Universidade Federal do Cariri

OCULTAÇÃO DE INFORMAÇÃO (INFORMATION HIDING)

Profª Paola Accioly
Material baseado nos slides de Thaís Alves Burity Rocha

Aula passada vimos

- Mais detalhes sobre a declaração de métodos
- Passagem de parâmetros
- Modificador static

E na aula de hoje?

```
public class Teste {

        public static void main(String[] args) {
                System.out.println("Olá, mundo!");
        }

}
```

Finalmente vamos entender o que é o **public**

Introdução

- Isso é possível, mas não é desejável...

```
Conta minhaConta = new Conta("11139-2", 500);
minhaConta.saldo = -1000;
```

  - O sistema bancário não admite saldo negativo

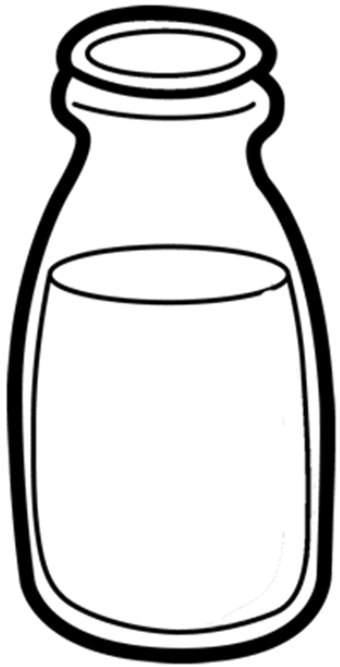
- **Idealmente o saldo só deveria ser alterado através dos métodos de creditar(), debitar() ou transferir()**

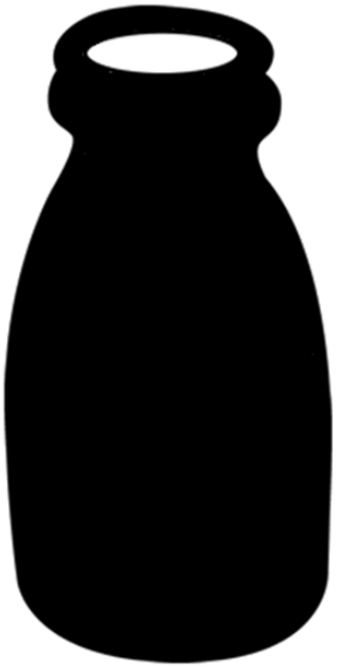
Ocultação de informação

- Information hiding
- Capacidade de ocultar dados dentro de modelos, permitindo que somente operações especializadas os manipulem
- “Esconder” uma ideia, ocultando detalhes internos para o usuário do código
- Em Java, as **classes** são os modelos e é necessário fazer com que o acesso aos **atributos** seja feito somente através de **métodos**

Não confundir com encapsulamento

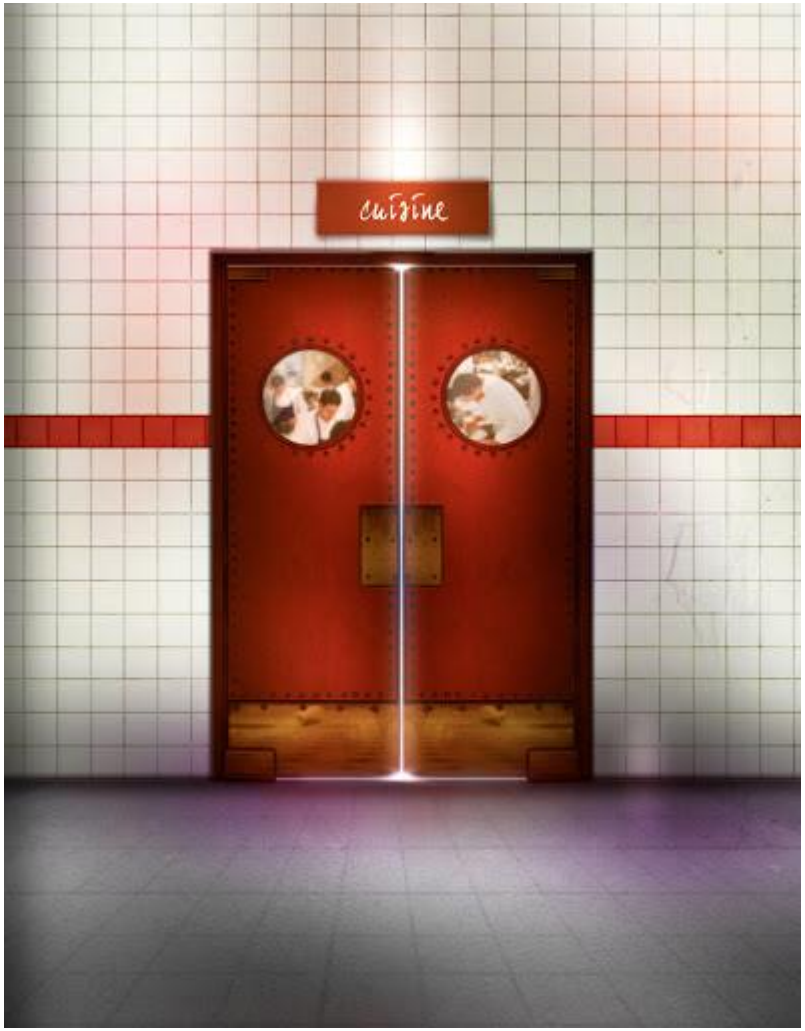
```
Conta minhaConta = new Conta("11139-2", 500);
minhaConta.saldo = 400; //encapsulamento
minhaConta.debitar(100); //ocultação de info.
```





Acessamos o líquido pela boca da garrafa em todo caso, mas aqui não o enxergamos do lado de fora da garrafa

Restaurantes



Mecanismos

- Utilizar **modificadores de acesso** sobre atributos e métodos

- Organizar classes em **pacotes** e utilizar **modificadores de acesso** sobre classes

Modificadores de acesso de membros

- Definem a visibilidade de atributos e métodos
- Retomaremos esse assunto depois de ver **herança**

private	Acesso apenas dentro da classe onde foi declarado
public	Acesso a partir de qualquer classe
default	Acesso apenas por classes no mesmo pacote
~~**protected**~~	~~Default + subclasses (herança)~~

  - Quando não é especificado pacote, é como se as classes estivessem no mesmo pacote

Telefones

+

-

#


Panasonic
ECO
12:00AM
Dec.31
MENU
VOL.
REDIAL
CID
TALK
OFF
1
2ABC
3DEF
4GHI
5JKL
6MNO
7PQRS
8TUV
9WXYZ
0OPER
#
SP-PHONE
INTERCOM
FLASH/CALL WAIT

Exercício

O código está correto?

```
class Conta{
  public String titular;
  String numero;
  private double saldo;

  private void creditar(){}

  public void debitar(){}

  double consultarSaldo(){}
}
```

1

```
class MeuPrograma {

  public void testar(){
    Conta conta = new Conta();
    conta.creditar();
    conta.debitar();
    double s1 = conta.consultarSaldo();
    double s2 = conta.saldo;
    String nome = conta.titular;
    String numeroConta = conta.numero;
  }

}
```

2

Resposta

```
class Conta{
  public String titular;
  String numero;
  private double saldo;

  private void creditar(){}

  public void debitar(){}

  double consultarSaldo(){}
}
```

1

```
class MeuPrograma {

  public void testar(){
    Conta conta = new Conta();
    conta.creditar();
    conta.debitar();
    double s1 = conta.consultarSaldo();
    double s2 = conta.saldo;
    String nome = conta.titular;
    String numeroConta = conta.numero;
  }

}
```

2

Controle de acesso à variável local

```
public int somar (int a, int b){
    private int resultado = a + b;
    return resultado;
}
```

```
public int somar (int a, int b){
    int resultado = a + b;
    return resultado;
}
```

resultado é uma variável local

Como ocultar informação?

- Preferencialmente, defina atributos **private**
- Acesso de leitura à um atributo **x**
  - Método **public <tipo x> getX()** que retorna **x**
  - No caso de boolean, **public boolean isX()**
- Acesso de escrita à um atributo **x**
  - Método **public void setX(x)** que modifica o valor de **x** com base no parâmetro recebido

Como ocultar informação?

```
class Cliente {
   private String nome;
   private String cpf;

   public String getNome(){
      return nome;
   }

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

E se o projeto mudar?

```
class Cliente {
   private String[] dados;

   public String getNome(){
      return dados[0];
   }

   public void setNome(String nome){
      this.dados[0] = nome;
   }
}
```

```
//0 nome
//1 cpf
//2 telefone
```

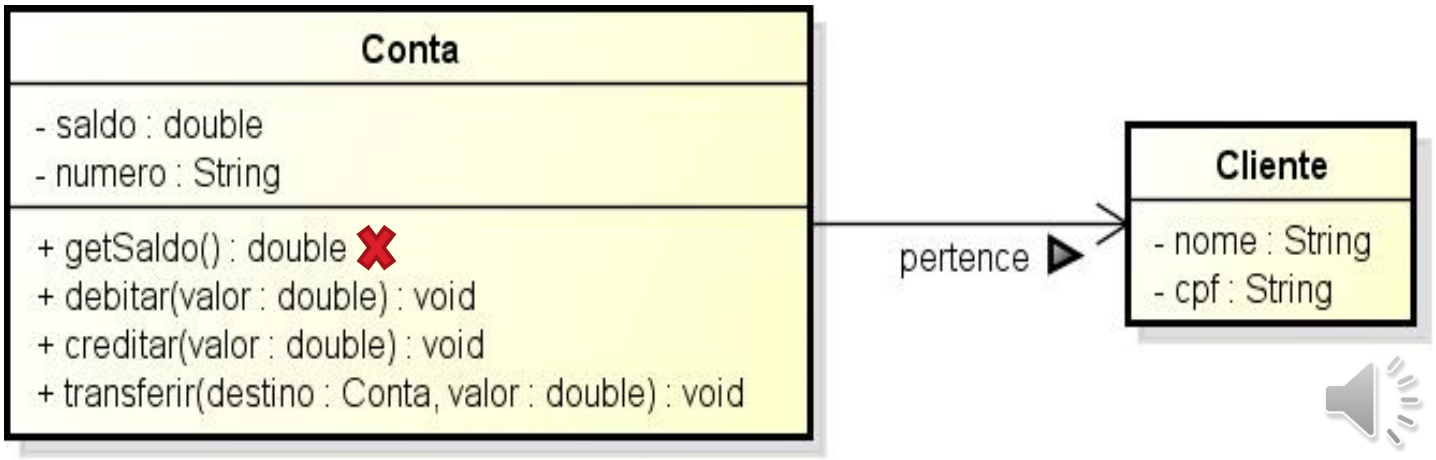
O que acontece com quem usa Cliente?

```
class RepositorioContasVetor {
  private Conta[] contas;
  ...
  public Conta consultar(Cliente cliente){
    Conta conta = null;
    for(int i=0; i<contas.length; i++){
      String nome = c[i].getCliente().getNome();
      if(cliente.getNome().equals(nome)){
         conta = c;
      }
    }
    return conta;
  }
}
```

Programas com menos erros, mais claros e mais fáceis de manter

Boas práticas

- Nunca crie um método sem necessidade
- As vezes **getters** e **setters** podem ser substituídos por outros métodos
  - Na classe **Conta**, não faz sentido existir **setSaldo()**
  - Já temos **creditar()**, **debitar()** e **transferir()**
- Não represente getters e setters no diagrama de classes



Pacotes

- Mecanismo para organizar classes, tal como uma estrutura de diretório
- Delimita um espaço de nomes (namespace)
- Não é possível existir duas classes com o mesmo nome em um mesmo pacote
- O nome completo de uma classe é o **nome do pacote + nome atribuído à classe**
  - Exemplo: java.util.Scanner

Declaração de pacotes

- A declaração do pacote deve ser a primeira coisa no arquivo de código-fonte que contém a classe

```
package pack;

class Conta{

    double saldo;

}
```

Qual é o nome da classe?

Nome de pacote

- Convenção: Iniciar com o nome de domínio da empresa invertido
- Exemplo
  - UFCA: **`www.ufca.edu.br`**
  - Pacote: **`br.edu.ufca.iu`**
  - Classe: **`br.edu.ufca.iu.TelaLogin.java`**

Como usar pacotes

- Na IDE, quando não são definidos pacotes explícitamente, todas as classes são mantidas em um único pacote (default)
- É uma boa prática definir pacotes
- É recomendável que as classes em um mesmo pacote estejam relacionadas entre si
- Exemplo
  - Classes que fazem interface com o usuário
  - Classes de repositório

Modificadores de acesso de classes

- Definem a visibilidade da classe
- public
  - A classe é acessível a partir de qualquer outra
  - A classe pode ser instanciada
  - A classe pode ser estendida (herança)
  - Os métodos e atributos definidos pela classe podem ser acessados, dependendo de seus modificadores de acesso
- default
  - Quando não é especificado
  - Significa acesso em nível de pacote

Exemplo

```
package pack1;

public class Conta{
   String numero;
   double saldo;
   pack2.Cliente cliente;

   void creditar(){...}

   void debitar(){...}
}
```

```
package pack2;

public class Cliente{
   String cpf;
   double nome;
}
```

Declarações import

Já usamos

- Recomendável para uma classe referenciar outra em um pacote diferente
- Possibilita usar nome abreviado da classe
- Importação de uma classe de um pacote

```
import <nome do pacote>.<nome da classe>;
```

- Importação de todas as classes de um pacote

```
import <nome do pacote>.*;
```

- Não é obrigatório usar import
- Não é necessário em se tratando de classes no pacote **java.lang**, como **System**

Exemplo

```
package pack1;

import pack2.Cliente;

public class Conta{
   String numero;
   double saldo;
   Cliente cliente;

   void creditar(){...}

   void debitar(){...}
}
```

Antes da declaração da classe

O código está correto?

```
package pack1;
import pack2.Cliente;

class Conta{
   String numero;
   double saldo;
   Cliente cliente;

   void creditar(){}

   void debitar(){}
   ...
}
```

```
package pack2;
import pack1.Conta;

public class Cliente{
   String cpf;
   String nome;
   Endereco endereco;
   Conta conta;
}
```

```
package pack2;

class Endereco{
   String rua;
   String bairro;
   String cep;
}
```

Resposta

```
package pack1;
import pack2.Cliente;

class Conta{
   String numero;
   double saldo;
   Cliente cliente;

   void creditar(){}

   void debitar(){}
   ...
}
```

```
package pack2;
import pack1.Conta;

public class Cliente{
   String cpf;
   String nome;
   Endereco endereco;
   Conta conta;
}
```

```
package pack2;

class Endereco{
   String rua;
   String bairro;
   String cep;
}
```

static import

- Notação comum

```
import java.lang.Math;
public class StaticImportTest {
  public static void main (String[] args){
    System.out.println(Math.sqrt(900.0));
    System.out.println(Math.log(Math.PI));
  }
}
```

- Notação alternativa

```
import static java.lang.Math.*;
public class StaticImportTest {
  public static void main (String[] args){
    System.out.println(sqrt(900.0));
    System.out.println(log(PI));
  }
}
```

Em resumo

- Encapsulamento
- Mecanismos
  - Modificadores de acesso
  - Pacotes e comando import