



Recuperação Arquitetural

Trabalho Prático 12

Christian Marlon Souza Couto
Luana Almeida Martins

Objetivo

- Detectar violações arquiteturais;
- Recomendar refatorações para a violação arquitetural.

Solução

- Fornecer regras de conformidade;
- Relatar violações conforme as regras estabelecidas;
- Recomendar refatorações de código;
- Realizar refatoração.

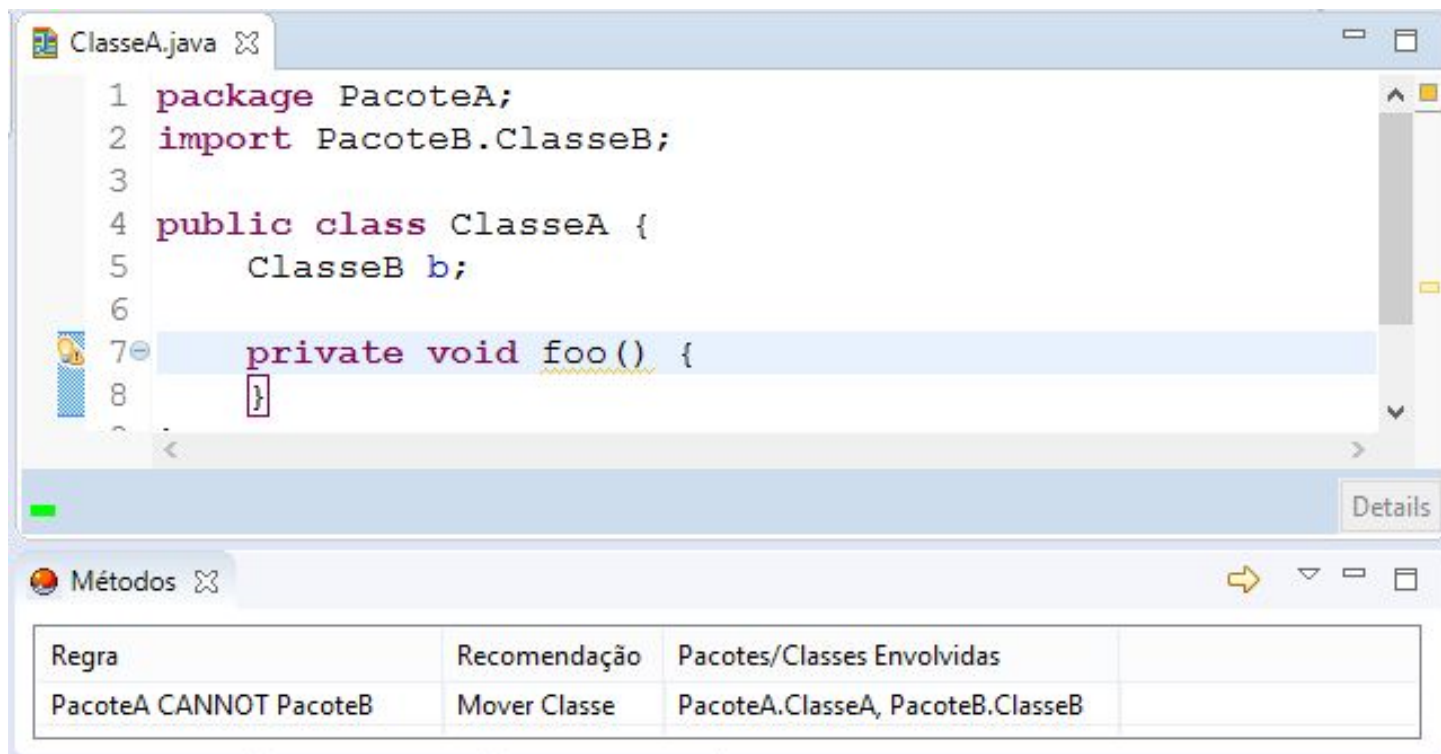
Terminologia

- CAN - indica uma dependência permitida;
- CANNOT - indica uma dependência não permitida;
- MUST - indica uma dependência obrigatória.

Tipos de recomendação

- CANNOT
 - Mover classe
 - Mover Método
- CAN e MUST
 - Verificação manual

Dependências de Atributos



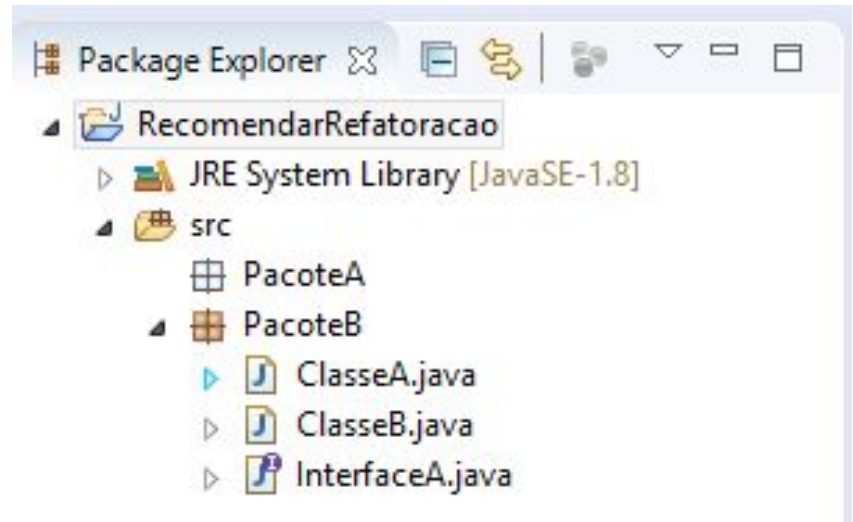
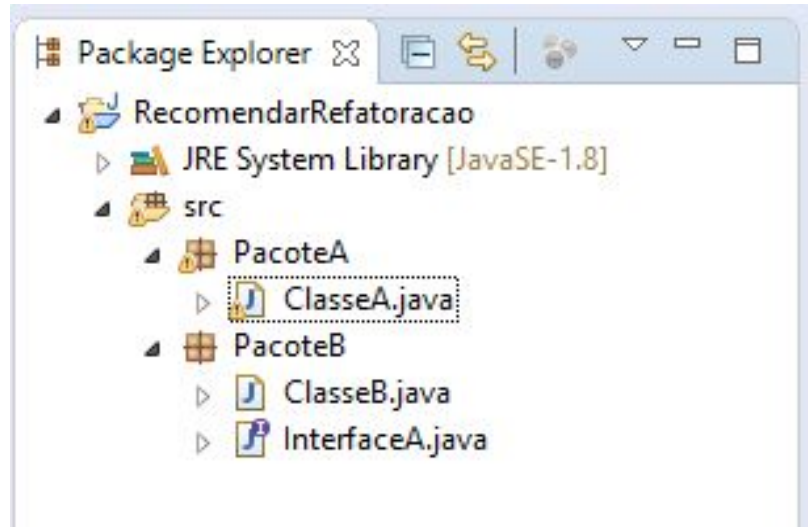
The screenshot shows an IDE window titled 'ClasseA.java' containing the following Java code:

```
1 package PacoteA;
2 import PacoteB.ClasseB;
3
4 public class ClasseA {
5     ClasseB b;
6
7     private void foo() {
8     }
```

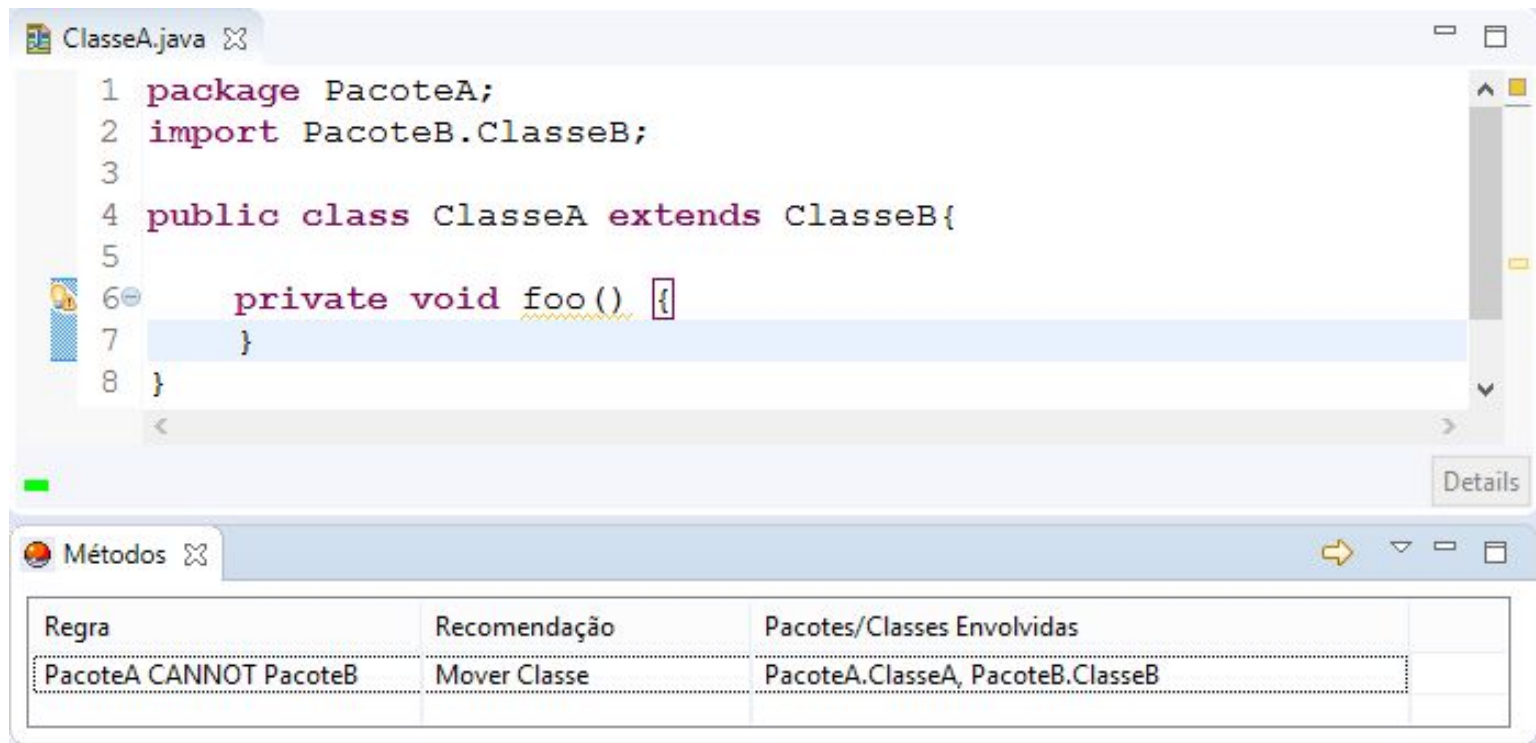
A red error icon is visible on the left margin next to line 7. Below the code editor, a 'Métodos' (Methods) tab is active, displaying a table with recommendations for resolving the error.

Regra	Recomendação	Pacotes/Classes Envolvidas
PacoteA CANNOT PacoteB	Mover Classe	PacoteA.ClasseA, PacoteB.ClasseB

Dependências de Atributos



Depêndencias de Extends/Implements



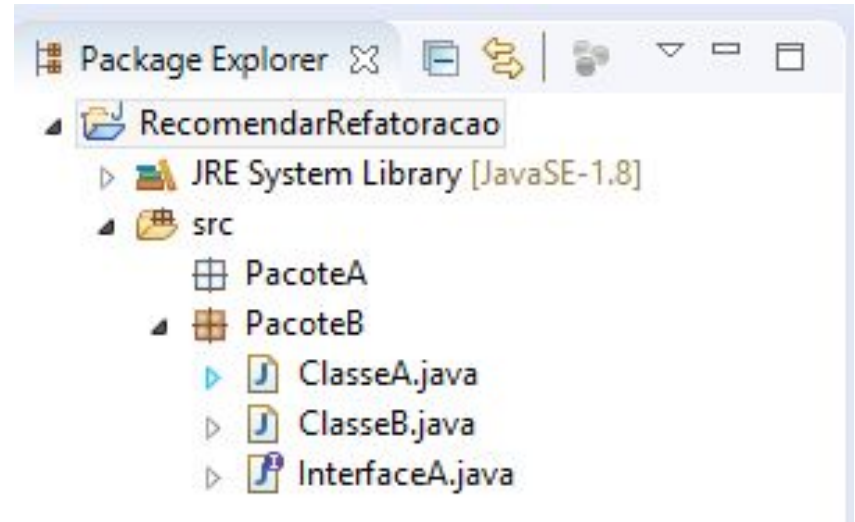
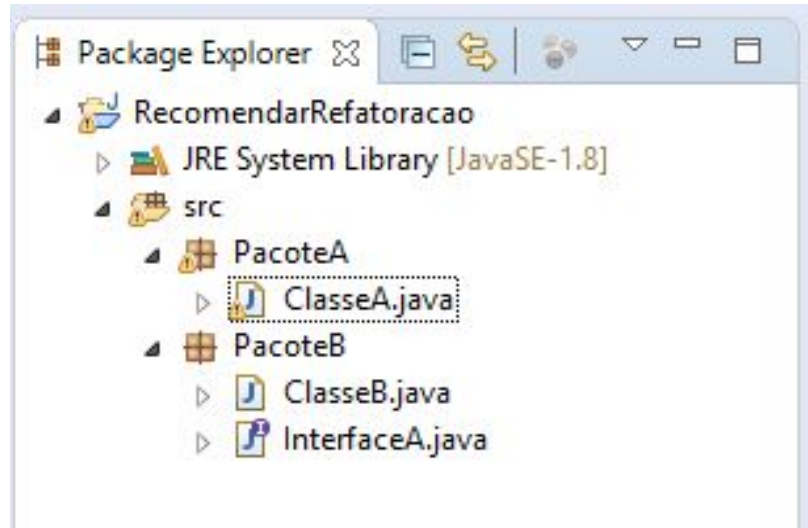
The screenshot displays an IDE window titled 'ClasseA.java' containing the following Java code:

```
1 package PacoteA;  
2 import PacoteB.ClasseB;  
3  
4 public class ClasseA extends ClasseB{  
5  
6     private void foo() {  
7     }  
8 }
```

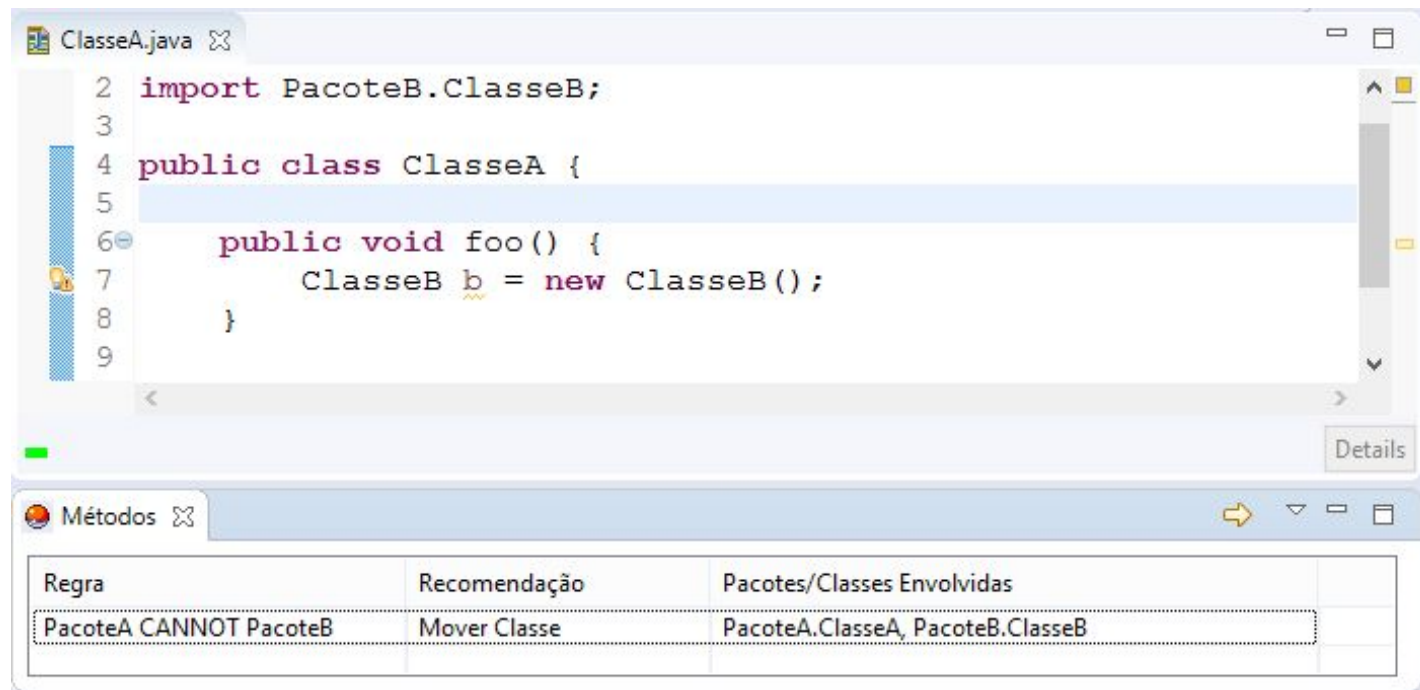
Below the code editor, the 'Métodos' (Methods) tab is active, showing a table of dependency warnings:

Regra	Recomendação	Pacotes/Classes Envolvidas
PacoteA CANNOT PacoteB	Mover Classe	PacoteA.ClasseA, PacoteB.ClasseB

Depêndencias de Extends/Implements



Dependências de instâncias



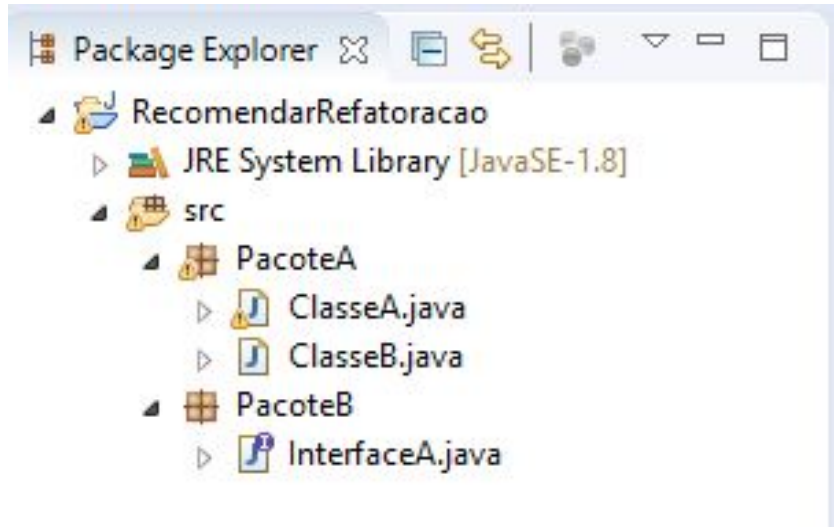
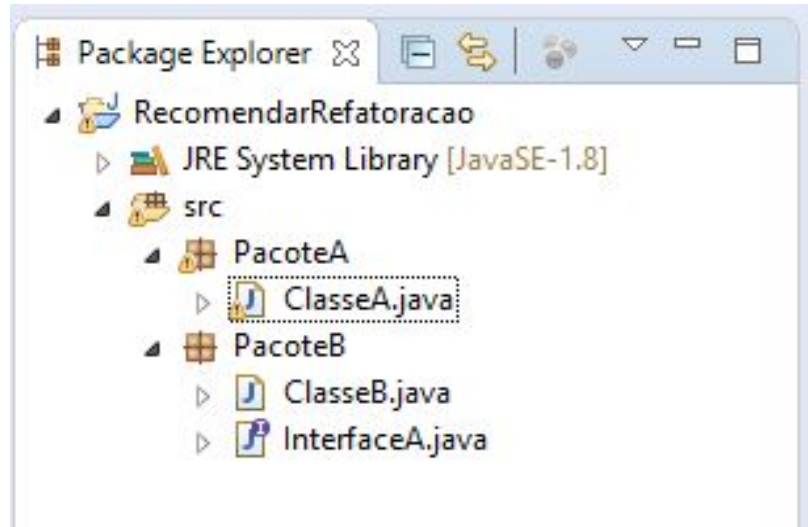
The screenshot shows an IDE window titled "ClasseA.java" containing the following Java code:

```
2 import PacoteB.ClasseB;
3
4 public class ClasseA {
5
6     public void foo() {
7         ClasseB b = new ClasseB();
8     }
9 }
```

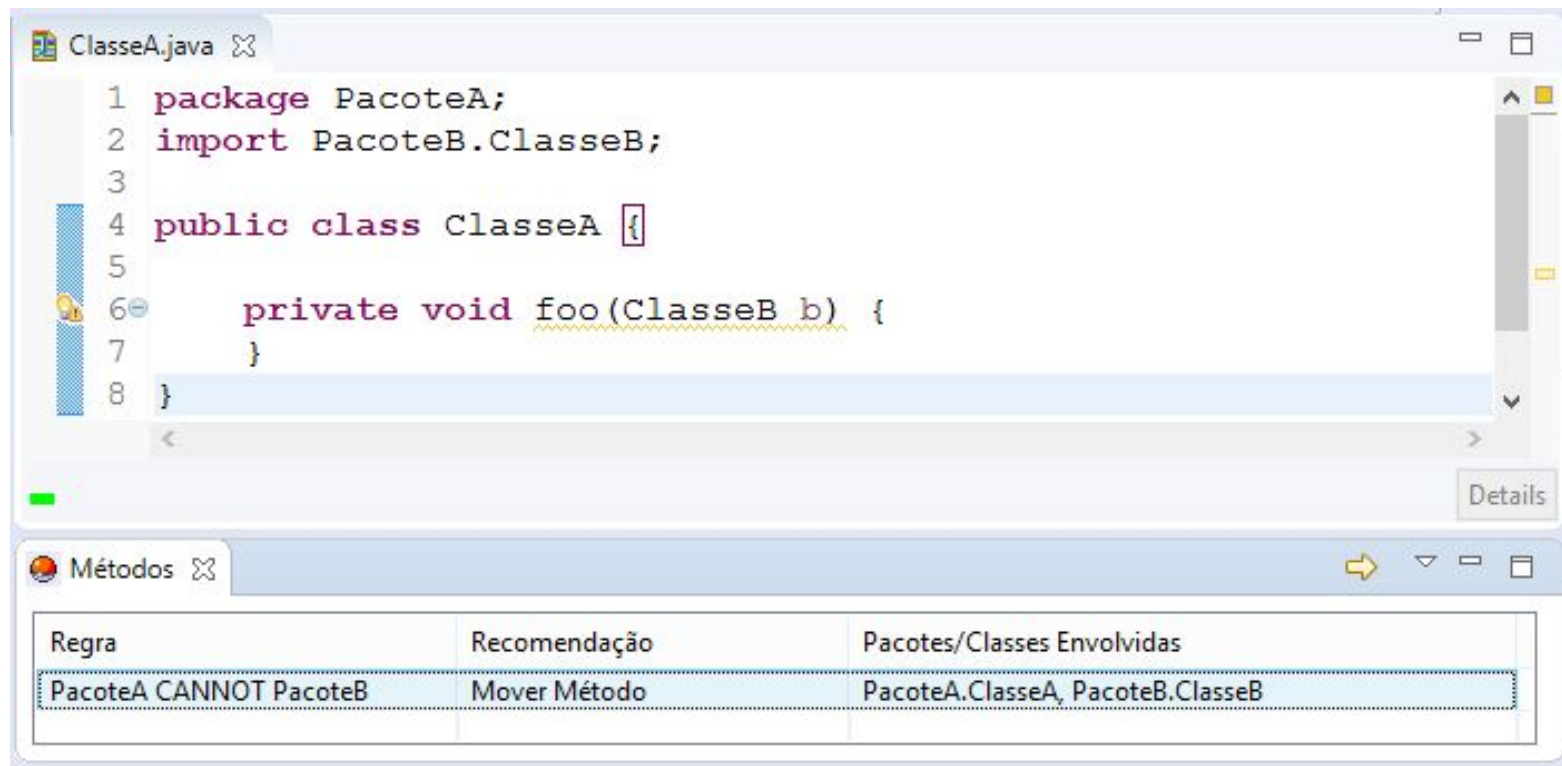
Below the code editor, a "Métodos" (Methods) tab is active, displaying a table with recommendations for resolving a dependency issue.

Regra	Recomendação	Pacotes/Classes Envolvidas
PacoteA CANNOT PacoteB	Mover Classe	PacoteA.ClasseA, PacoteB.ClasseB

Dependências de instâncias



Dependências de Parâmetros



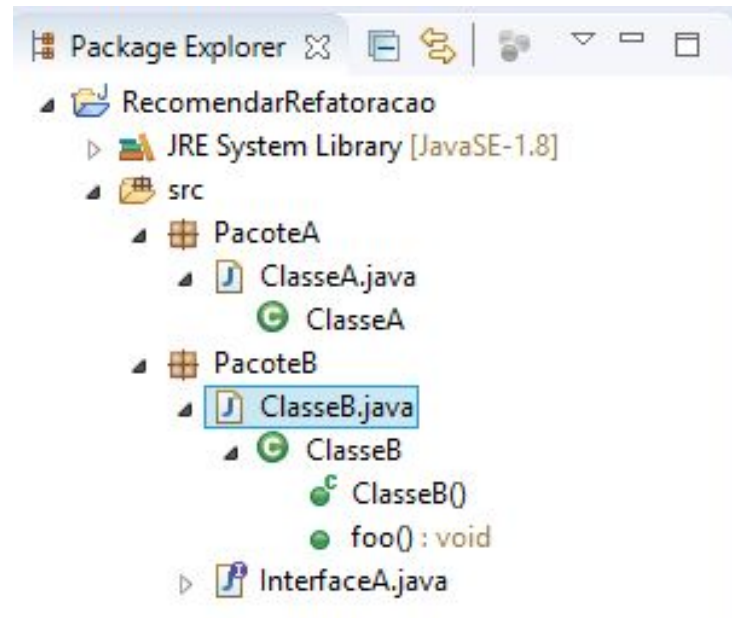
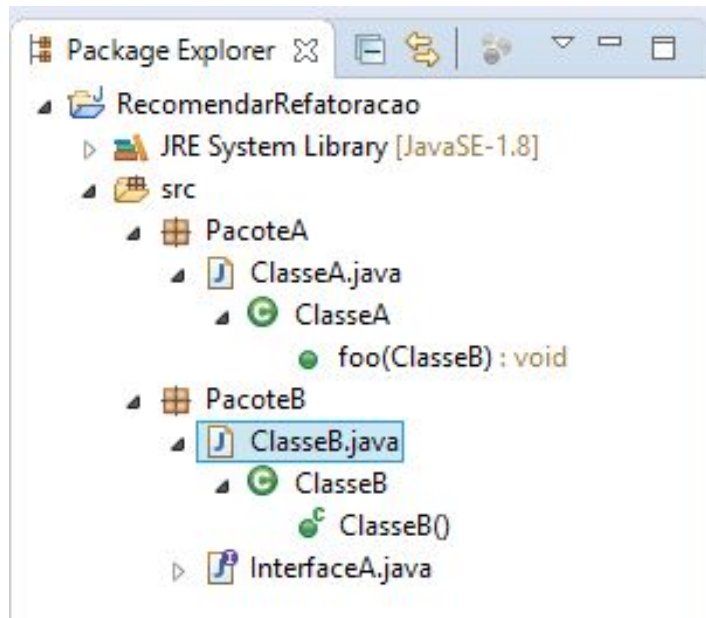
The screenshot shows an IDE window titled 'ClasseA.java' containing the following Java code:

```
1 package PacoteA;  
2 import PacoteB.ClasseB;  
3  
4 public class ClasseA {  
5  
6     private void foo(ClasseB b) {  
7     }  
8 }
```

A yellow warning icon is visible next to line 6. Below the code editor is a panel titled 'Métodos' (Methods) which displays a table of recommendations:

Regra	Recomendação	Pacotes/Classes Envolvidas
PacoteA CANNOT PacoteB	Mover Método	PacoteA.ClasseA, PacoteB.ClasseB

Dependências de Parâmetros



Implementação

