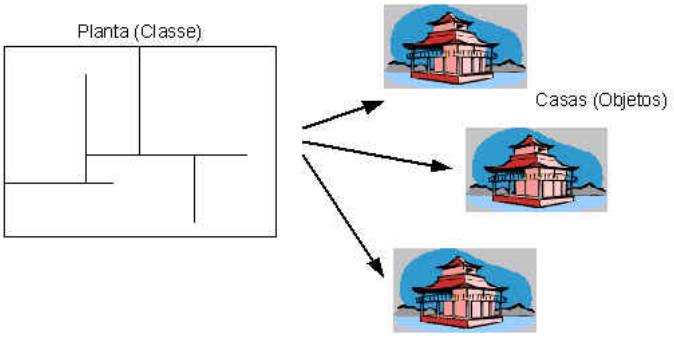
Orientação a Objeto - Usando Objetos

Objetivos da seção

- Introduzir a Orientação a Objeto (OO) e o uso de objetos
  - Objetos do mundo real
  - Comportamento dos objetos do mundo real
  - Reificação e objetos de software "vivos"
  - Interfaces de objetos
  - Estado de um objeto (atributos e sua persistência)
  - Classificação dos objetos
  - Referências a objetos
  - Identidade de objetos
  - Encapsulamento
  - Comportamento definido pela classe e pelos métodos
  - Métodos como envio de mensagens
  - Tratamento de erros com exceções

Introdução à Orientação a Objeto (OO)

- Pode-se pensar sobre o mundo real como uma coleção de objetos relacionados
- Exemplos de objetos do mundo bancário:
  - Contas correntes
  - Contas de poupança
  - Clientes
  - Caixas
  - Agências
  - Cheques
  - Extratos
- Objetos podem ser agrupados em classes
  - Conta corrente
  - Conta de poupança
  - Cliente
  - Caixa
  - Agência
  - Cheque
  - Extratos
- Observe que existem várias Contas correntes de uma mesma classe "Conta corrente"
  - A diferença entre classe e objeto
    - "Classe" é um gabarito (como a planta de uma casa)
    - "Objeto" é a concretização do gabarito (casas feitas a partir da mesma planta)



- Objetos de uma certa classe têm atributos
  - Uma Conta tem um número, um saldo, um histórico de transações
  - Um Cliente tem um nome, um endereço
  - Um Cheque tem um valor
- Objetos de uma mesma classe têm um mesmo comportamento
  - Clientes entram numa agência
  - Clientes fazem depósitos e saques
  - Clientes emitem cheques

- Certos objetos não têm comportamento
  - Contas não são vivas: não "fazem" nada
- Objetos podem estar relacionados
  - Um cliente possui várias Contas
- Podemos usar objetos ao fazer software também
- Há várias vantagens de fazer isso
  - É um pouco difícil entender todas as vantagens de OO agora
  - Mencionemos apenas duas:
    - É mais fácil conversar com o cliente que pediu o software se falarmos com objetos que existem no mundo dele (o mundo do software fica mais perto do mundo real)
    - O software feito com OO pode ser feito com maior qualidade e pode ser mais fácil de escrever e, principalmente, *alterar* no futuro
    - Esses pontos não ficarão claros na disciplina de Programação 2 mas certamente o ficarão adiante no curso

O Primeiro Programa OO: Instanciação e Uso de Objetos

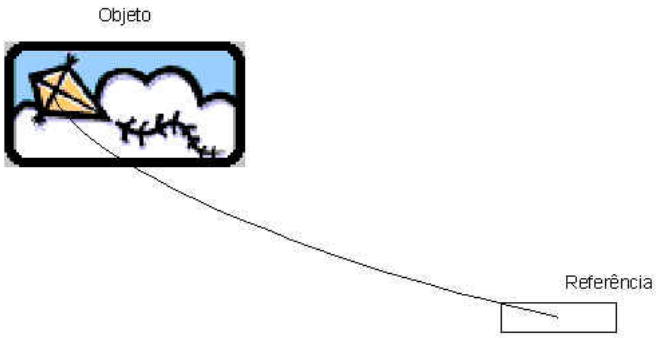
- Nesta seção, queremos aprender a usar objetos e não a criá-los ainda
- Problema a resolver: modelar Contas bancárias simples e fazer algumas manipulações
- Usaremos uma classe chamada ContaSimples que já existe na biblioteca de classes escritas pelo professor
  - Esta biblioteca se chama p1.aplic.banco e serve para escrever aplicações simples do mundo bancário
  - Adiante nesta disciplina, veremos a implementação dessas classes
- A solução está no arquivo Banco1.java

```
/*
 * Movimentação simples de uma conta bancária
 */

// Precisaremos usar classes já prontas no programa
// Aqui, estamos trazendo essas classes para "dentro" do nosso programa
import p1.aplic.banco.*; // para poder manusear contas bancárias

// Programa Banco1
public class Banco1 {
        // O programa sempre tem um "método" main que é onde começa a execução
        public static void main(String args[]) {
                // Abra uma conta de número 1 para João com CPF 309140605-06
                // A conta será "referenciada" com a variável umaConta
                ContaSimples umaConta = new ContaSimples("Joao", "30914060506",
                // Nesta conta, deposite R$1000,00
                umaConta.depositar(1000.0);

                // Imprima o saldo da conta de João
                double saldo = umaConta.getSaldo();
                System.out.print("Saldo da conta de Joao antes do saque: ");
                System.out.println(saldo);

                // Saque R$300,00 desta conta
                umaConta.sacar(300.0);
                // Imprima o objeto umaConta
                System.out.println(umaConta);
                // Imprima o saldo da conta de João
                System.out.println("Saldo da conta de Joao depois do saque: "
                                + umaConta.getSaldo());
        } // fim do método main
} // fim da classe Banco1
```

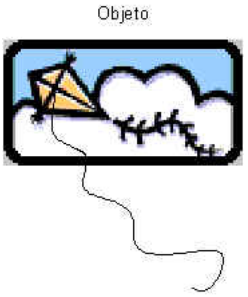
- Saída do programa:

```
Saldo da conta de Joao antes do saque: 1000.0
ContaSimples numero 1, titular Joao, data 01/06/2001, saldo R$700,00
Saldo da conta de Joao depois do saque: 700.0
```

- A linha com "import" é usada para dizer ao compilador Java que estaremos usando classes do "pacote" p1.aplic.banco
- Um objeto é criado com a palavra new, especificando sua classe
  - Diz-se "chamamos o construtor da classe"
  - A operação também se chama instanciar o objeto
- No caso da classe ContaSimples, o construtor tem 3 parâmetros que são o valor de certos atributos do objeto (nome do titular, CPF do titular, número da conta)
- Observe que não temos classe Cliente para representar o titular da conta
  - A conta em si assume os atributos do titular
- A variável umaConta é do tipo da classe ContaSimples e armazena uma referência ao objeto
  - É como se o objeto fosse uma pipa e a referência fosse uma linha amarrada à pipa
  - É semelhante a ponteiros em outras linguagens



- Um objeto existe enquanto houver pelo menos uma referência a ele
  - Depois que não houver mais referências, o objeto some (que nem a pipa sem linha!)



- No programa, estamos vendo chamadas a 3 comportamentos do objeto umaConta
  - depositar(...)
  - sacar(...)
  - getSaldo()
  - toString()
- A sintaxe umaConta.depositar(1000.0) significa que estamos chamado o método depositar() do objeto umaConta
  - Um método é como um sub-programa, subrotina ou função de outras linguagens
  - Também se fala que estamos enviando a mensagem "depositar" para o objeto "umaConta"
- Certos métodos podem ter parâmetros e outros não
- Podemos imprimir um objeto como um todo!

```
System.out.println(umaConta);
```

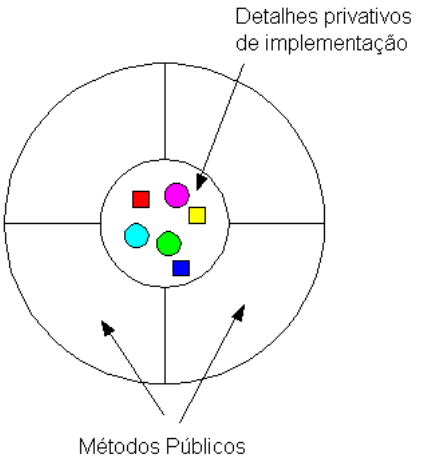
- Isso é equivalente a chamar:

```
System.out.println(umaConta.toString());
```

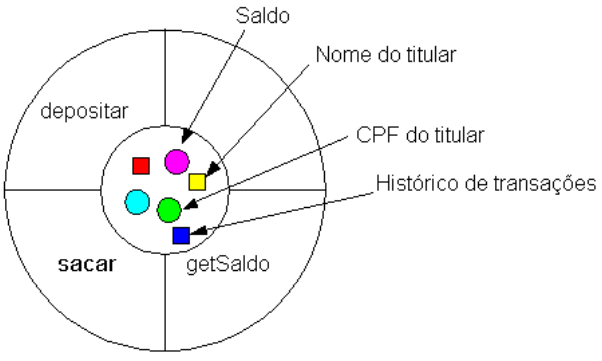
- ... porque toda classe tem um método toString que retorna uma representação do objeto e dos seus atributos como String
  - Pergunta: qual é o String retornado por umaConta.toString()?
  - A partir deste String, você pode identificar um provável atributo de uma ContaSimples que não vimos até agora nem informamos no construtor?

Encapsulamento

- Muito importante: conseguimos usar objetos da classe ContaSimples sem saber nada sobre como ContaSimples está escrita em Java!
  - Isso se chama Ocultação de Informação e é muito importante na programação
  - É a forma básica de lidar com a complexidade dos programas
- Também podemos dizer que a Classe ContaSimples encapsula dados e comportamento em cima desses dados
  - Os dados são os atributos escondidos de nós (saldo, histórico de transações, dados do titular, ...)
  - O comportamento são os métodos que podemos chamar para manipular o objeto
  - Só podemos "mexer" com o objeto através de seus métodos



- Portanto, vê-se que classes como ContaSimples têm comportamento
  - Mais precisamente, os objetos dessa classe é que têm comportamento definido pela classe
  - Podemos ver uma definição mais completa do comportamento da classe ContaSimples aqui



Próximo Programa: Identidade de Objetos

- Pense sobre a implementação do seguinte programa, dado em pseudo-código

```
Programa Banco2
        Abra uma conta de número 1 para João com CPF 309140605-06
        Nesta conta, deposite R$1000,00
```

```
        Abra uma conta de número 2 para Ana com CPF 123456789-01
        Transfira R$400,00 da conta de João para a conta de Ana
        Imprima o saldo da conta de João
        Imprima o saldo da conta de Ana
```

- A solução está no arquivo Banco2.java

```
/*
 * Movimentação simples de duas contas bancárias
 */

import p1.aplic.banco.*; // para poder manusear contas bancárias

public class Banco2 {
        public static void main(String args[]) {
                // Abra uma conta de número 1 para João com CPF 309140605-06
                // A conta será "referenciada" com a variável umaConta
                ContaSimples umaConta = new ContaSimples("Joao", "30914060506",
                // Nesta conta, deposite R$1000,00
                umaConta.depositar(1000.0);

                // Abra uma conta de número 2 para Ana com CPF 123456789-01
                // A conta será "referenciada" com a variável outraConta
                ContaSimples outraConta = new ContaSimples("Ana", "12345678901"
                // Transfira R$400,00 da conta de João para a conta de Ana
                umaConta.transferir(outraConta, 400.0);

                // Imprima o saldo da conta de João
                double saldo = umaConta.getSaldo();
                System.out.print("Saldo da conta de Joao: ");
                System.out.println(saldo);

                // Imprima o saldo da conta de Ana
                System.out.println("Saldo da conta de Ana: " + outraConta.getSa
                // Imprima o saldo da conta de Ana, novamente, com outro método
                System.out.println("Saldo da conta de Ana: "
                                + outraConta.getSaldoMonetário());

                // Impressao dos objetos
                System.out.println("Situacao da primeira conta: ");
                System.out.println(umaConta);
                System.out.println("Situacao da segunda conta: ");
                System.out.println(outraConta);
        } // fim do método main
} // fim da classe Banco2
```

- Saída do programa:

```
Saldo da conta de Joao: 600.0
Saldo da conta de Ana: 400.0
Saldo da conta de Ana: R$400,00
Situacao da primeira conta:
ContaSimples numero 1, titular Joao, data 01/06/2001, saldo R$600,00
Situacao da segunda conta:
ContaSimples numero 2, titular Ana, data 01/06/2001, saldo R$400,00
```

- Este exemplo mostra o conceito de identidade de objetos
  - Os objetos cujas referências são umaConta e outraConta são dois objetos diferentes (com identidade diferente) embora sejam da mesma classe (do mesmo molde)

- Mesmo que todos os atributos dos dois objetos fosse iguais, eles teriam identidade diferente
  - Seriam objetos diferentes
  - Eles seriam armazenados em lugares diferentes da memória
  - A referência é essencialmente o endereço de memória do objeto
- Observe que, no programa, uma referência a um objeto pode ser passada como parâmetro de um método como qualquer outro valor
  - Identifique isso no programa Banco2

Próximo Programa: Várias Classes

- Nos programas acima, o titular das contas bancárias não aparece como objeto
- Suspeitamos de sua existência porque falamos do nome e CPF do titular
- No próximo programa, trataremos o titular de uma conta de forma explícita como objeto
- A solução está no arquivo Banco4.java

```
/*
 * Uso de objetos de várias classes, entrada de dados
 */

import p1.aplic.banco.*;
import p1.aplic.geral.*; // para mexer com a classe Pessoa
import p1.io.*; // para fazer entrada de dados

public class Banco4 {
        public static void main(String args[]) {
                // declaração de variáveis de 2 classes (tipos) diferentes
                // no fim, teremos 4 objetos de 2 classes
                Pessoa oTitular1, oTitular2;
                ContaSimples umaConta, outraConta;

                // tempo de vida: os objetos ainda nao existem
                // a seguinte linha daria erro:
                // umaConta.depositar(1000.0);

                // cria dois objetos Pessoa com nome e CPF
                oTitular1 = new Pessoa("Joao", "30914060506");
                oTitular2 = new Pessoa("Ana", "12345678901");

                // cria as duas contas
                // Observe que ContaSimples pode ser criada com um titular
                // do tipo Pessoa em vez de dar nome e CPF separadamente
                umaConta = new ContaSimples(oTitular1, 1);
                outraConta = new ContaSimples(oTitular2, 2);

                // Vamos ver quanto vamos depositar
                double valorADepositar = Entrada.in
                                .lerdouble("Entre com o valor a depositar: ");
                System.out.println("Vou depositar " + valorADepositar);
                umaConta.depositar(valorADepositar);

                // Vamos ver quanto vamos transferir
                double valorATransferir = Entrada.in
                                .lerdouble("Entre com o valor a transferir: ");
                System.out.println("Vou transferir " + valorATransferir);
                umaConta.transferir(outraConta, valorATransferir);

                // Fecha a agencia e guarda toda a informação em arquivo
                Agencia.fecharCaixa();
```

```
                System.out.println("OK. Caixa fechado.");
        } // main
} // Banco4
```

- A saída do programa:

```
Entre com o valor a depositar: 1000
Vou depositar 1000.0
Entre com o valor a transferir: 34
Vou transferir 34.0
OK. Caixa fechado.
```

- Observe as múltiplas importações
- Observe a criação de 4 objetos, de 2 classes
  - Quais são as 4 referências?
  - Quais são as 2 classes?
- A declaração de uma referência não significa que ela "aponte" para um objeto
  - Precisa instanciar o objeto primeiro
- Observe o construtor de ContaSimples
  - É diferente da chamadas que vimos até agora
  - nome e cpf não mais passados no construtor da ContaSimples, mas no construtor da Pessoa
  - Neste programa, passamos uma Pessoa como titular e não passamos nome e CPF
  - Isso se chama overload de métodos
    - Dois métodos com o mesmo nome mas com parâmetros diferentes são métodos completamente diferentes
    - É semelhante ao overload do operador binário +
- Observe a chamada ao novo transferir() de ContaSimples
  - Veja o que este método faz aqui
- Finalmente, tem um método estranho chamado método de classe

```
Agencia.fecharCaixa();
```

- Não há referência a um objeto chamado Agencia
  - Na realidade, Agencia é o nome de uma classe e o método Agencia.fecharAgencia() é um método que pertence à classe em si e não aos objetos da classe Agencia
  - Isso foi feito para simplificar a situação para termos uma única agência
  - Num programa real, a Agencia teria que ser instanciada como qualquer objeto
  - Você pode ver o que o métodos fecharCaixa() faz aqui
  - Como você vai ver, trata-se de salvar os dados das contas, transações, saldos, etc. em arquivo
  - Chamamos isso de persistência
  - Faremos uso disso no próximo programa

Próximo Programa: Acesso a Dados Persistentes

- Como manipular contas bancárias que já existem?
- Usaremos um método da classe Agencia para achar uma conta cujo número sabemos
- A solução está no arquivo Banco5.java

```
/*
 * Persistência
 */

import p1.aplic.banco.*;
import p1.aplic.geral.*; // para mexer com a classe Data
import p1.io.*;
```

```
public class Banco5 {
        public static void main(String args[]) {
                // declaração de variáveis de 3 classes (tipos) diferentes
                // no fim, teremos 4 objetos de 3 classes
                ContaSimples umaConta;
                Extrato umExtrato;
                Data hoje, ontem;

                // localiza a conta de número 1
                // (ela já foi criada e armazenada anteriormente)
                // O "cast" (ContaSimples) será explicado em outro momento
                // O motivo é que localizarConta retorna um objeto do tipo Cont
                // que representa uma conta bancária genérica e não apenas uma
                // ContaSimples
                umaConta = (ContaSimples) Agencia.localizarConta(1);

                // Vamos fazer um saque
                System.out.println("Voce tem " + umaConta.getSaldoMonetário()
                                + " na conta");
                double valor = Entrada.in.lerdouble("Quanto voce quer sacar? ")
                umaConta.sacar(valor);

                // Vamos tirar um extrato de conta entre ontem e hoje
                hoje = new Data(); // hoje representa a data de hoje
                ontem = new Data(); // ontem ainda representa hoje
                ontem.somarDia(-1); // agora, temos a data de ontem
                umExtrato = umaConta.criarExtrato(ontem, hoje);
                // Imprime o extrato
                System.out.println(umExtrato.formatar());
                System.out.println("Saldo: " + umaConta.getSaldoMonetário());

                // Vamos atualizar as informações persistentes
                Agencia.fecharCaixa();
        } // main
} // Banco5
```

- A saída do programa:

```
Voce tem R$966,00 na conta
Quanto voce quer sacar? 53
Extrato de conta entre 31/05/2001 16:35 e 01/06/2001 16:35
Data                  Debito Credito          Valor Descricao
01/06/2001 16:18     0000000 0000001       R$1000,00 deposito
01/06/2001 16:18     0000001 0000002         R$34,00 transferencia para conta 2
01/06/2001 16:35     0000001 0000000         R$53,00 saque

Saldo: R$913,00
```

- Observe as várias classes usadas
  - Quais são?
- Observe como as referências são inicializadas
  - O objeto "umaConta" não é criado com new mas recebido como retorno do método localizarConta() da classe Agencia
  - A classe Agencia achou este objeto num arquivo (salvo no programa Banco4)
- Observe com que facilidade manipulamos e usamos datas para tirar um extrato
  - Na OO, tudo vira um objeto, até datas, extratos, etc.
- Estamos fazendo coisas até complexas sem dificuldade
  - Porque abstrações adequadas (classes) já existem para modelar um mundo

- Exercício: gere um extrato em formato HTML
  - Dica: ver a documentação da classe Extrato

Próximo Programa: um Exemplo Usando while e switch

- A solução está no arquivo Banco8.java

```
/*
 * Laço: leitura/processamento (até um marcador final)
 */

import p1.aplic.banco.ContaSimples;
import p1.io.Entrada;

public class Banco8 {
        public static void main(String args[]) {
                final int DEPOSITAR = 1;
                final int SACAR = 2;
                final int SALDO = 3;
                final int SAIR = 4;
                // o caractere de fim de linha pode ser diferente de um sistema
                // outro
                // exemplo: "\r\n" no MSDOS/Windows, "\n" no UNIX, "\r" no MacI
                // etc.
                final String fimDeLinha = System.getProperty("line.separator");
                final String prompt = "Digite a opcao desejada:" + fimDeLinha
                                + DEPOSITAR + ". Depositar" + fimDeLinha + SACA
                                + fimDeLinha + SALDO + ". Saldo" + fimDeLinha +
                                + fimDeLinha + "Opcao: ";
                if (args.length != 3) {
                        System.err
                                        .println("Sintaxe: java Banco8 titular
                        System.exit(1);
                }
                ContaSimples aConta = new ContaSimples(args[0], args[1], Intege
                                .parseInt(args[2]));

                int opção = Entrada.in.lerInt(prompt);
                while (opção != SAIR) {
                        switch (opção) {
                        case DEPOSITAR:
                                double valorADepositar = Entrada.in
                                                .lerdouble("Entre com o valor a
                                aConta.depositar(valorADepositar);
                                break;
                        case SACAR:
                                double valorASacar = Entrada.in
                                                .lerdouble("Entre com o valor a
                                aConta.sacar(valorASacar);
                                break;
                        case SALDO:
                                System.out.println("Saldo: " + aConta.getSaldoM
                                break;
                        default:
                                System.err.println("Opcao " + opção + " desconh
                                break;
                        }
                        // leia novamente para voltar ao início do laço
                        opção = Entrada.in.lerInt(prompt);
                }
```

```
                System.out.println("Saldo final: " + aConta.getSaldoMonetário()
        } // main
} // Banco8
```

- A saída do programa:

```
C:\...\p2\html\src>java -classpath .;packagep1\p1.jar Banco8 joao 12345678901 4
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 1
Entre com o valor a depositar: 34
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 2
Entre com o valor a sacar: 567
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 3
Saldo: R$34,00
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 2
Entre com o valor a sacar: 30
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 3
Saldo: R$4,00
Digite a opcao desejada:
1. Depositar
2. Sacar
3. Saldo
4. Sair
Opcao: 4
Saldo final: R$4,00
```

- Observe a declaração de constantes com a palavra final
  - *Nunca* usar números mágicos
- A inicialização de fimDeLinha é um macete para ter portabilidade do programa
  - O programa deve executar corretamente em vários ambientes
  - \r significa "RETURN" e \n significa "NEWLINE"
- Observe como aConta é criada a partir de argumentos de linha de comando
  - Integer.parseInt(String) converte um String a um inteiro (int)
- O comando switch implementa uma decisão múltipla

- Às vezes, pode ser mais legível do que uma sequência de if-else

O Mundo de Software Simulando o Mundo Real

- A princípio, o mundo de software será semelhante ao mundo real
- Classes, objetos e atributos que existem no mundo real normalmente existem no mundo de software que quer tratar do mundo real
- Portanto, software (aplicações) que lidam com o mundo bancário:
  - Trataria das classes Agência, Conta, Cliente, etc.
  - Teria objetos como no mundo real
  - Esses objetos teriam os mesmos atributos do mundo real
- O que foi falado acima não é estritamente verdade porque o mundo de software é um modelo do mundo real e só trata das coisas importantes para o business que utilizará o software
  - Exemplo: Clientes têm um tamanho de sapato mas o software não incluirá este atributo na classe Cliente
- Apesar de nem sempre ser verdade, é um bom ponto de partida
- Porém, há diferenças grandes quando pensamos no comportamento
- Há dois grandes motivos
- Primeiro motivo: não vamos modelar todo o comportamento dos objetos
  - Exemplo: Clientes tomam café no mundo real mas provavelmente não no software
- Segundo motivo: os comportamentos frequentemente são assumidos por objetos diferentes
  - Exemplo: No mundo real, quem faz um depósito? Um Cliente
  - No software, qual classe assumiria a responsabilidade de fazer um depósito?
    - A classe Conta
- Por que isso????
  - A primeira grande regra de programação Orientada a Objeto é a seguinte: "Comportamentos são associados aos objetos que possuem os atributos afetados pelo comportamento"
  - Um depósito afeta duas coisas, do ponto de vista de um sistema bancário:
    - O saldo de uma conta
    - O histórico de transações feitas a uma conta
  - Como se pode ver, o depósito afeta apenas atributos de uma Conta
  - Portanto, a responsabilidade de implementar este comportamento será atribuída à Conta e não ao Cliente, embora, no mundo físico, seja o Cliente que faz o depósito
  - A primeira grande regra acima tem outros nomes:
    - "Colocar as responsabilidades com os dados"
    - Padrão de Projeto Expert
      - Quem faz algo é quem é o expert nos dados envolvidos
- Resultado: no mundo de software, objetos são "vivos" não passivos
  - No mundo real, a Conta é passiva, não viva
  - No mundo de software, a Conta faz várias coisas que afetam seus atributos
    - Saque
    - Depósito
    - Informar seu saldo
    - etc.

Programas Adicionais para o Aluno Estudar:

- Banco3.java
- Banco6.java
- Banco7.java
- Banco9.java
- Banco10.java
- Carta1.java
- Carta2.java
- Carta3.java
- Data1.java
- Data2.java

- Data3.java
- Data4.java

Documentação para Melhor Entender o Package p1

- Aqui está o ponto de entrada para a documentação do package p1

oo-1 programa próxima