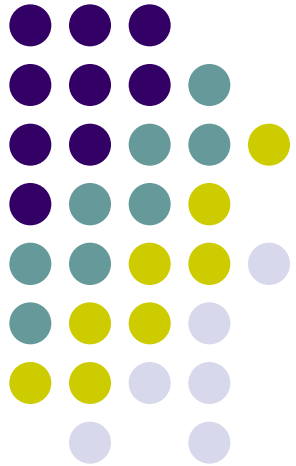
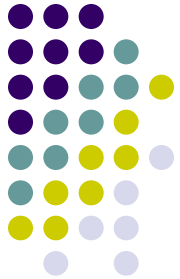
COM220

Aula 11: Pacotes e Interfaces

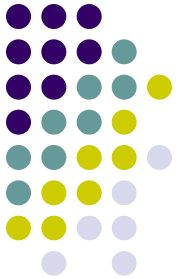
Prof. Laércio Baldochi



Pacotes



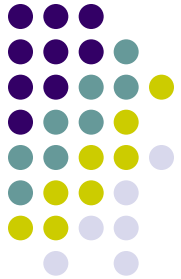
- A construção de qualquer programa Java requer a utilização de bibliotecas pré-existentes na API Java
- Cada classe e interface pertence a um pacote específico
  - Ex: JOptionPane: javax.swing.*;
- Devemos, nas nossas aplicações, utilizar a mesma abordagem
  - Usando pacotes conseguimos gerenciar melhor a complexidade dos componentes das nossas aplicações



Pacotes

- Facilitam também o reuso de código
  - Podemos importar apenas uma classe de um pacote, ou o pacote como um todo
  - Ex:
    - import javax.swing.JOptionPane // só uma classe
    - import javax.swing.* // o pacote todo

Pacotes



- Estrutura
  - Os pacotes possuem uma definição física representada por uma estrutura hierárquica através de diretórios no sistema operacional
- Voltando ao exemplo da aula 9

info

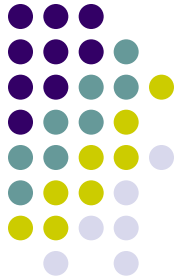
academico

disciplina -> Disciplina

pessoa -> (Pessoa, Professor e Aluno)

geral -> Util

Pacotes

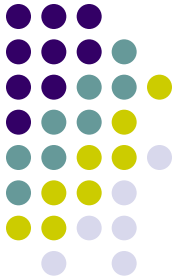


- Para declarar um pacote utilizamos a palavra reservada package antes da definição da classe

Definição do pacote

```
package info.geral;
public class Util {

  public static final String MASCULINO = "M";
  public static final String FEMININO = "F";

  public static final int MIN_IDADE = 10;
  public static final int MAX_IDADE = 120;

  public static final String GRADUACAO = "Graduação";
  public static final String ESPECIALIZACAO = "Especialização";
   public static final String MESTRADO = "Mestrado";
  public static final String DOUTORADO = "Doutorado";

  public static final String CC = "Ciências da Computação";
  public static final String SI = "Sistemas de Informação";
}
```

Pacotes



- Cada arquivo fonte (.java) deve estar no mesmo diretório identificado na linha de definição do package (ex: c:\...\src\info\geral para o caso da classe Util);
- Para o projeto elaborado na aula 9 teríamos:

c:\...\src\info\academico\disciplina\Disciplina.java

c:\...\src\info\academico\pessoa\Pessoa.java

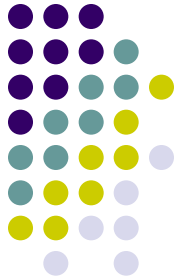
c:\...\src\info\academico\pessoa\Aluno.java

c:\...\src\info\academico\pessoa\Professor.java

c:\...\src\info\geral\Util.java

c:\...\src\ExemploClasse.java

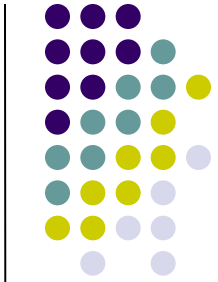
Pacotes



- Agora para que seja possível a compilação devemos realizar os ***imports*** correspondentes
- Na classe ExemploClasse deve-se importar os pacotes e classes a seguir

```
import info.geral.Util;
import info.academico.disciplina.*;
import info.academico.pessoa.*;
```

Exercício 1



- Altere os arquivos fonte do exemplo utilizado na aula 9
  - Não esqueça de salvar cada arquivo fonte dentro do diretório adequado
  - Exemplo:

c:\...\src\info\academico\disciplina\Disciplina.java

c:\...\src\info\academico\pessoa\Pessoa.java

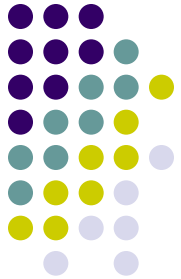
c:\...\src\info\academico\pessoa\Aluno.java

c:\...\src\info\academico\pessoa\Professor.java

c:\...\src\info\geral\Util.java

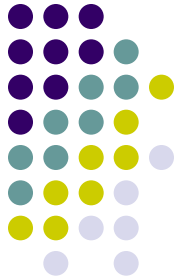
c:\...\src\ExemploClasse.java

Interfaces Java



- Definem um conjunto de métodos que uma classe deverá implementar
  - Coleção de métodos abstratos e constantes
- Separam o comportamento de um objeto de sua implementação concreta
- Dessa forma, a interface age como um contrato, o qual define explicitamente quais métodos uma classe deve obrigatoriamente implementar

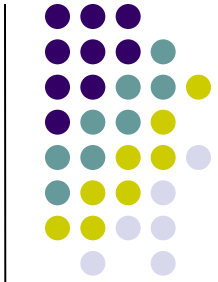
Interface X Classe



- Interface
  - Não pode ser instanciada
  - Pode ser implementada por diversas classes
    - Polimorfismo

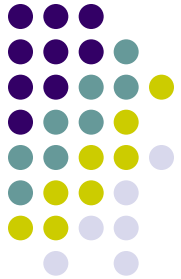
- Classe
  - Pode implementar diversas interfaces
  - Quando implementa uma interface pode implementar também outros métodos

Interface



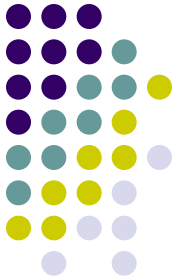
- Métodos em uma interface são sempre públicos e abstratos
  - public abstract
- Dados em uma interface são sempre constantes públicas
  - public static final

Interface
Definição



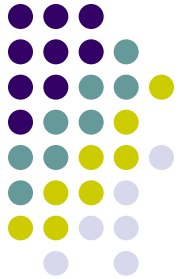
- Uma interface é uma coleção de constantes e métodos abstratos, não fornecendo nenhuma implementação

Interface



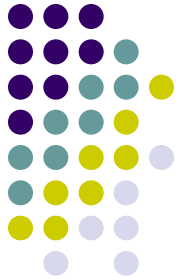
- Para utilizar uma interface
  - Implementa-se a mesma em uma classe
  - Dentro da classe, deve-se escrever o código para cada um dos métodos declarados na interface implementada

Interface
Exemplo



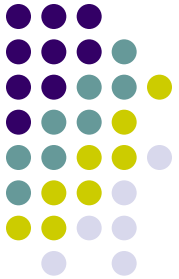
```
public interface FormaGeometrica {
    public abstract double area();
    public abstract double volume();
    public abstract String getNome();
}

public class Ponto implements FormaGeometrica {
    private int x,y;
    public void setPonto(int x1, int y1) {...}
    public int getX() {...}
    public double area(){ return 0.0; }
    public double volume() { return 0.0; }
    public String getNome() { return "Ponto"; }
}
```

Interface Exemplo



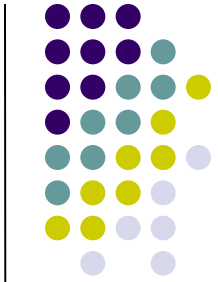
```
public class Circulo implements FormaGeometrica {
   Ponto centro;
   private int raio;
   public void setRaio(int r) {...}
   public int getRaio() {...}
   public void setCentro(Ponto c) {...}
   public Ponto getCentro() {...}
   public double area(){ return Math.PI * raio * raio; }
   public double volume() { return 0.0; }
   public String getNome() { return "Circulo"; }
}
public class Cilindro implements FormaGeometrica {
   private int altura;
   private Circulo circulo;
   ...
   public double area(){
       return 2 * super.area() + 2 * Math.PI * raio * altura;
   }
   public double volume() { return super.area() * altura; }
   public String getNome() { return "Cilindro"; }
}
```

Interface

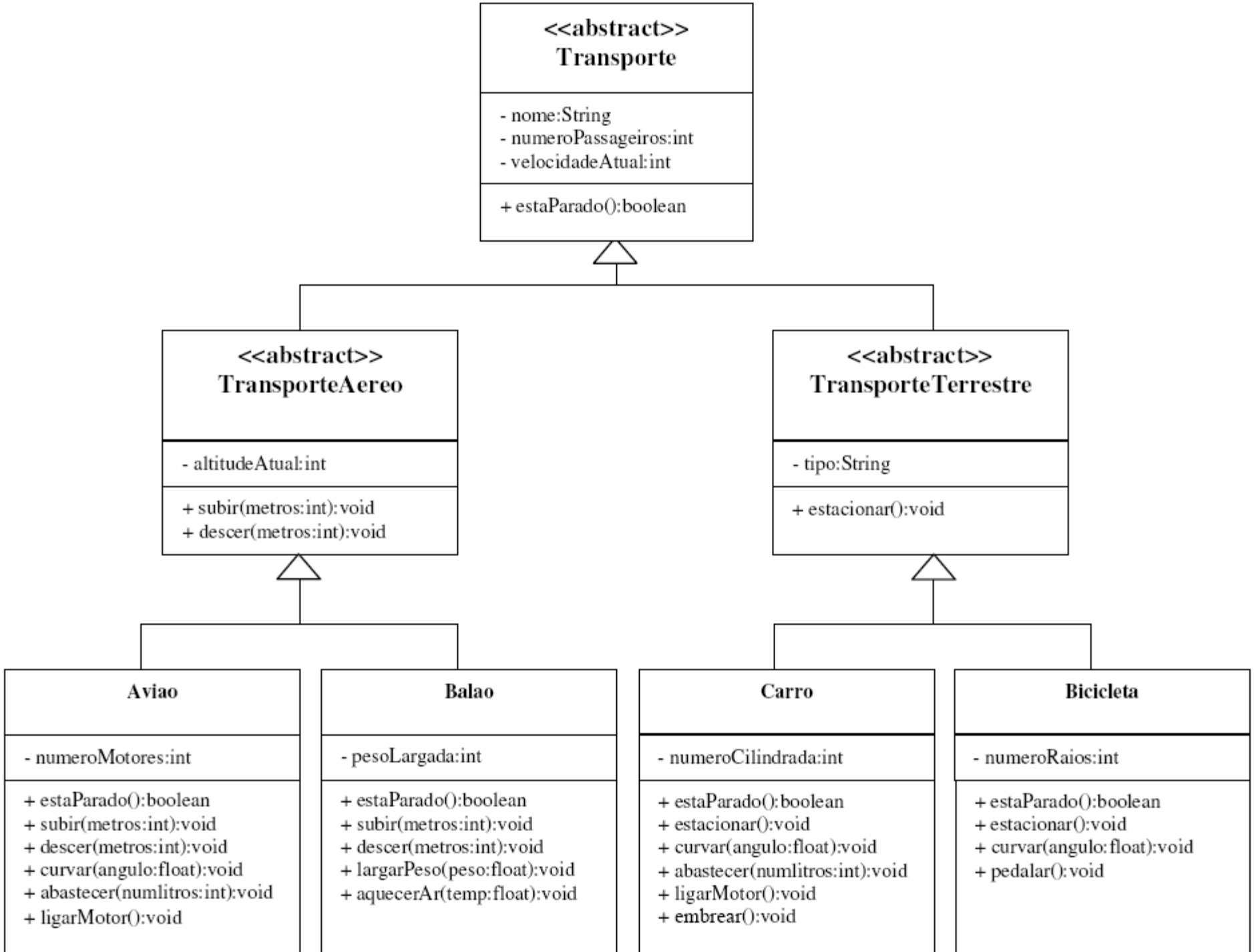


- Resumindo
  - Uma interface define um conjunto de métodos que outras classes devem implementar, mas não define como esses métodos devem ser implementados

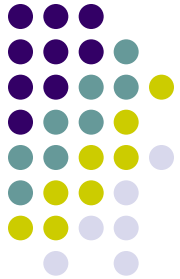
Exercício 2



- Considere o diagrama UML mostrado a seguir


<<abstract>>
Transporte
- nome:String
- numeroPassageiros:int
- velocidadeAtual:int
+ estaParado():boolean
<<abstract>>
TransporteAereo
- altitudeAtual:int
+ subir(metros:int):void
+ descer(metros:int):void
<<abstract>>
TransporteTerrestre
- tipo:String
+ estacionar():void
Aviao
- numeroMotores:int
+ estaParado():boolean
+ subir(metros:int):void
+ descer(metros:int):void
+ curvar(angulo:float):void
+ abastecer(numlitros:int):void
+ ligarMotor():void
Balao
- pesoLargada:int
+ estaParado():boolean
+ subir(metros:int):void
+ descer(metros:int):void
+ largarPeso(peso:float):void
+ aquecerAr(temp:float):void
Carro
- numeroCilindrada:int
+ estaParado():boolean
+ estacionar():void
+ curvar(angulo:float):void
+ abastecer(numlitros:int):void
+ ligarMotor():void
+ embrear():void
Bicicleta
- numeroRaios:int
+ estaParado():boolean
+ estacionar():void
+ curvar(angulo:float):void
+ pedalar():void

Exercício 2
Entrega: 16/04



a) Crie uma interface de nome *Motorizado* onde são declarados os métodos *void ligarMotor()* e *void abastecer(int numLitros)*.

b) Implemente a interface *Motorizado* nas classes *Aviao* e *Carro*.

c) Crie uma interface de nome *Conduzivel* onde é declarado o método *void curvar(float angulo)*.

d) Implemente a interface *Conduzivel* nas classes *Aviao*, *Carro* e *Bicicleta*.

e) Escreva um programa de teste que crie dois ArrayList. O primeiro (A1) deve conter instâncias de objetos motorizados. O segundo (A2) deve conter instâncias de objetos conduzíveis. Processe A1 ligando o motor das instâncias. Processe A2 curvando as instâncias (o método curvar deve imprimir o ângulo da curva).