Consider the following inheritance structure and the methods blow, note the specifiers on the inheritance arrows.

The method *foo* is defined in A. (The method itself is public. All methods in Squeak are public).

**Private Inheritance:**
B is the only class that knows that B inherits from A. This means that all methods of A are considered private in B. Consider the following methods: (The green comment shows what will happen when the method is executed)

```plaintext
Methods in B:
bar1
  self foo //ok
bar2
  B new foo //ok
bar3
  E new foo //error

Methods in E:
bar1
  self foo //error
bar2
  B new foo //error
bar3
  E new foo //error

Methods in G:
bar1
  B new foo //error
Bar2
  E new foo //error
```

Running the following line in Squeak:

```squeak
B classifyInheritedMethod: #foo.
```

Will return ‘private’
Protected Inheritance:
C and classes that derive from C are the only classes that know that C inherits from A. This means that all methods of A are considered protected in C. Consider the following methods:

Methods in C:
- fun1
  self foo //ok
- fun2
  C new foo //ok
- Fun3
  F new foo //error

Methods in F:
- fun1
  self foo //ok
- fun2
  C new foo //ok
- Fun3
  F new foo //ok

Methods in G:
- fun1
  C new foo //error
- fun2
  F new foo //error

Running the following line in Squeak:
C classifyInheritedMethod: #foo.
Will return ‘protected’

Protected Inheritance:
All classes in Squeak know that D inherits from A. This means that all methods of A are considered public in D. Consider the following methods:

Methods in D:
- f1
  self foo //ok
- f2
  D new foo //ok

Methods in B:
- f1
  D new foo //ok

Methods in G:
- f1
  D new foo //ok

Running the following line in Squeak:
D classifyInheritedMethod: #foo.
Will return ‘public’

Of course, running the following line in Squeak:
G classifyInheritedMethod: #foo.
Will return ‘undefined’