Acesso a Classes e Membros de Classes

Java possui especificadores de acesso para permitir ao criador de uma biblioteca de classes especificar o que está disponível para o programador cliente e o que não está. Deste modo o criador da biblioteca pode fazer modificações (melhoramentos) sem afetar o código do programador cliente. Normalmente os campos de dados e alguns métodos não se destinam a ser usados diretamente pelo programador cliente.

Os níveis de acesso são:

- *public*
- *friendly* ou *package* (sem palavra-chave)
- *protected*
- *private*

Acesso a Classes

Alguns especificadores de acesso podem ser usados para determinar que classes de uma biblioteca estão disponíveis para um utilizador dessa biblioteca.

Uma classe pode ter 2 tipos de acesso: público e *package* ou *friendly*.

público: declaradas com a palavra-chave *public*, tornam a classe acessível ao programador cliente a partir de qualquer *package*.

friendly: não se coloca nenhum especificador de acesso, a classe só é acessível de outra qualquer do mesmo *package*, mas não de fora do *package*. No entanto os membros estáticos públicos de uma classe *friendly* são acessíveis de outros *packages*. Este tipo de acesso usa-se para classes que apenas suportam tarefas realizadas por outras classes. Deste modo o criador da classe não necessita de criar documentação e mantém a possibilidade de poder alterá-la porque nenhum programa cliente depende da implementação particular da classe.

Acesso a Membros de Classes

Os especificadores de acesso *public*, *protected* e *private* colocados à frente na definição de cada membro (campo ou método) de uma classe controlam o acesso a esses membros.

Acesso *public*

Um membro de uma classe declarado com o modificador de acesso *public* pode ser acedido por qualquer programa que tenha acesso à classe desse membro.

Acesso *friendly*

É o tipo de acesso por omissão, quando não se especifica nenhum modificador de acesso. Todas as classes no mesmo *package* têm acesso a membros *friendly*, mas classes de outros *packages* não têm acesso.

Acesso *private*

Membros declarados com o modificador de acesso *private* só podem ser acedidos por métodos dentro da própria classe. Normalmente campos de dados e métodos que apenas auxiliam outros métodos são declarados *private* para assegurar que não são usados acidentalmente e assim podem ser modificados sem que afetem outras classes no mesmo *package*.

Acesso *protected*

Um membro declarado com o modificador de acesso *protected* pode ser acedido por todas as classes no mesmo *package* e por classes, mesmo que noutros *packages*, que herdem da classe a que o membro pertence.

Uma classe que herde de outra classe (superclasse) herda todos os campos de dados e métodos dessa superclasse. Os membros privados e *friendly* da superclasse embora herdados não são acessíveis da subclasse, enquanto os membros públicos são acessíveis não só da subclasse como também de todas as outras classes. O modificador de acesso *protected* é útil para permitir acesso às subclasses, sem permitir acesso a todas as outras classes não relacionadas.

No entanto se uma classe B é uma subclasse de A pertencente a um *package* diferente, a classe B pode aceder a membros *protected* (campos de dados ou métodos) de A mas só em objetos do tipo B ou de subclasses de B. A classe B não pode aceder a membros *protected* de A em objetos do tipo A.

Exemplo:

```
package  primeiro;
class  A  {
    protected int i;
    protected void f()  {
        System.out.println("Classe A – metodo protected");
    }
}
```

```
package  segundo;
import  primeiro.*;
class  B  extends  A  {
    public static void main(String args [])
                                throws java.io.IOException  {
        A a = new A();
        B b = new B();
        b.i = 5;
        b.f();
        //  a.i = 5; ilegal
        //  a,f();   ilegal
        System.in.read();
    }
}
```

Encapsulamento

O encapsulamento, característico da programação orientada por objetos, consiste:

- Na inclusão de dados e métodos dentro de classes, e
- No controlo do acesso aos membros da classe.

O controlo do acesso estabelece o que o programador cliente pode usar separando-o das estruturas e mecanismos internos, isto é, separa a interface da implementação.