

}

public final void depositar(float valor){ // método concreto

this.setSaldo(this.getSaldo() + valor);

}

public abstract void sacar(float valor); // método polimórfico

}

**contas\ContaComum.java**

package contas;

import classes.Pessoa;

public final class ContaComum extends Conta {

public ContaComum() {

}

public ContaComum(int numero, Pessoa correntista, float valor) {

super(numero, correntista, valor);

}

@Override

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

**contas\ContaEspecial.java**

package contas;

import classes.Pessoa;

import contas.Conta;

public final class ContaEspecial extends Conta {

private float limite;

public ContaEspecial(){

}

public ContaEspecial(int numero, Pessoa correntista, float valor, float limite){

super(numero, correntista, valor);

this.setLimite(limite);

}

public float getLimite() {

return limite;

}

public void setLimite(float limite){

this.limite = limite;

}

@Override

public boolean sacar(float valor) {

if(this.getSaldo() - valor >= this.getLimite()) {

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

# Aula 07 - Herança múltipla, o conceito de interface

- Em interfaces somente podemos ter métodos abstratos (sem assinatura e bloco de código ou corpo) e atributos.

**contas\ ContaPoupanca.java**

package contas;

import classes.Pessoa;

public class ContaPoupanca extends Conta {

public ContaPoupanca() {

}

public ContaPoupanca(int numero, Pessoa correntista, float valor) {

super(numero, correntista, valor);

}

@Override

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

}

## Interface

**contas\Taxas.java**

package contas;

public interface Taxas {

float getTaxaManutencao();

void descontarTaxaManutencao();

}

- Mova a classe Taxas.java da pasta contas para a pasta classes

**classes\Taxas.java**

package classes;

public interface Taxas {

float getTaxaManutencao();

void descontarTaxaManutencao();

}

**contas\ContaComum.java**

package contas;

import classes.Pessoa;

public final class ContaComum extends Conta implements Taxas {

public ContaComum() {

}

public ContaComum(int numero, Pessoa correntista, float valor) {

super(numero, correntista, valor);

}

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

@Override

public float getTaxaManutencao(){

return 15.00f;

}

@Override

public void descontarTaxaManutencao(){

this.setSaldo(this.getSaldo() - this.getTaxaManutencao());

}

}

**contas\ContaEspecial.java**

package contas;

import classes.Pessoa;

import contas.Conta;

public final class ContaEspecial extends Conta implements Taxas {

private float limite;

public ContaEspecial(){

}

public ContaEspecial(int numero, Pessoa correntista, float valor, float limite){

super(numero, correntista, valor);

this.setLimite(limite);

}

public float getLimite() {

return limite;

}

public void setLimite(float limite){

this.limite = limite;

}

public boolean sacar(float valor){

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

return true;

} else {

return false;

}

}

@Override

public float getTaxaManutencao(){

return 15.00f;

}

@Override

public void descontarTaxaManutencao(){

this.setSaldo(this.getSaldo() - this.getTaxaManutencao());

}

}

**apps\appContaEspecial.java**

package apps;

import classes.Pessoa;

import contas.ContaEspecial;

public class appContaEspecial {

public static void main(String[] args) {

ContaEspecial e1;

Pessoa p1 = new Pessoa("Bruno Teixeira", "bruno\_teixeira@hotmail.com");

e1 = new ContaEspecial(123458, p1, 0.00f, -1000.00f);

System.out.printf("Saldo inicial da conta: %.2f\n", e1.getSaldo());

System.out.printf("Limite: %.2f\n\n", e1.getLimite());

if(e1.sacar(1200.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

if(e1.sacar(800.00f)){

System.out.println("Operação de saque realizada com sucesso!");

System.out.printf("Saldo atual da conta é %.2f\n\n", e1.getSaldo());

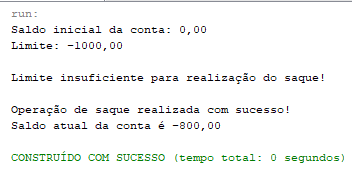
} else {

System.out.println("Limite insuficiente para realização do saque!\n");

}

}

}



# Aula 08 - Atributos e métodos de classe, uso de static

**contas\Conta.java**

package contas;

import classes.Pessoa;

public abstract class Conta {

// atributos de instância

protected int numero;

protected Pessoa correntista;

protected float saldo;

// atributo de classe

private static int numero\_contas;

public Conta() {

incrementa\_contas();

}

public Conta(int numero, Pessoa correntista, float valor) {

this();

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public final int getNumero() {

return numero;

}

public final void setNumero(int numero) {

this.numero = numero;

}

public final Pessoa getCorrentista() {

return correntista;

}

public final void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public final float getSaldo() {

return saldo;

}

public final void setSaldo(float saldo) {

this.saldo = saldo;

}

public int getNumeroContas(){

return this.numero\_contas;

}

public final void depositar(float valor){ // método concreto

this.setSaldo(this.getSaldo() + valor);

}

public abstract boolean sacar(float valor); // método polimórfico

private static void incrementa\_contas(){

numero\_contas++;

}

}

**apps\appContas.java**

package apps;

import classes.Pessoa;

import contas.ContaComum;

import contas.ContaEspecial;

public class appContas {

public static void main(String[] args) {

Pessoa p1 = new Pessoa("Fernando Borges", "fernando\_borgessgmail.com");

ContaComum c1 = new ContaComum(123456, p1, 150.00f);

System.out.println("Número de contas instanciadas até o momento: " + c1.getNumeroContas());

ContaEspecial e1 = new ContaEspecial(145362, p1, 2500.00f, 5500.00f);

System.out.println("Número de contas instanciadas até o momento: " + e1.getNumeroContas());

}

}



# Aula 09 - Atributos de valor constante, uso de final

**contas\Conta.java**

package contas;

import classes.Pessoa;

public abstract class Conta {

// atributos de instância

protected int numero;

protected Pessoa correntista;

protected float saldo;

public final static int SACAR = 0;

public final static int DEPOSITAR = 1;

// atributo de classe

private static int numero\_contas;

public Conta() {

incrementa\_contas();

}

public Conta(int numero, Pessoa correntista, float valor) {

this();

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public final int getNumero() {

return numero;

}

public final void setNumero(int numero) {

this.numero = numero;

}

public final Pessoa getCorrentista() {

return correntista;

}

public final void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public final float getSaldo() {

return saldo;

}

public final void setSaldo(float saldo) {

this.saldo = saldo;

}

public int getNumeroContas(){

return this.numero\_contas;

}

public final void depositar(float valor){ // método concreto

this.setSaldo(this.getSaldo() + valor);

}

public abstract boolean sacar(float valor); // método polimórfico

public boolean movimentar(float valor, int operacao){

boolean retorno = true;

switch(operacao){

case DEPOSITAR:

this.depositar(valor);

break;

case SACAR:

retorno = this.sacar(valor);

break;

default:

retorno = false;

}

return retorno;

}

private static void incrementa\_contas(){

numero\_contas++;

}

}

**apps\appContas.java**

package apps;

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

import contas.ContaEspecial;

public class appContas {

public static void main(String[] args) {

Pessoa p1 = new Pessoa("Fernando Borges", "fernando\_borgessgmail.com");

ContaComum c1 = new ContaComum(123456, p1, 150.00f);

System.out.println("Número de contas instanciadas até o momento: " + c1.getNumeroContas());

ContaEspecial e1 = new ContaEspecial(145362, p1, 2500.00f, 5500.00f);

System.out.println("Número de contas instanciadas até o momento: " + e1.getNumeroContas());

System.out.println("");

if(c1.sacar(1000.00f)){

System.out.println("Saque efetuado com sucesso!");

} else {

System.out.println("Saldo insuficiente!");

}

if(c1.movimentar(1000.00f, Conta.SACAR)){

System.out.println("Saque efetuado com sucesso!");

} else {

System.out.println("Saldo insuficiente!");

}

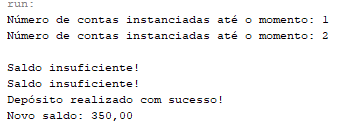
c1.movimentar(200.00f, Conta.DEPOSITAR);

System.out.println("Depósito realizado com sucesso!");

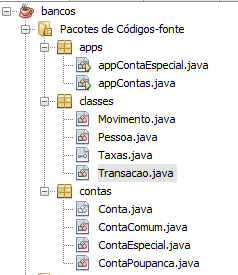
System.out.printf("Novo saldo: %.2f \n\n", c1.getSaldo());

}

}



# Aula 10 - Aplicação de agregação e composição



**classes\Movimento.java**

package classes;

import contas.Conta;

import java.util.Date;

// agregação

public class Movimento {

private Date data;

private Conta conta;

private String historico;

private float valor;

private float saldoanterior;

private int operacao;

public Movimento(Date data, Conta conta, String historico, float valor, int operacao) {

this.data = data;

this.conta = conta;

this.historico = historico;

this.valor = valor;

this.operacao = operacao;

}

public Date getData() {

return data;

}

public void setData(Date data) {

this.data = data;

}

public Conta getConta() {

return conta;

}

public void setConta(Conta conta) {

this.conta = conta;

}

public String getHistorico() {

return historico;

}

public void setHistorico(String historico) {

this.historico = historico;

}

public float getValor() {

return valor;

}

public void setValor(float valor) {

this.valor = valor;

}

public float getSaldoanterior() {

return saldoanterior;

}

public void setSaldoanterior(float saldoanterior) {

this.saldoanterior = saldoanterior;

}

public int getOperacao() {

return operacao;

}

public void setOperacao(int operacao) {

this.operacao = operacao;

}

public boolean movimentar(){

this.saldoanterior = conta.getSaldo();

if(operacao == Conta.SACAR)

this.conta.movimentar(this.valor, Conta.SACAR);

else if(operacao == Conta.DEPOSITAR) {

this.conta.movimentar(this.valor, Conta.DEPOSITAR);

return true;

}

return false;

}

}

**classes\Transacao.java**

package classes;

import contas.Conta;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

// composição transação/movimento (relação mais forte que agregação)

public class Transacao {

private List<Movimento> movimentos;

public Transacao(){

movimentos = new ArrayList<>();

}

// Efetiva uma movimentação na conta

public boolean realizarTTransacao(Date data, Conta conta, float valor, String historico, int operacao){

Movimento movimento = new Movimento(data, conta, historico, valor, operacao);

if(!movimento.movimentar()){

return false;

} else {

this.movimentos.add(movimento);

return true;

}

}

public void extornaTransacao(){

for(Movimento movimento: this.movimentos){

movimento = null;

}

}

public List<Movimento> getMovimentos(){

return this.movimentos;

}

}

# Aula 11 - Criando uma aplicação que realiza transações

**apps\appTransacoes.java**

package apps;

import classes.Movimento;

import classes.Pessoa;

import classes.Transacao;

import contas.Conta;

import contas.ContaComum;

import java.text.SimpleDateFormat;

import java.util.Date;

public class appTransacoes {

public static void main(String[] args) {

Transacao transacoes = new Transacao();

Pessoa pessoa = new Pessoa("Adalberto Ribeiro", "adalberto\_ribeiro@gmail.com");

Conta contacomum = new ContaComum(102030, pessoa, 450.00f);

float saldoanterior = contacomum.getSaldo();

transacoes.realizarTTransacao(new Date(), contacomum, 100.00f, "Depósito em dinheiro.", Conta.DEPOSITAR);

transacoes.realizarTTransacao(new Date(), contacomum, 50.00f, "Pagamento conta luz", Conta.SACAR);

transacoes.realizarTTransacao(new Date(), contacomum, 120.00f, "Pagamento conta telefone", Conta.SACAR);

transacoes.realizarTTransacao(new Date(), contacomum, 850.00f, "Transferência entre contas.", Conta.DEPOSITAR);

// Exibindo o extrato da conta

System.out.println("Emitindo extrato da conta comum número " + contacomum.getNumero());

System.out.println("Correntista: " + contacomum.getCorrentista().getNome());

System.out.println("Saldo anterior: " + saldoanterior);

System.out.println("==============================================");

for(Movimento movimento: transacoes.getMovimentos()){

System.out.println("Data: " + new SimpleDateFormat("dd/MM/yyyy").format(movimento.getData()));

System.out.println("Histórico: " + movimento.getHistorico());

System.out.printf("Valor: %.2f \n", movimento.getValor());

System.out.println("Movimento: " + (movimento.getOperacao() == Conta.DEPOSITAR ? "Depósito" : "Saque"));

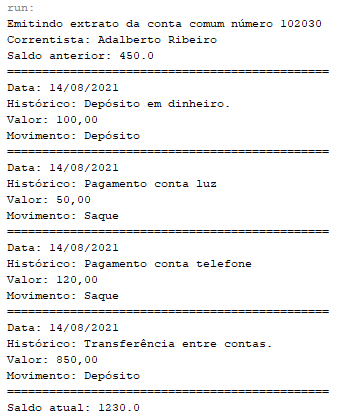
System.out.println("==============================================");

}

System.out.println("Saldo atual: " + contacomum.getSaldo() + "\n");

}

}



# Aula 12 - Classe Object e seus métodos

**apps\appContas.java**

package apps;

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

import contas.ContaEspecial;

public class appContas {

public static void main(String[] args) {

Pessoa p1 = new Pessoa("Fernando Borges", "fernando\_borgessgmail.com");

ContaComum c1 = new ContaComum(123456, p1, 150.00f);

System.out.println("Número de contas instanciadas até o momento: " + c1.getNumeroContas());

ContaEspecial e1 = new ContaEspecial(145362, p1, 2500.00f, 5500.00f);

System.out.println("Número de contas instanciadas até o momento: " + e1.getNumeroContas());

System.out.println("");

if(c1.sacar(1000.00f)){

System.out.println("Saque efetuado com sucesso!");

} else {

System.out.println("Saldo insuficiente!");

}

if(c1.movimentar(1000.00f, Conta.SACAR)){

System.out.println("Saque efetuado com sucesso!");

} else {

System.out.println("Saldo insuficiente!");

}

c1.movimentar(200.00f, Conta.DEPOSITAR);

System.out.println("Depósito realizado com sucesso!");

System.out.printf("Novo saldo: %.2f \n\n", c1.getSaldo());

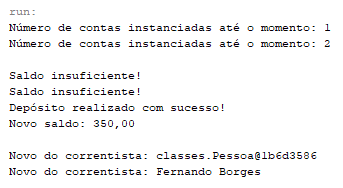
System.out.println("Novo do correntista: " + c1.getCorrentista());

System.out.println("Novo do correntista: " + c1.getCorrentista().getNome());

System.out.println("");

}

}



## Usando conceitos da classe Object

**classes\Pessoas.java**

package classes;

public class Pessoa {

private String nome;

private String email;

public Pessoa() {

}

public Pessoa(String nome, String email) {

this.nome = nome;

this.email = email;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

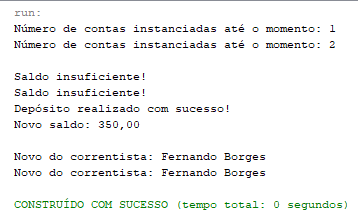
public String toString() {

return this.nome;

}

}

Agora, rodando appContas.java teremos:



# Aula 13 - Documentação com javadoc

- Veremos como documentar o código de uma determinada classe usando a ferramenta javadoc

**contas\Conta.java**

package contas;

import classes.Pessoa;

/\*\*

\* A classe <strong>Conta</strong> define um tipo de dado abstrato para a criação de estrutura de classes de contas bancárias

\* @author Roberto Pinheiro

\* @since ago/2021

\* @version 1.0

\*/

public abstract class Conta {

// atributos de instância

/\*\*

\* O atributo número é utilizado para referenciar o número da conta.

\*/

protected int numero;

/\*\*

\* O atributo correntista, do tipo <b>Pessoa</b> é utilizado para referenciar um correntista.

\*/

protected Pessoa correntista;

/\*\*

\* O atributo saldo é utilizado para referenciar a saída da conta.

\*/

protected float saldo;

// atributo de classe

/\*\*

\* Constante que define a operação de saque.

\*/

public final static int SACAR = 0;

/\*\*

\* Constante que define a operação de depósito.

\*/

public final static int DEPOSITAR = 1;

private static int numero\_contas;

/\*\*

\* construtor default da classe <b>Conta</b>

\* <b>Uso: </b><br>

\* Conta conta = new ContaComum();

\*/

public Conta() {

incrementa\_contas();

}

/\*\*

\* construtor sobrecarregado da classe <b>Conta</b>

\* <b>Uso: </b><br>

\* Conta conta = new ContaComum(102374, new Pessoa("Fulano", "fulano@gmail.com"), 150.00f)<br>;

\* <b>Onde:</b><br>

\* @param numero inteiro que identifica o número da conta.

\* @param correntista objeto do tipo <b>Pessoa</b> que identifica o correntista da conta

\* @param valor float que identifica o saldo inicial da conta

\*/

public Conta(int numero, Pessoa correntista, float valor) {

this();

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public final int getNumero() {

return numero;

}

public final void setNumero(int numero) {

this.numero = numero;

}

public final Pessoa getCorrentista() {

return correntista;

}

public final void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public final float getSaldo() {

return saldo;

}

public final void setSaldo(float saldo) {

this.saldo = saldo;

}

public int getNumeroContas(){

return this.numero\_contas;

}

public final void depositar(float valor){ // método concreto

this.setSaldo(this.getSaldo() + valor);

}

public abstract boolean sacar(float valor); // método polimórfico

public boolean movimentar(float valor, int operacao){

boolean retorno = true;

switch(operacao){

case DEPOSITAR:

this.depositar(valor);

break;

case SACAR:

retorno = this.sacar(valor);

break;

default:

retorno = false;

}

return retorno;

}

private static void incrementa\_contas(){

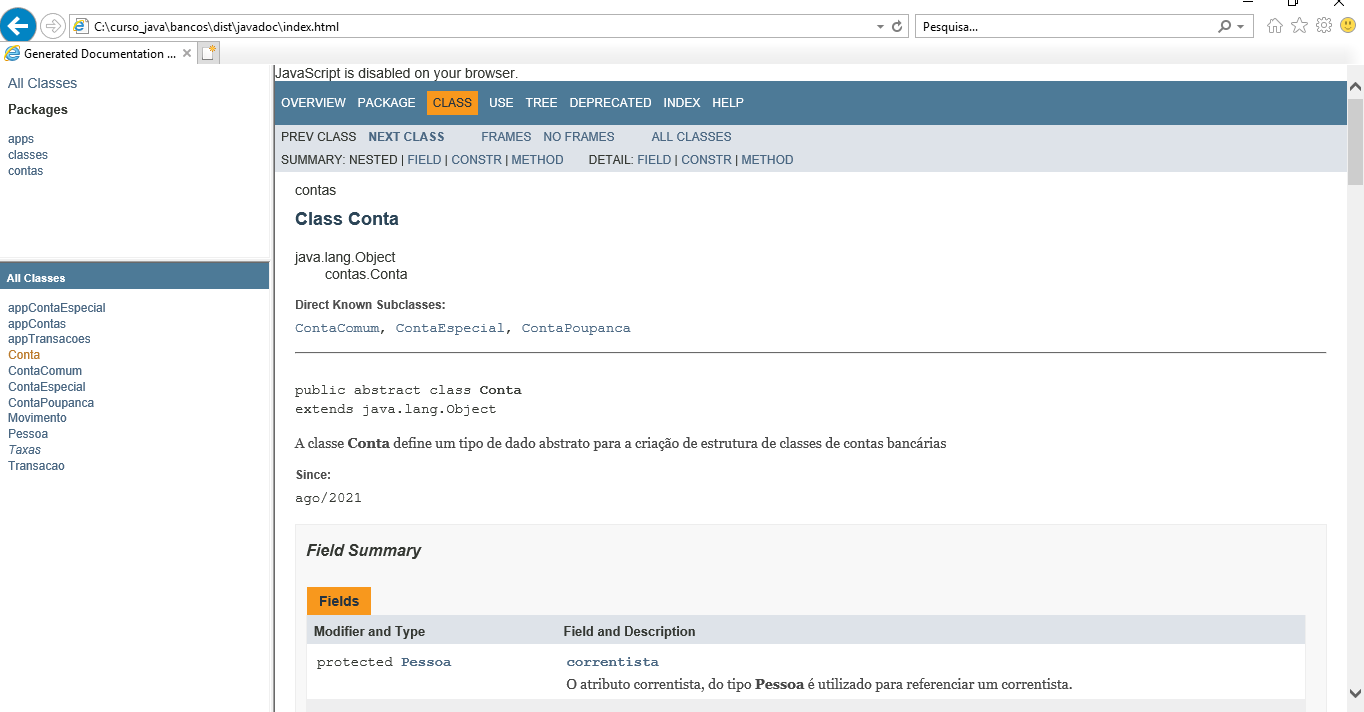
numero\_contas++;

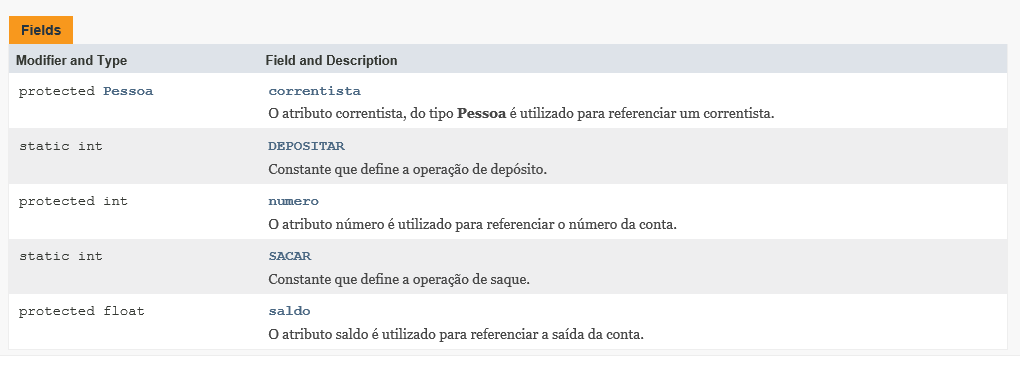
}

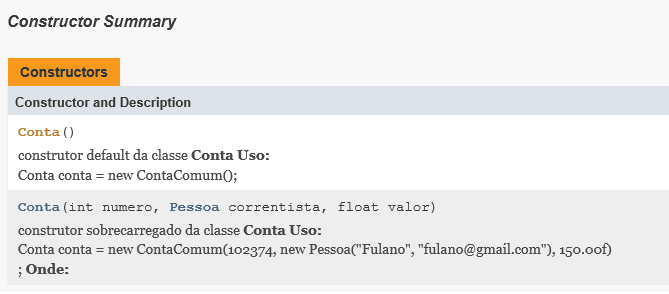
}

## Gerando o arquivo javadoc

bancos --> Gerar javadoc

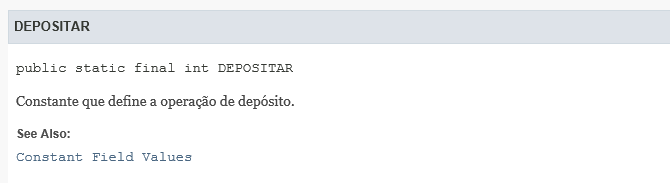


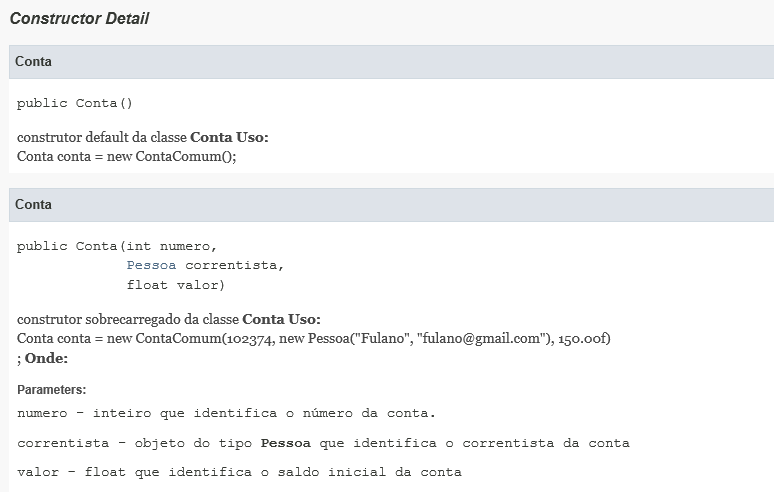




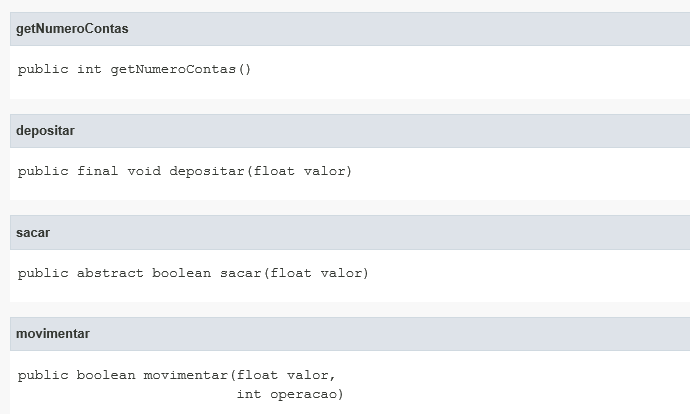


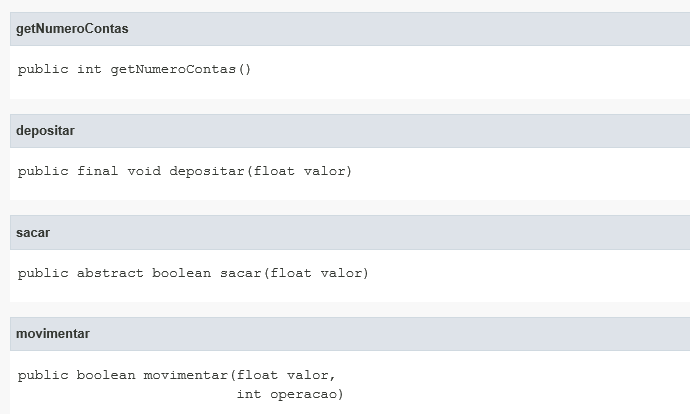








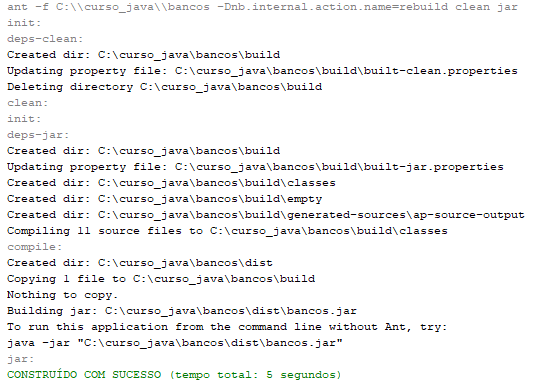




# Aula 14 - Criando bibliotecas de componentes

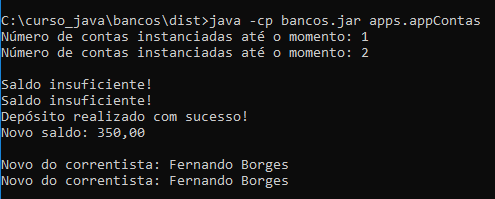
## Criando o aplicativo

bancos --> Limpar e construir

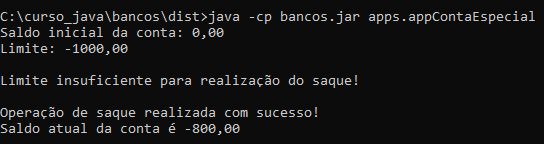


## Executando os aplicativos

java -cp bancos.jar apps.appContas



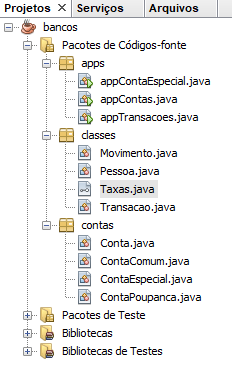
java -cp bancos.jar apps.appContaEspecial



java -cp bancos.jar apps.appTransacoes



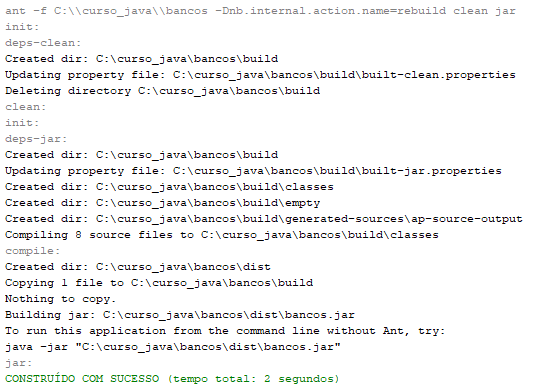
## Gerando uma biblioteca de componentes



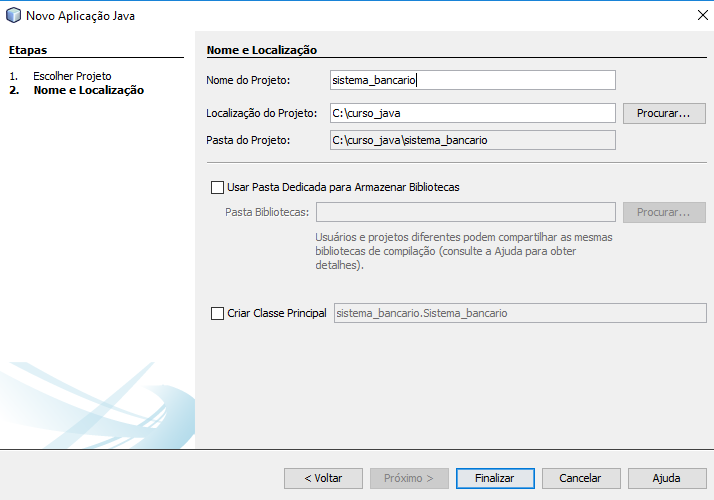
- Apague a pasta apps

- Gere a biblioteca:

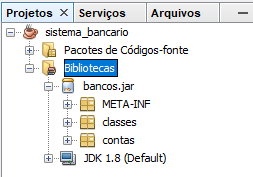
bancos --> Limpar e construir



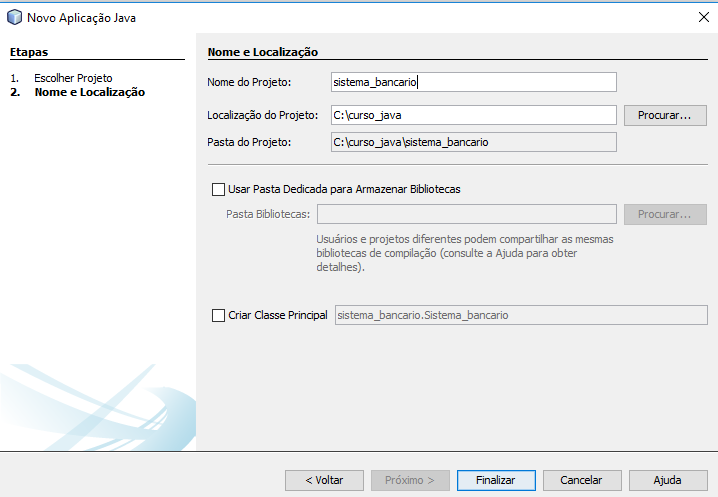
- Feche o projeto e crie um novo projeto chamado "sistema\_bancario":

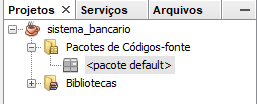


- Importe a biblioteca bancos.jar criada anteriormente

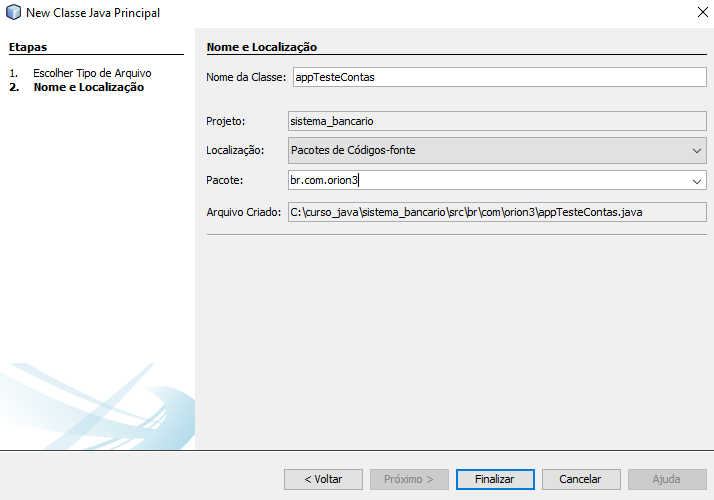


- Crie uma nova aplicação chamada appTesteContas.java





- Crie a classe appTestesConta.java



**appTesteContas.java**

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

public class appTesteContas {

public static void main(String[] args) {

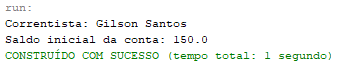
Conta contacomum = new ContaComum(198764, new Pessoa("Gilson Santos", "gilson\_santos@gmail.com"), 150.00f);

System.out.println("Correntista: " + contacomum.getCorrentista().getNome());

System.out.println("Saldo inicial da conta: " + contacomum.getSaldo());

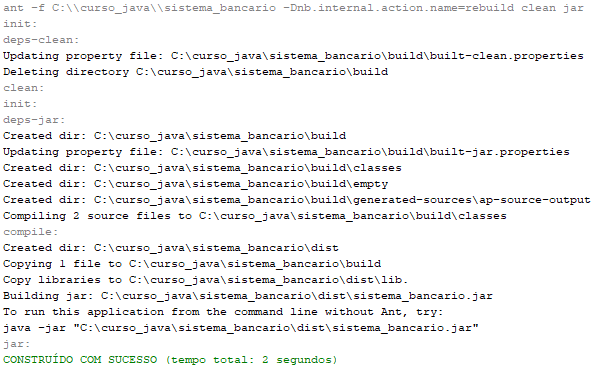
}

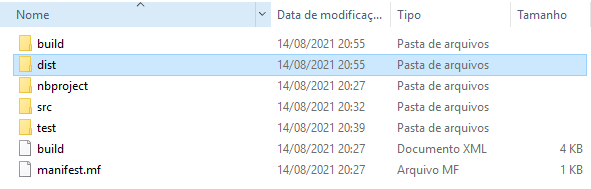
}

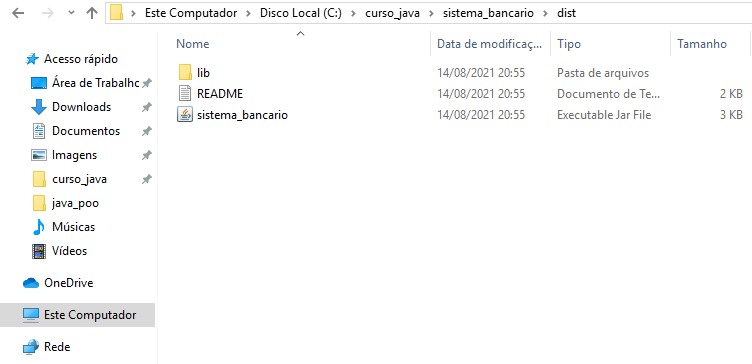


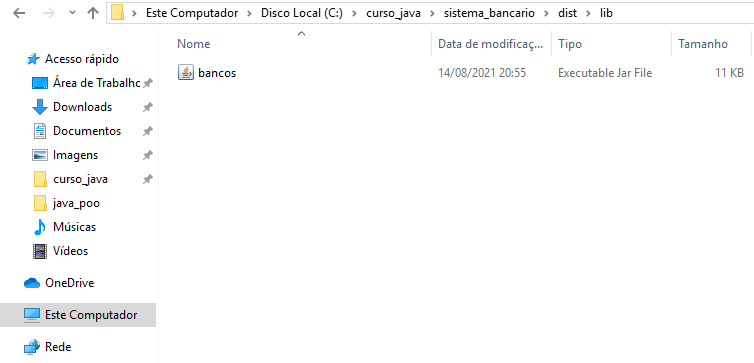
= Gere a aplicação:

bancos --> Limpar e construir



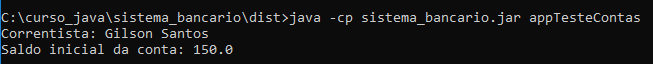






- No prompt DOS:

java -cp sistema\_bancario.jar appTesteContas



**appTesteContas.java**

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

public class appTesteContas {

public static void main(String[] args) {

Conta contacomum = new ContaComum(198764, new Pessoa("Gilson Santos", "gilson\_santos@gmail.com"), 150.00f);

System.out.println("Correntista: " + contacomum.getCorrentista().getNome());

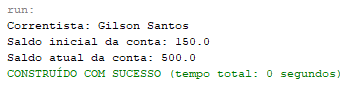
System.out.println("Saldo inicial da conta: " + contacomum.getSaldo());

contacomum.depositar(350.00f);

System.out.println("Saldo atual da conta: " + contacomum.getSaldo());

}

}

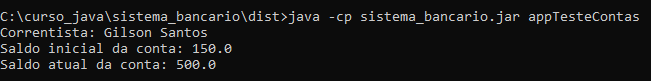


= Novamente gere a aplicação:

bancos --> Limpar e construir

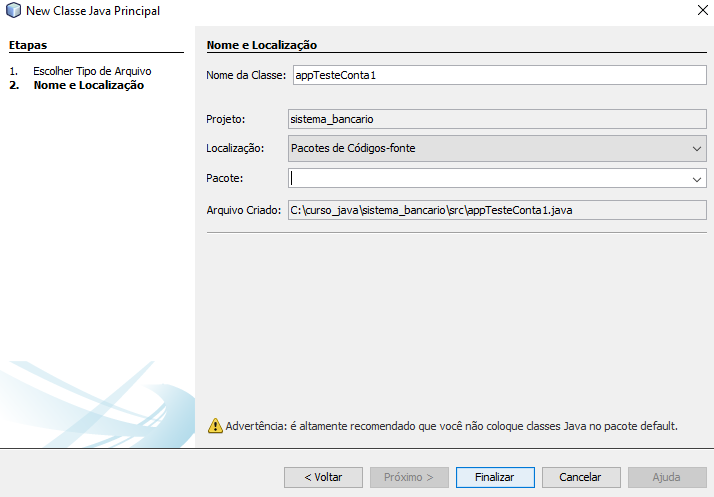
- No prompt DOS:

java -cp sistema\_bancario.jar appTesteContas



# Aula 15 - Interação com o usuário

- Crie uma nova app chamada appTesteConta1.java



**appTesteConta1.java**

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

import java.util.Scanner;

public class appTesteConta1 {

public static void main(String[] args) {

// Solicitando ao usuário os dados para criar uma conta comum

Scanner teclado = new Scanner(System.in);

int numero;

String nome, email;

float saldo, valordeposito;

System.out.print("Número da conta: ............... ");

numero = teclado.nextInt();

System.out.print("Correntista: ................... ");

nome = teclado.next();

System.out.print("Email: ......................... ");

email = teclado.next();

System.out.print("Saldo inicial: ................. ");

saldo = teclado.nextFloat();

Conta contacomum = new ContaComum(numero, new Pessoa(nome, email), saldo);

System.out.print("Valor do depósito: ............. ");

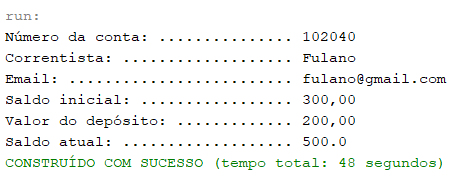
valordeposito = teclado.nextFloat();

contacomum.depositar(valordeposito);

System.out.println("Saldo atual: ................... " + contacomum.getSaldo());

}

}



# Aula 16 - Tratamento de exceções em aplicações Java

**appTesteConta1.java**

import classes.Pessoa;

import contas.Conta;

import contas.ContaComum;

import java.util.InputMismatchException;

import java.util.Locale;

import java.util.Scanner;

public class appTesteConta1 {

public static void main(String[] args) {

// Solicitando ao usuário os dados para criar uma conta comum

Scanner teclado = new Scanner(System.in);

teclado.useLocale(Locale.FRENCH);

int numero = 0;

String nome, email;

float saldo = 0, valordeposito = 0;

System.out.print("Número da conta: ............... ");

try{

numero = teclado.nextInt();

} catch(InputMismatchException e) {

System.out.println("Informe um valor numérico inteiro para o número da conta, ex: 123456");

System.exit(0);

}

System.out.print("Correntista: ................... ");

nome = teclado.next();

System.out.print("Email: ......................... ");

email = teclado.next();

System.out.print("Saldo inicial: ................. ");

try{

saldo = teclado.nextFloat();

} catch(InputMismatchException e) {

System.out.println("Informe um valor numérico válido para o saldo inicial, ex: 300,00");

System.exit(0);

}

Conta contacomum = new ContaComum(numero, new Pessoa(nome, email), saldo);

System.out.print("Valor do depósito: ............. ");

try{

valordeposito = teclado.nextFloat();

} catch(InputMismatchException e) {

System.out.println("Informe um valor numérico válido para o valor do depósito, ex: 100,00");

System.exit(0);

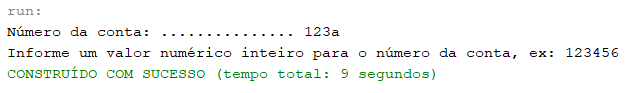
}

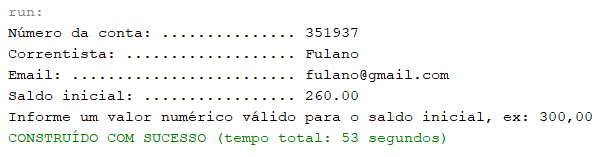
contacomum.depositar(valordeposito);

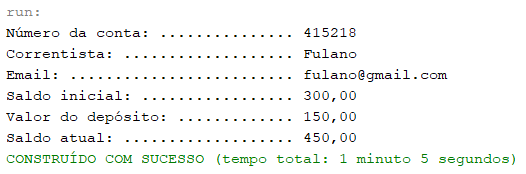
System.out.printf("Saldo atual: ................... %.2f \n", contacomum.getSaldo());

}

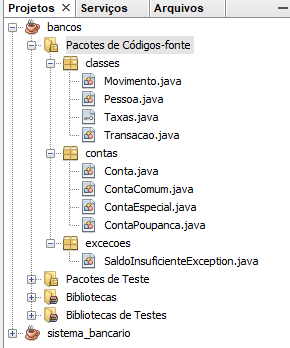
}







# Aula 17 - Criando nossas próprias exceções



**excecoes\SaldoInsuficienteException.java**

package excecoes;

public class SaldoInsuficienteException extends Exception {

private float saldo;

private float valor;

public SaldoInsuficienteException(float saldo, float valor) {

this.saldo = saldo;

this.valor = valor;

}

@Override

public String toString() {

String saldo\_formatado = String.format("%.2f", saldo);

String valor\_formatado = String.format("%.2f", valor);

return "Saldo insuficiente para o saque! O saldo atual é R$" + saldo\_formatado + " e você está tentando sacar R$" + valor\_formatado;

}

}

**contas\Conta.java**

package contas;

import classes.Pessoa;

import excecoes.SaldoInsuficienteException;

/\*\*

\* A classe <strong>Conta</strong> define um tipo de dado abstrato para a criação de estrutura de classes de contas bancárias

\* @author Roberto Pinheiro

\* @since ago/2021

\* @version 1.0

\*/

public abstract class Conta {

// atributos de instância

/\*\*

\* O atributo número é utilizado para referenciar o número da conta.

\*/

protected int numero;

/\*\*

\* O atributo correntista, do tipo <b>Pessoa</b> é utilizado para referenciar um correntista.

\*/

protected Pessoa correntista;

/\*\*

\* O atributo saldo é utilizado para referenciar a saída da conta.

\*/

protected float saldo;

// atributo de classe

/\*\*

\* Constante que define a operação de saque.

\*/

public final static int SACAR = 0;

/\*\*

\* Constante que define a operação de depósito.

\*/

public final static int DEPOSITAR = 1;

private static int numero\_contas;

/\*\*

\* construtor default da classe <b>Conta</b>

\* <b>Uso: </b><br>

\* Conta conta = new ContaComum();

\*/

public Conta() {

incrementa\_contas();

}

/\*\*

\* construtor sobrecarregado da classe <b>Conta</b>

\* <b>Uso: </b><br>

\* Conta conta = new ContaComum(102374, new Pessoa("Fulano", "fulano@gmail.com"), 150.00f)<br>;

\* <b>Onde:</b><br>

\* @param numero inteiro que identifica o número da conta.

\* @param correntista objeto do tipo <b>Pessoa</b> que identifica o correntista da conta

\* @param valor float que identifica o saldo inicial da conta

\*/

public Conta(int numero, Pessoa correntista, float valor) {

this();

this.numero = numero;

this.correntista = correntista;

this.saldo = valor;

}

public final int getNumero() {

return numero;

}

public final void setNumero(int numero) {

this.numero = numero;

}

public final Pessoa getCorrentista() {

return correntista;

}

public final void setCorrentista(Pessoa correntista) {

this.correntista = correntista;

}

public final float getSaldo() {

return saldo;

}

public final void setSaldo(float saldo) {

this.saldo = saldo;

}

public int getNumeroContas(){

return this.numero\_contas;

}

public final void depositar(float valor){ // método concreto

this.setSaldo(this.getSaldo() + valor);

}

public abstract void sacar(float valor) throws SaldoInsuficienteException; // método polimórfico

public void movimentar(float valor, int operacao) throws SaldoInsuficienteException {

switch(operacao){

case DEPOSITAR:

this.depositar(valor);

break;

case SACAR:

this.sacar(valor);

}

}

private static void incrementa\_contas(){

numero\_contas++;

}

}

**contas\ContaComum.java**

package contas;

import classes.Taxas;

import classes.Pessoa;

import excecoes.SaldoInsuficienteException;

public final class ContaComum extends Conta implements Taxas {

public ContaComum() { }

public ContaComum(int numero, Pessoa correntista, float valor) {

super(numero, correntista, valor);

}

@Override

public void sacar(float valor) throws SaldoInsuficienteException{

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

} else {

throw new SaldoInsuficienteException(this.getSaldo(), valor);

}

}

@Override

public float getTaxaManutencao(){

return 15.00f;

}

@Override

public void descontarTaxaManutencao(){

this.setSaldo(this.getSaldo() - this.getTaxaManutencao());

}

}

**contas\ContaPoupanca.java**

package contas;

import classes.Pessoa;

import excecoes.SaldoInsuficienteException;

public class ContaPoupanca extends Conta {

public ContaPoupanca() {

}

public ContaPoupanca(int numero, Pessoa correntista, float valor) {

super(numero, correntista, valor);

}

@Override

public void sacar(float valor) throws SaldoInsuficienteException{

if(this.getSaldo() >= valor){

this.setSaldo(this.getSaldo() - valor);

} else {

throw new SaldoInsuficienteException(this.getSaldo(), valor);

}

}

}

**contas\ContaEspecial.java**

package contas;

import classes.Pessoa;

import contas.Conta;

import excecoes.SaldoInsuficienteException;

public final class ContaEspecial extends Conta {

private float limite;

public ContaEspecial(){

}

public ContaEspecial(int numero, Pessoa correntista, float valor, float limite){

super(numero, correntista, valor);

this.setLimite(limite);

}

public float getLimite() {

return limite;

}

public void setLimite(float limite){

this.limite = limite;

}

@Override

public void sacar(float valor) throws SaldoInsuficienteException {

if(this.getSaldo() - valor >= this.getLimite()) {

this.setSaldo(this.getSaldo() - valor);

} else {

throw new SaldoInsuficienteException(this.getSaldo(), valor);

}

}

}

**classes\Movimento.java**

package classes;

import contas.Conta;

import excecoes.SaldoInsuficienteException;

import java.util.Date;

public class Movimento {

private Date data;

private Conta conta;

private String historico;

private float valor;

private float saldoanterior;

private int operacao;

// agregação

public Movimento(Date data, Conta conta, String historico, float valor, int operacao) {

this.data = data;

this.conta = conta;

this.historico = historico;

this.valor = valor;

this.operacao = operacao;

}

public Date getData() {

return data;

}

public void setData(Date data) {

this.data = data;

}

public Conta getConta() {

return conta;

}

public void setConta(Conta conta) {

this.conta = conta;

}

public String getHistorico() {

return historico;

}

public void setHistorico(String historico) {

this.historico = historico;

}

public float getValor() {

return valor;

}

public void setValor(float valor) {

this.valor = valor;

}

public float getSaldoanterior() {

return saldoanterior;

}

public void setSaldoanterior(float saldoanterior) {

this.saldoanterior = saldoanterior;

}

public int getOperacao() {

return operacao;

}

public void setOperacao(int operacao) {

this.operacao = operacao;

}

public boolean movimentar() throws SaldoInsuficienteException {

this.saldoanterior = conta.getSaldo();

if(operacao == Conta.SACAR)

this.conta.movimentar(this.valor, Conta.SACAR);

else if(operacao == Conta.DEPOSITAR) {

this.conta.movimentar(this.valor, Conta.DEPOSITAR);

return true;

}

return false;

}

}

**classes\Transacao.java**

package classes;

import contas.Conta;

import excecoes.SaldoInsuficienteException;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

// composição transação/movimento (relação mais forte que agregação)

public class Transacao {

private List<Movimento> movimentos;

public Transacao(){

movimentos = new ArrayList<>();

}

// Efetiva uma movimentação na conta

public void realizarTTransacao(Date data, Conta conta, float valor, String historico, int operacao) throws SaldoInsuficienteException {

Movimento movimento = new Movimento(data, conta, historico, valor, operacao);

movimento.movimentar();

this.movimentos.add(movimento);

}

public void extornaTransacao(){

for(Movimento movimento: this.movimentos){

movimento = null;

}

}

public List<Movimento> getMovimentos(){

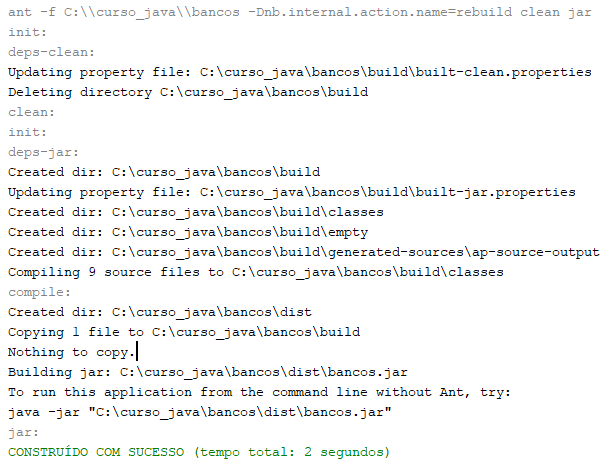
return this.movimentos;

}

}

## Gerando o pacote de distribuição do aplicativo

bancos --> Limpar e Construir



## Rodando appTesteConta1.java

