Programação Orientada a Objetos e a Linguagem Java

Rodrigo da Cruz Fujioka
fujiokabr@gmail.com

Package

- Antes de definirmos os especificadores de acessos precisamos conhecer o conceito de pacote

– Um conjunto de classes sobre algum domínio

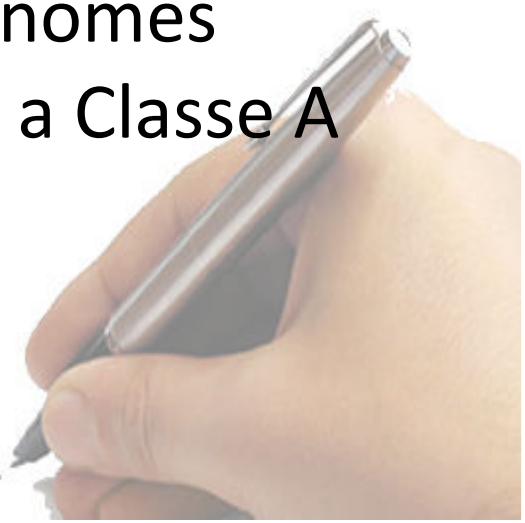
– Ex: biblioteca de elementos gráficos, biblioteca matemática

- import

```
import java.util.*;
```

– Mecanismo para gerenciar espaço de nomes

– Através do import podemos distinguir a Classe A da biblioteca X da classe A da biblioteca Y



Package

- Um arquivo fonte implica em uma unidade de compilação
  - extensão “.java”
  - uma única classe pública, outras são de suporte
  - compilação gera “.class”
- Em cada arquivo fonte devemos definir a qual pacote as classes daquele arquivo pertencem.

Package

- Um arquivo fonte implica em uma unidade de compilação
  - extensão “.java”
  - uma única classe pública, outras são de suporte
  - compilação gera “.class”
- Em cada arquivo fonte devemos definir a qual pacote as classes daquele arquivo pertencem.

Package


```
package figuras.quadrilateros;
public class Quadrado {
}
```

```
package figuras.quadrilateros;
public class Retangulo {
}
```

Package

- Quando não definimos um pacote, as classes pertencerão ao *package default, que é ormado* por todas as classe do “diretório corrente”

Package

- A estrutura de diretórios na qual uma classe se encontra deve respeitar o pacote definido para ela. Por exemplo, seja o pacote:

```
package figuras.quadrilateros;
public class Quadrado {
}
```

- A classe Quadrado deve estar no diretório:

```
...\figuras\quadrilateros
```

Modificadores de acesso e **visibilidade**

- Em ordem crescente de acesso
  - private
  - “package-private”
    - modificador ausente
  - protected
  - public

public

- Acessível
  - na própria classe
  - nas subclasses
  - nas classes do mesmo pacote
  - em todas as outras classes

Classe
+campoPublico: tipo
+metodoPublico: tipo

- Use para
  - construtores e métodos que fazem parte da interface do objeto
  - métodos estáticos utilitários
  - constantes (estáticas) utilitárias
  - é raro ter atributos públicos, mas é comum com métodos
- Evite usar em
  - construtores e métodos de uso restrito
  - campos de dados (propriedades) de objetos

protected

- Acessível
  - na própria classe
  - nas subclasses
  - nas classes do mesmo pacote
    - A intenção é dar acesso ao programadores que estenderão sua classe

Classe
#campoProt: tipo
#metodoProt: tipo

- Use para
  - construtores que só devem ser chamados pelas subclasses (através de super())
  - métodos que só devem ser usados se forem sobrepostos
- Evite usar em
  - construtores em classes que não criam objetos
  - métodos com restrições à sobreposição
  - campos de dados de objetos

package-private

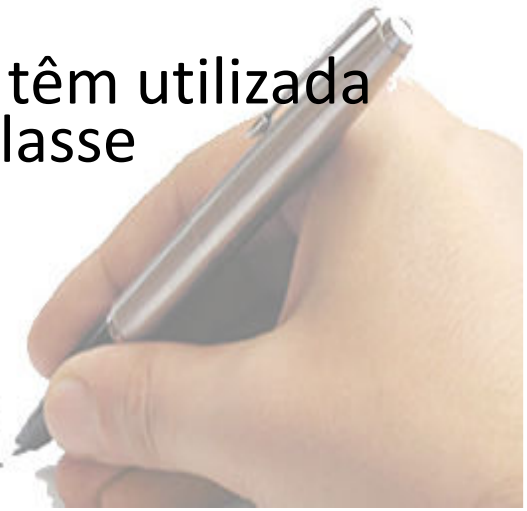
- Modificador ausente
  - se não houver outro modificador de acesso, o acesso será "package-private"
    - é como public, mas apenas dentro do pacote
- Acessível
  - na própria classe
  - nas classes e subclasses do mesmo pacote
- Use para
  - construtores e métodos que só devem ser chamados por classes e subclasses do pacote
  - constantes estáticas úteis apenas dentro do pacote
- Evite usar em
  - construtores em classes que não criam objetos
  - métodos cujo uso externo seja limitado ou indesejável
  - campos de dados de objetos

Classe
~campoAmigo: tipo
~metodoAmigo: tipo

private

Classe
-campoPrivate: tipo
-metodoPrivate: tipo

- Acessível
  - na própria classe (nos métodos, funções estáticas, blocos estáticos e construtores)
- Use para
  - construtores de classes que só devem criar um número limitado de objetos
  - métodos que não fazem parte da interface do objeto
  - funções estáticas que só fazem sentido (ou só têm utilidade) dentro da classe
  - variáveis e constantes estáticas que não têm utilizada ou não podem ser modificadas fora da classe
  - campos de dados de objetos



Sobreposição

- Métodos sobrepostos nunca podem ter **menos** acesso que os métodos originais
  - Se método original for **public**, novas versões têm que ser public
  - Se método original for **protected**, novas versões só podem ser protected ou public
  - Se método original **não tiver modificador de acesso** (é "package-private"), novas versões podem ser declaradas sem modificador de acesso, protected ou public
  - Se método original for **private**, ele não será visível fora da classe e, portanto, jamais poderá ser estendido

Exemplo

Especificador	classe A	subclasse de A	classes pacote	classes mundo
private	ok			
protected	ok	ok	ok	
public	ok	ok	ok	ok
"friendly"	ok	ok*	ok	

Dúvidas?

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?

? ? ? ? ? ? ? ?

Bibliografia

- **PEREIRA**, Frederico C. G.; Slides do Curso de Java. Disponível por WWW em http://asterix.coinfo.cefetpb.edu.br/~fred
- **ROCHA**, Helder da; Curso de Java. Disponível por WWW em http://www.argonavis.com.br.
- **The Java Tutorial**. Disponível por WWW em http://java.sun.com/docs/books/tutorial/
- **DEITEL**, Harvey M.; Paul.J. Java How to Program. 3rd. ed. - Prentice Hall.