Programação Orientada por Objectos

Visibilidade e Pacotes

Prof. Rui César das Neves, Prof. José Cordeiro

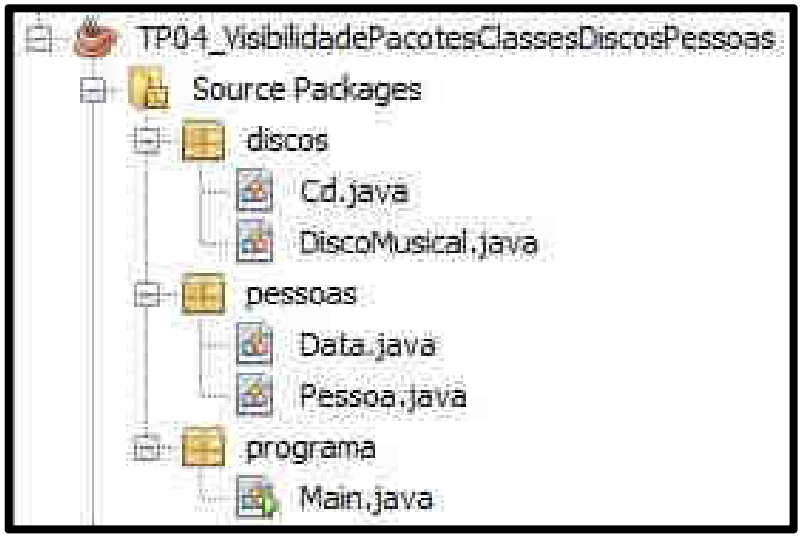
Departamento de Sistemas e Informática
Escola Superior de Tecnologia de Setúbal – Instituto Politécnico de Setúbal
2014/2015

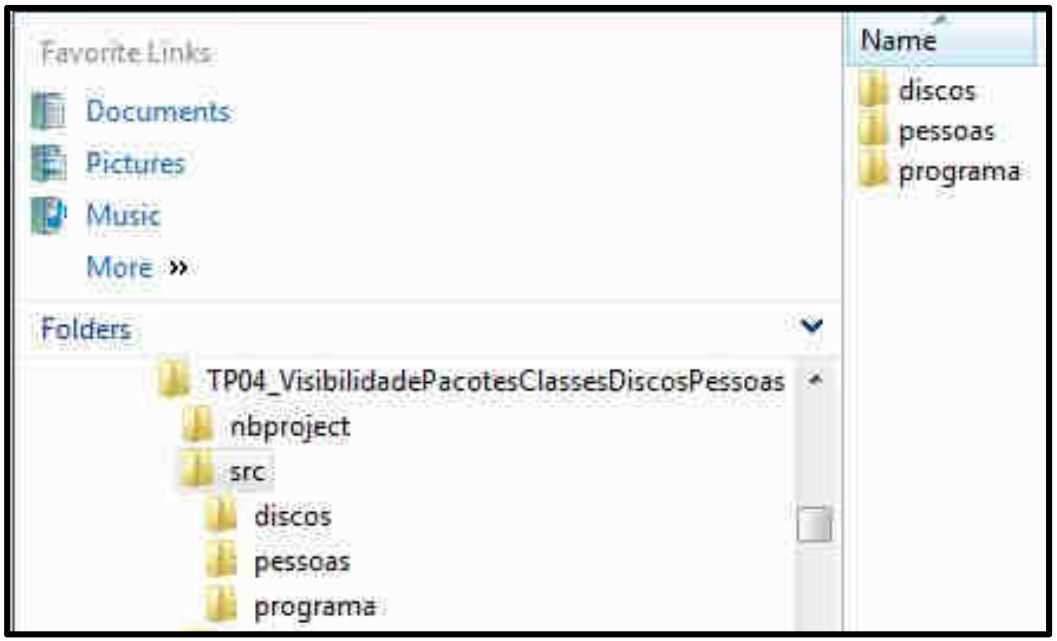
Sumário

- **Noções de Visibilidade e Pacotes:**
  - Utilização de pacotes em Java
  - Visibilidade de Membros de instância
    - Modificadores de acesso:
      - private,
      - nenhum (package),
      - protected
      - public
  - Visibilidade de Classes
    - Modificadores de acesso:
      - public
      - nenhum (package)
  - Visibilidade Crescente ao descer numa hierarquia

Visibilidade e Pacotes - Package

- **package** – palavra reservada do java usada para agrupar um conjunto de classes.
  - Normalmente agrupa-se um conjunto de classes de alguma forma relacionadas entre si, com o objetivo de modularizar e reutilizar o código.
  - Ex: package discos;
    package pessoas;





Visibilidade e Pacotes - Import

```
package pessoas;
public class Data {
  private int dia, mes, ano;
  public Data(int dia, int mes, int ano) {
    this.dia = dia;
    this.mes = mes;
    this.ano = ano;}
    // [geters & setters] … publicos

    public String toString(){
      return "DATA: " + ano + "/" + mes +
              "/" + dia;
    }
}
```

```
package pessoas;
public class Pessoa {
  private String nome; private Data dataNascimento;
  public Pessoa(String nome,Data dataNascimento) {
    this.nome = nome;
    this.dataDeNascimento = dataDeNascimento;}
    // [geters & setters] … publicos

    public String toString(){
      return "NOME: " + nome +
    " DATA DE NASCIMENTO" +
          dataDeNascimento.toString();
   }
}
```

```
// a classe DiscoMusical pertence ao
// pacote discos
package discos;

//Para utilizar as classes de outro
pacote de fazer import:
import pessoas.Data;
import pessoas.Pessoa;
//Alternativamente, podemos importar
"todo" o pacote (evitar esta utilização
para não enchermos o ambiente de nomes
desnecessários e com isso podermos ter
colisões de nomes que teremos que
resolver explicitamente):
//import pessoas.*;

public class DiscoMusical {
  private String titulo;
  private Data dataGravacao;
  private Pessoa autor;


  //... Métodos da classe ...

}
```

Visibilidade e Pacotes - Membros

Modificadores de Acesso

- **private** – os membros só são **acessíveis no interior da própria classe.**

- **nenhum** – os membros são acessíveis no interior da própria classe **e de classes do mesmo package.**
  - Tornam-se públicos para todas as classes do pacote.

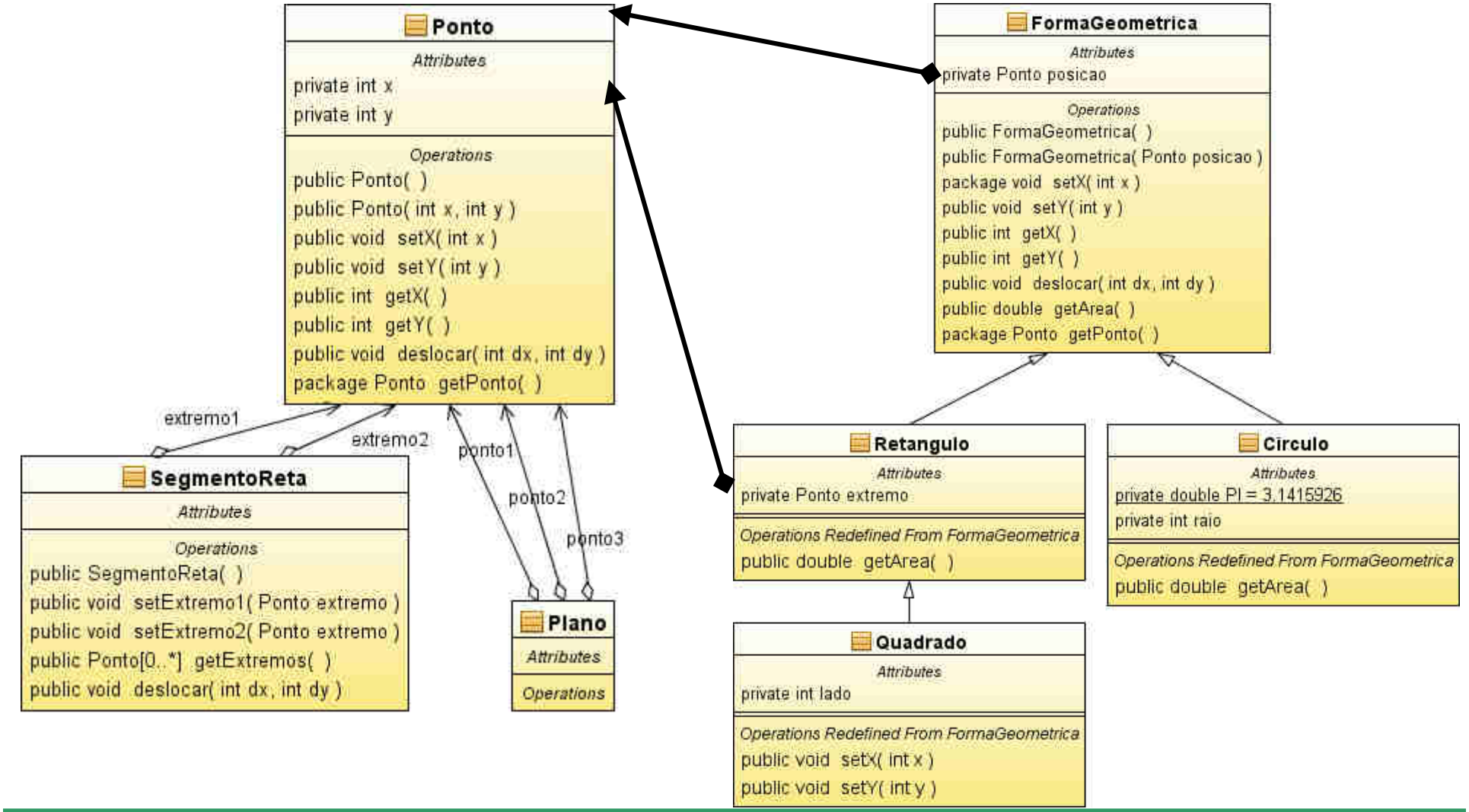
- **protected** – os membros são acessíveis no interior da própria classe, de classes do mesmo package **e das classes que herdem direta ou indiretamente desta classe.**
  - Tornam-se públicos para todas as classes do pacote e
  - Para classes de outros pacotes que herdem desta classe.

- **public** – os membros são acessíveis a qualquer classe

Visibilidade e Pacotes - Membros

No **package primitivas** temos:

No **package formas** temos:



Visibilidade e Pacotes - Membros

- **private**
  - Todos os membros declarados como private em qualquer das 7 classes do anterior slide são visíveis apenas dentro de cada uma delas.
  - Ex:
    - Os atributos **x e y de Ponto só são acessíveis**/visíveis **dentro da classe Ponto.**
    - Os atributos **extremo1 e extremo2 de SegmentoReta só são acessíveis**/visíveis **dentro da classe SegmentoReta**
    - O atributo **posicao só é acessível dentro de FormaGeometrica**
    - O atributo **extremo só é acessível dentro de Retangulo**

Visibilidade e Pacotes - Membros

package (sem modificador)

- Os membros declarados em qualquer das 7 classes exemplificadas só são acessíveis/visíveis nas restantes classes do mesmo package.

- Ex: Em **FormaGeometrica** ao omitir o "public" e definir o método:

```
Ponto getPonto(){
    return new Ponto (posicao.getX(), posicao.getY());
}
```

  - Ele só é visível para FormaGeometrica, Retangulo, Quadrado e Circulo.
  - **Pergunta**: Porque é que não usámos aqui o getPonto() da Classe Ponto?

- Ex: Em **Ponto** se omitissemos o "public" e definissemos o método:

```
void deslocar( int dx, int dy ) {
    x += dx;  y += dy;  }
```

  - Ele só seria visível pelas classes Ponto, SegmentoReta e Plano.
  - Pergunta: Como faríamos para deslocar objetos de FormaGeometrica?

Visibilidade e Pacotes - Membros

protected

- Os membros declarados em qualquer das 7 classes exemplificadas são acessíveis/visíveis nas restantes classes do mesmo package e por quaisquer classes de outros packages que delas herdassem.

- Ex: Se em **FormaGeometrica** declarássemos o método:

```
protected int[] retornaPonto() {
    return posicao.getPonto();  }
```

  - Ele seria visível pelas classes FormaGeometrica, Retangulo, Quadrado, Circulo e ainda por quaisquer outras classes que herdassem de FormaGeometrica

- Ex: Se em **Ponto** declarássemos o método:

```
protected void deslocar( int dx, int dy )  {
    x += dx;  y += dy;  }
```

  - Ele seria visível pelas classes Ponto, SegmentoReta, Plano e ainda por quaisquer outras classes que herdassem de Ponto

Visibilidade e Pacotes - Membros

- **public**
  - Os membros declarados em qualquer das 7 classes exemplificadas são acessíveis/visíveis em qualquer das restantes classes.
  - Ex: O método público da classe **Ponto**:

```
public void deslocar( int dx, int dy ) {
    x += dx;
    y += dy;
}
```

    - É visível pelas restantes seis classes!

Visibilidade e Pacotes - Membros

- **Comparação com a linguagem C# (mais próxima da utilizada em UML):**

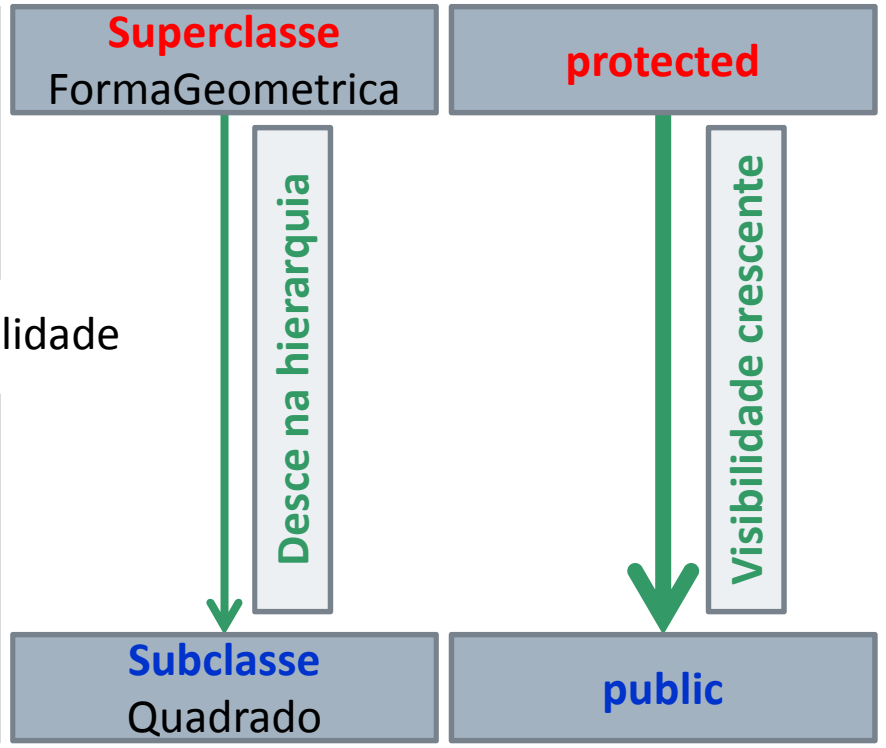
	private	*nenhum*	protected	public	Internal	protected internal
Java	Classe	Classe Package	Classe Derivadas Package	Total		
C#	Classe	*o mais restritivo**	Classe Derivadas	Total	Assembly	Classe Derivadas Assembly

*-O mais restritivo será: **private** para membros e **internal** para classes

Visibilidade e Pacotes - Membros

- **Note que:**
  - **A Visibilidade Crescente ao descer numa hierarquia de classes é permitida**
  - Ex: Posso ter um método protegido em FormaGeometrica:

```
protected void setX(int x) {
    posicao.setX(x);
}
```



  - E redefini-lo em Quadrado com maior visibilidade

```
@Override
public void setX(int x) {
    super.setX(x);
    setX2(getX() + lado);
}
```

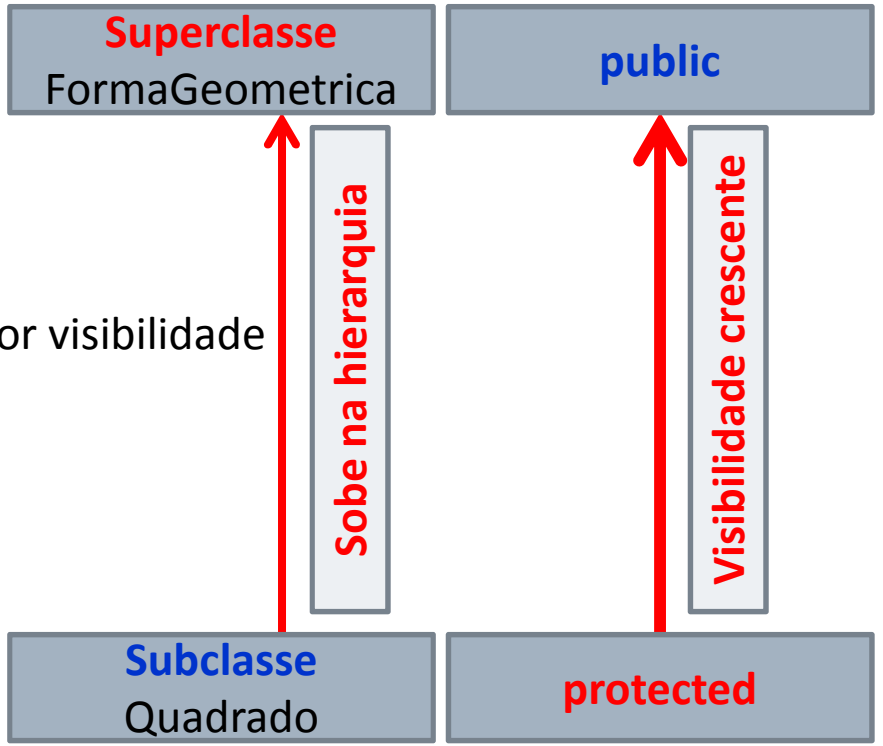
Visibilidade e Pacotes - Membros

- **Note que:**
  - **A Visibilidade Crescente ao subir numa hierarquia de classes não é permitida**
    - Ex: Não posso ter um método público em FormaGeometrica:

```
public void setX(int x) {
      posicao.setX(x);
}
```

    - E tentar redefini-lo em Quadrado com menor visibilidade

```
protected void setX(int x) {
      super.setX(x);
      setX2(getX() + lado);
  }
```



Visibilidade e Pacotes - Classes

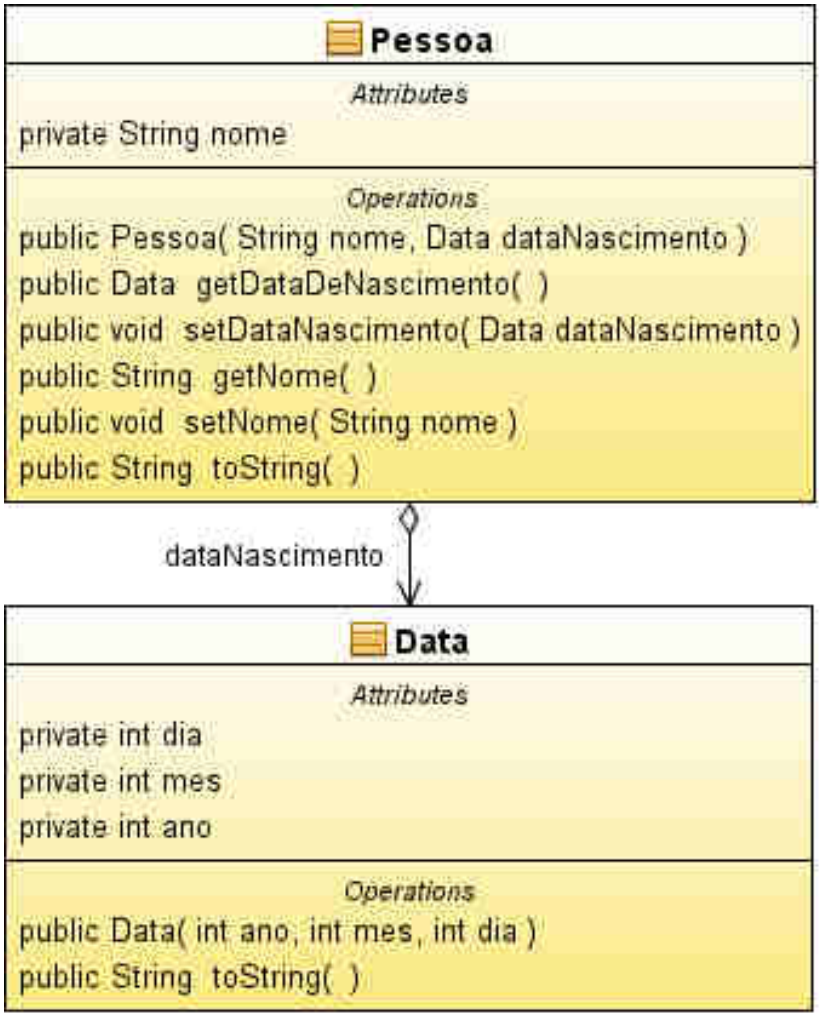
Modificadores de Acesso - Classes

- private – Usado em classes internas - *inner* classes (é um membro)
- **nenhum** – Quando a definição da classe não é antecedida por nenhum modificador **é visível por qualquer classe dentro do pacote** mas não pode ser usada num pacote externo
  - não é possível fazer **import** desta classe**.**
  - Dizemos que a classe tem visibilidade **package**
  - EX: **class Pessoa** { //membros da classe }
- protected – Usado em classes internas - *inner* classes (é um membro)
- **public** – A classe **é visível** por qualquer outra classe, **dentro ou fora do pacote a que pertence** podendo assim ser usada num pacote externo desde que se faça o respetivo **import.**
  - Inserir import pessoas.Data;
  - EX: **public class Data** { // membros da classe }

Visibilidade e Pacotes - Package

No **package discos** temos:

No **package pessoas** temos:



Visibilidade e Pacotes - Classes

```
package pessoas;
public class Data {
  private int dia, mes, ano;
  public Data(int dia, int mes, int ano) {
    this.dia = dia;
    this.mes = mes;
    this.ano = ano;}
    // [geters & setters] … publicos

    public String toString(){
      return "DATA: " + ano + "/" + mes +
              "/" + dia;
    }
}
```

```
package pessoas;
class Pessoa { // sem modificador de acesso
  private String nome; private Data dataNascimento;
  public Pessoa(String nome,Data dataNascimento) {
    this.nome = nome;
    this.dataDeNascimento = dataDeNascimento;}
    // [geters & setters] … publicos

    public String toString(){
      return "NOME: " + nome +
    " DATA DE NASCIMENTO" +
          dataDeNascimento.toString();
   }
}
```

```
// a classe DiscoMusical pertence ao
// pacote discos
package discos;

//consegue importar (ver) a classe Data
//do pacote pessoas porque ela é pública
import pessoas.Data;
// Mas não consegue importar a classe
// Pessoa porque ela é "package"
public class DiscoMusical {
  private String titulo;
  private Data dataGravacao;

  public DiscoMusical() {
    titulo = "ND";
    dataGravacao = new Data(2009,10,19);}

    public DiscoMusical(String titulo,
                        Data dataGravacao) {
      this.titulo = titulo;
      this.dataGravacao = dataGravacao;
    }
    // [geters & setters] … publicos

    public String toString(){
        return "TITULO: " + titulo +
          " " + dataGravacao.toString();
    }
}
```

Visibilidade e Pacotes - Classes

```
package programa;
import discos.cd;
import pessoas.Data;

public class Main {

  public static void main(String[] args) {

    // Podemos criar um objeto da classe Data
    // porque, apesar de estar noutro pacote,
    // foi declarada como pública

    Data dataDisco = new Data (2011, 9, 25);

    // A Classe Cd do pacote discos
    // consegue ver, e portanto usar,
    // objetos da classe Data
    // porque esta, apesar de estar noutro
    // pacote, foi delarada como pública


    Cd disco = new
        Cd("Canções", dataDisco, "Cantor");

    System.out.println (disco.toString());
  }
}
```

```
package programa;
//import pessoas.Pessoa;
import pessoas.Data;

public class Main {

  public static void main(String[] args) {




    Data dataNascimento = new Data(28,11,1999);

    // a classe Pessoa, por ser "package“,
    // só é visível dentro do package pessoas
    // (o package a que pertence)
    // não conseguimos fazer "import“
    // desta classe a partir do package
    // programa

    //Pessoa ana;
    //ana = new Pessoa (“Ana",dataNascimento);

    //System.out.println (ana.toString());
  }
}
```

Resumindo

- **Noções de Visibilidade e Pacotes:**
  - Modificadores de Acesso – **Membros**
    - private – são visíveis só dentro da classe,
    - nenhum – são visíveis só dentro do pacote,
    - protected – são visíveis dentro do pacote + subclasses externas ao pacote
    - public – são visíveis por quaisquer classes
  - Modificadores de Acesso – **Classes**
    - **public** - são visíveis por quaisquer outras classes
    - **nenhum** – são visíveis apenas dentro do seu pacote
  - Ao descer numa hierarquia de classes a visibilidade de uma redefinição pode manter-se ou crescer (tornar-se mais visível), mas nunca decrescer (tornar-se menos visível), para se garantir a aplicabilidade do "princípio de substituição".

Leitura Complementar

- Modificadores de Acesso
  - Capítulo 3
    - pgs 81 a 84