

TI-220 Java Orientado a Objetos

ANTONIO CARVALHO - TREINAMENTOS

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

Modificadores

Modificadores de Acesso

Modificadores de Acesso

Existem 4 modificadores de acesso

- *default* (package)
- public
- private
- protected

Modificador de Acesso (default)

- **default (package)**, sinaliza a classe ou o membro que o acesso só pode ser feito por classes que pertençam ao mesmo package.
- Pode ser usado em **classes, métodos e variáveis de instância**

Modificador de Acesso (default)

- Acesso padrão (package)
 - Não possui *keyword* para identificar este modificador, quando quiser usar ***package*** não coloque nada
 - Se uma classe A estiver declarada como package e se uma classe B que estiver em outro pacote tentar acessá-la, ocorrerá erro porque a classe B não está no mesmo pacote da classe A
 - Para que o acesso ocorra a classe B precisa ser criado no mesmo package que a classe A

Modificador de Acesso (default)



```
package a;  
class Animal {  
    public void locomover() {  
        System.out.println("locomovendo");  
    }  
}
```

Classe **Gato**
Pertence a outro
Package



```
package a;  
public class Cachorro extends Animal {  
    @Override  
    public void locomover() {  
        System.out.println("Andando");  
    }  
}
```

Compile success

```
package b;  
import a.Animal;  
public class Gato extends Animal {  
    @Override  
    public void locomover() {  
        System.out.println("Andando");  
    }  
}
```

a.Animal não é público no package a; não pode ser acessado por fora do package

Modificador de Acesso (default)

Para corrigir o problema no exemplo anterior

- É preciso colocar ambas as classes **Animal** e **Gato** no mesmo ***package***
- **Ou** trocar o modificador da classe **Animal** para **public**

Modificador de Acesso (default)

Por que usamos o package ?

- O uso do package assegura que a classe não pode ser manipulada por outras classes fora do pacote.
- Como exemplo uma classe chamada Criptografia que contém métodos e variáveis necessários a um sistema de segurança.
 - Neste caso o desenvolvedor do sistema não deseja que sua classe Criptografia seja acessada por sistemas externos, portanto ele declara-a como package, dessa forma esta classe so é visível as classes do sistema de segurança

Modificador de Acesso (public)

public, indica que todas as outras classes independente do pacote em que estejam, podem acessar esta classe ou este membro, assumindo é claro que a classe onde o membro se encontra esteja visível para a classe que vai acessá-lo.

- Pode ser usado em **classes, métodos e variáveis de instância**

Modificador de Acesso (public)

O modificador ***public*** torna a classe, o método ou a variável visível para todas as outras classes.

Por exemplo corrija o problema anterior modificando a classe **Animal** para ***public***

Dessa forma a classe **Animal** será visível para todas as outras classes

A solid blue horizontal bar spanning the width of the slide at the bottom.

Modificador de Acesso (private)

private, demarca o elemento como privado, ou seja apenas os métodos internos da classe podem acessar os elementos marcados como private.

- Somente pode ser usado em **métodos** e **variáveis de instância**

Modificador de Acesso (private)


Um membro private não é transmitido através da herança.

No caso abaixo é possível compreender melhor como se classifica esta situação

- Este caso não é uma sobrescrita porque a classe **Gerente** não recebeu o método **trabalhar** por herança da classe **Funcionario**
- Portanto o que está ocorrendo é apenas a declaração de um novo método chamado **trabalhar** na classe **Gerente**

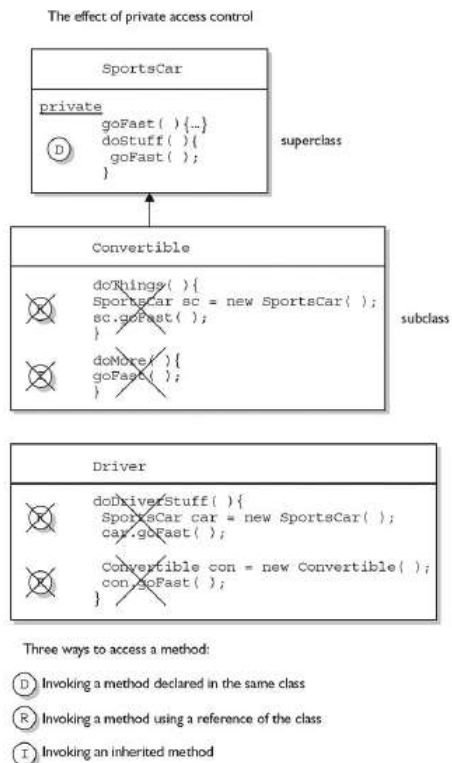
```
package ocjp.java.certification;  
public class Funcionario {  
    private void trabalhar() {  
        System.out.println("trabalhando");  
    }  
}
```

```
package ocjp.java.certification;  
public class Gerente extends Funcionario {  
    private void trabalhar() {  
        System.out.println("trabalhando");  
    }  
}
```

A blue arrow points from the Gerente class box to the Funcionario class box, indicating that Gerente inherits from Funcionario.

Observe a figura 1-3 no livro da Katy Sierra

Modificador de Acesso (private)



Fonte : SCJP Sun Certified Programmer
for Java 6 Study Guide (Exam 310065)


Modificador de Acesso (private)

Ao tentar usar um método ou variável que esta na superclasse ocorrerá erro de compilação.

- A classe **Gerente** não recebeu o método trabalhar por herança da classe **Funcionario**
- Portanto este método não pode ser chamado da classe **Gerente**

```
package a;  
public class Funcionario {  
    private void trabalhar() {  
        System.out.println("trabalhando");  
    }  
}
```

```
package a;  
public class Gerente extends Funcionario {  
    private void liderar() {  
        trabalhar();  
        System.out.println("liderando");  
    }  
}
```



O método trabalhar() originário do tipo Funcionario não é visível.

Modificador de Acesso (protected)

protected, sinaliza a que o membro somente pode ser acessado por classes que pertençam ao mesmo package ou através da herança mesmo que a classe pertença a outro package


- Pode ser usado em **métodos** e **variáveis de instância**

Modificador de Acesso (protected)

Veja o exemplo ao lado e responda as questões

- Por que o método **trabalhar()** pode ser acessado por herança ?
- Por que este mesmo método não pode ser acessado via objeto (**f**) do tipo **Funcionario** ?
- O que pode ser feito para que o método **trabalhar** seja executado das duas formas ? (não vale torná-lo público)

```
package a;  
public class Funcionario {  
    protected void trabalhar() {  
        System.out.println("trabalhando");  
    }  
}
```



```
package b;  
public class Gerente extends Funcionario {  
    private void liderar() {  
        trabalhar();  
        Funcionario f = new Funcionario();  
        f.trabalhar();  
        System.out.println("liderando");  
    }  
}
```

The method `trabalhar()` from the type `Funcionario` is not visible

Modificador de Acesso (protected)

Por que o método **trabalhar()** pode ser acessado por herança ?

- **R:** Porque ele é um método **protected** e está sendo acessado através de herança da classe **Funcionario**

Por que este mesmo método não pode ser acessado via objeto (**f**) do tipo **Funcionario** ?

- **R:** Pelo fato do método ser **protected** ele somente pode ser acessado por classes do mesmo pacote ou através da herança. O objeto (**f**) do tipo **Funcionario** está tentando acessar o método **trabalhar()**, mas está sendo invocado em uma classe que não pertence ao **package**.

O que pode ser feito para que o método **trabalhar** seja executado das duas formas ? (não vale torná-lo público)

- **R:** Para que o problema seja resolvido basta colocar ambas as classes no mesmo **package**

Modificador de Acesso (protected)

Exercício

1. Formem pequenos grupos e verifiquem a situação do próximo quadro.
2. Debatam por 5 minutos se há algum erro no próximo quadro ou se todos os códigos funcionarão sem problemas.
3. Verifiquem a solução na página seguinte

Modificador de Acesso (protected)

Veja o exemplo abaixo e responda as questões

```
package a;
public class Funcionario {
    protected void trabalhar() {
        System.out.println("trabalhando");
    }
}
```

1

```
package b;
public class Gerente extends Funcionario {
    private void liderar() {
        trabalhar();
        System.out.println("liderando");
    }
}
```

2

```
package b;
public class Estagiario {
    public void acompanharTrabalho() {
        Gerente o = new Gerente();
        o.trabalhar();
    }
}
```

3

```
package b;
public class Diretor extends Gerente {
    public void acompanharTrabalho() {
        trabalhar();
    }
}
```

4

- Será possível acessar o método trabalhar da classe Estagiario através de uma relação que não seja a herança ?
- Ocorrerá algum erro, em algum quadro ?

Modificador de Acesso (protected)

Será possível acessar o método trabalhar da classe **Gerente** através de uma relação que não seja a herança ?

- **R:** Não como o método é **protected** a única forma de acesso a ele é estando dentro do mesmo **package**, ou através da **herança**

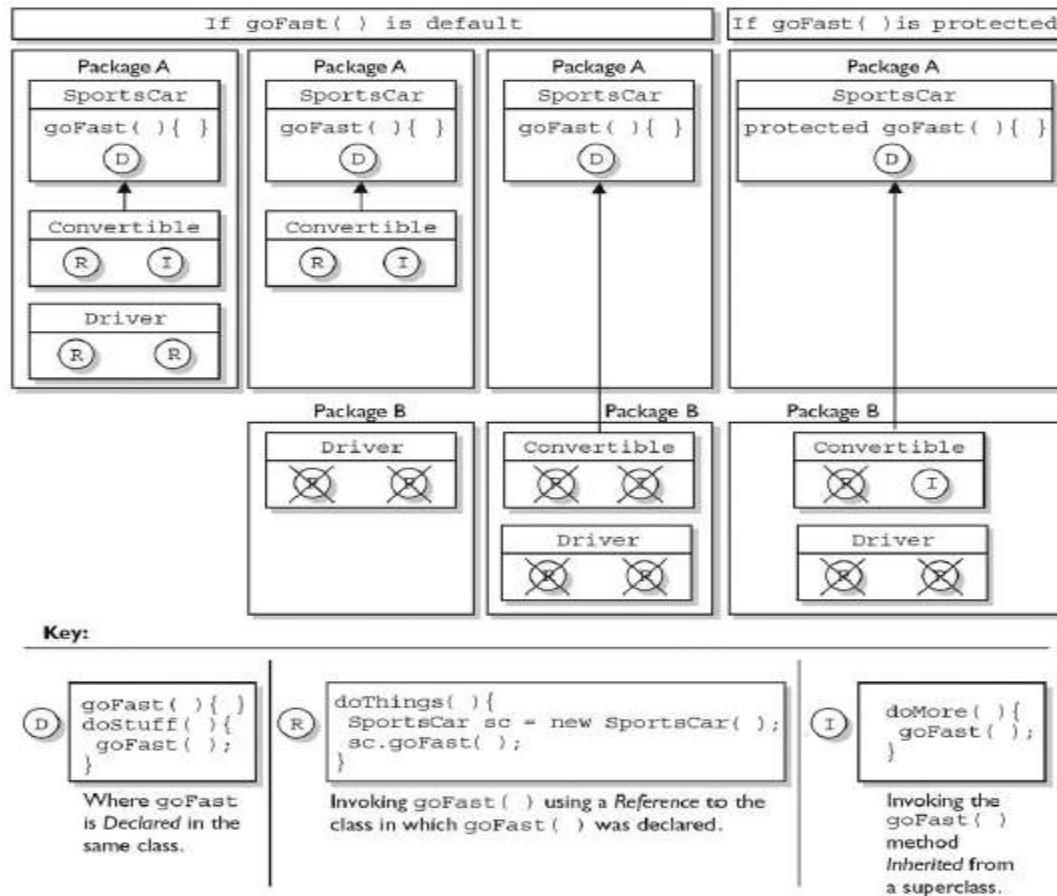
Ocorrerá algum erro em algum dos quadros ?

- **R:** Ocorrerá um erro no quadro 3, pois está tentando fazer um acesso através da **referência** e não através de **herança**

Observe a figura 1-4 no livro da Katy Sierra

Nota : Quando ocorre herança os membros são transmitidos com o mesmo modificador de acesso que foram criados.

Modificador de Acesso (protected)



Fonte : SCJP Sun Certified Programmer for Java 6 Study Guide (Exam 310065)

Modificador de Acesso

Visibilidade	Public	Protected	<i>Default</i>	Private
Da mesma classe	Sim	Sim	Sim	Sim
De uma subclasse do mesmo package	Sim	Sim	Sim	Não
De qualquer classe do mesmo package	Sim	Sim	Sim	Não
De uma subclasse de outro package	Sim	Sim	Não	Não
De qualquer classe de outro package (sem herança)	Sim	Não	Não	Não

Fonte : (SIERRA, 2008) – Adaptado pelo autor

Nota : Variáveis locais não podem receber modificadores de acesso