Concepts of Object-Oriented Programming

Prof. Dr. Peter Müller

Chair of Programming Methodology

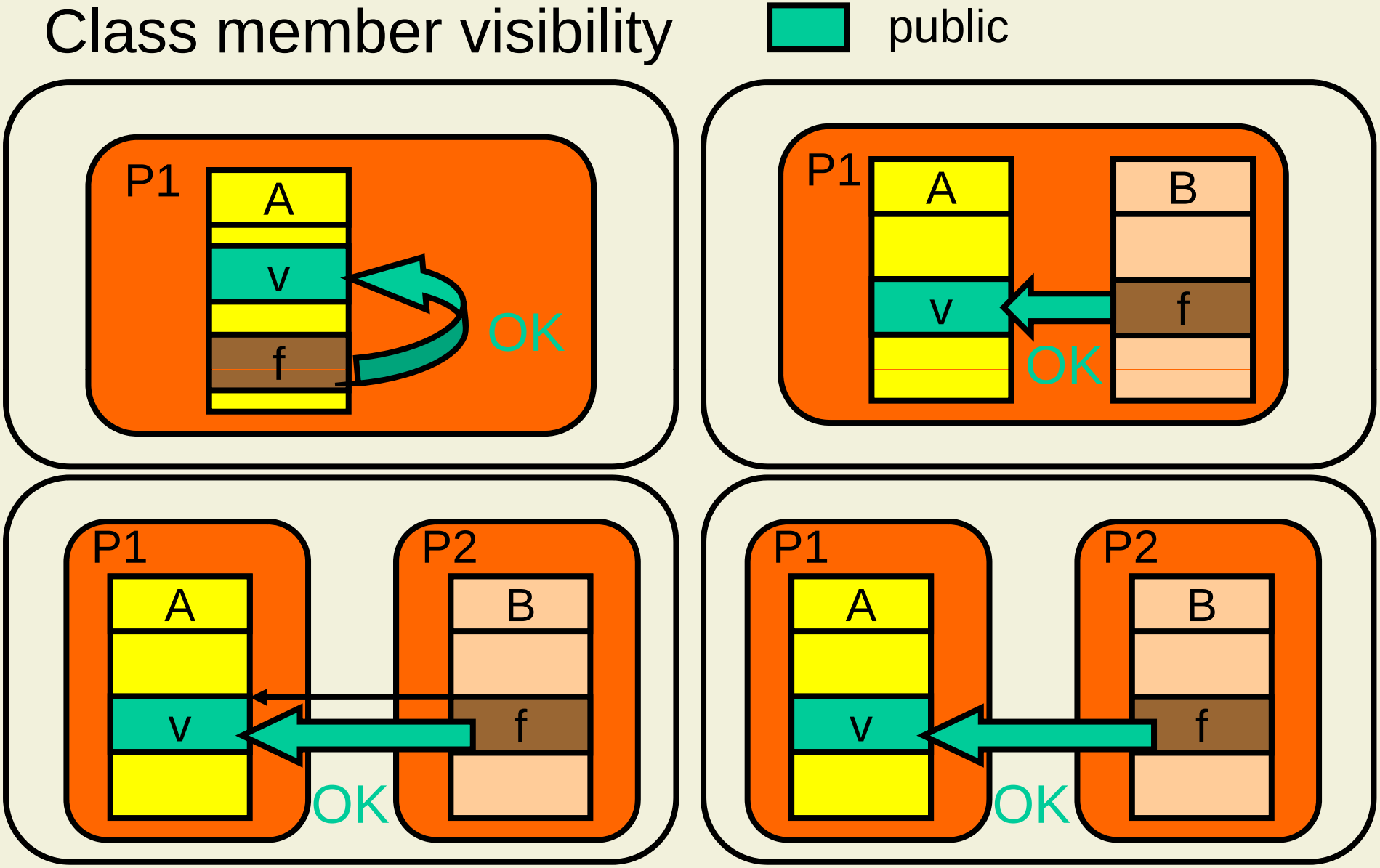
Exercises 6: Information Hiding

Information Hiding

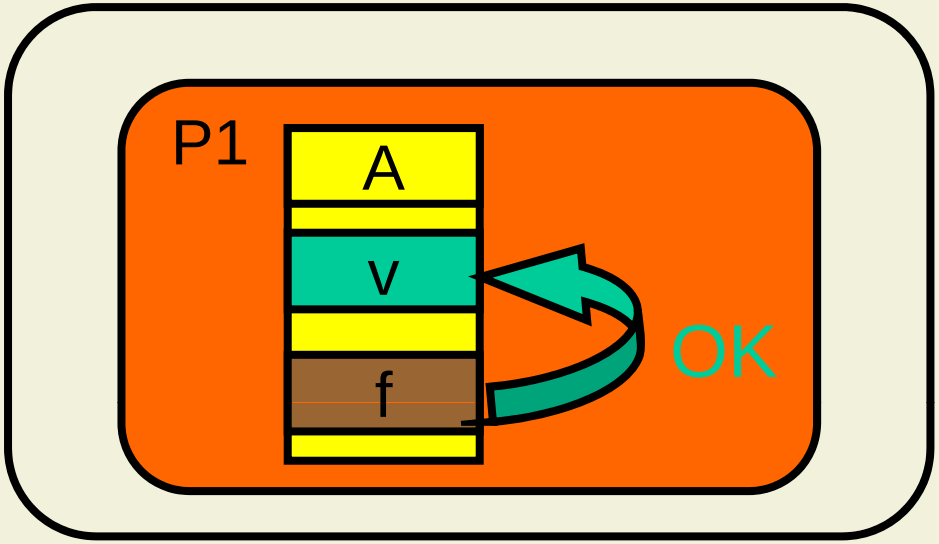
- Definition
  Information hiding is a technique for reducing the dependencies between modules:
  - *The intended client is provided with all the information needed to use the module **correctly**, and **with nothing more***
  - *The client uses only the (publicly) available information*

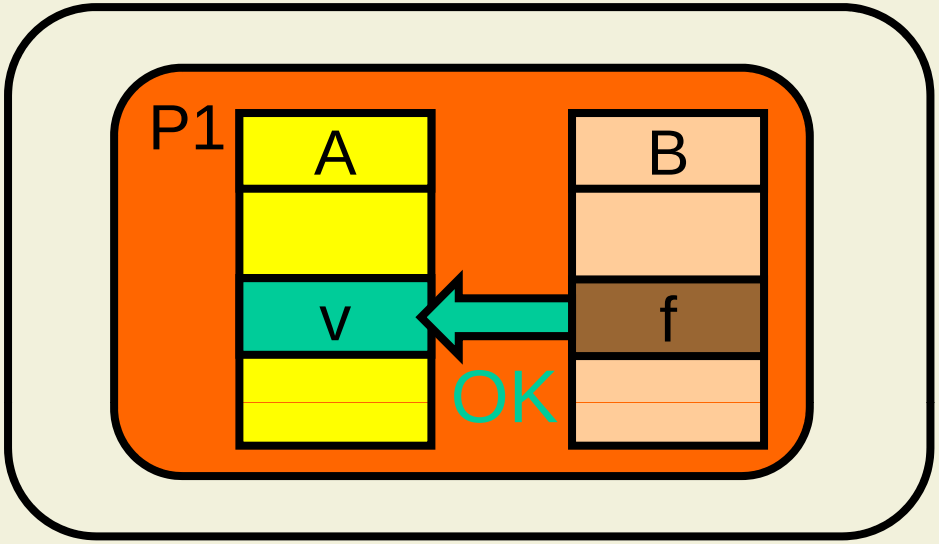
- Information hiding deals with programs, that is, with static aspects
- Contracts are part of the exported interfaces

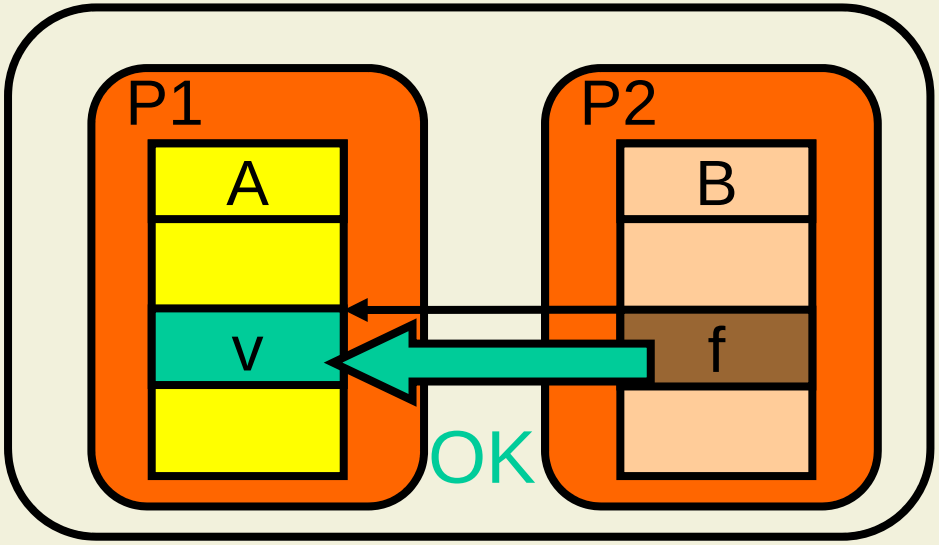
Class member visibility

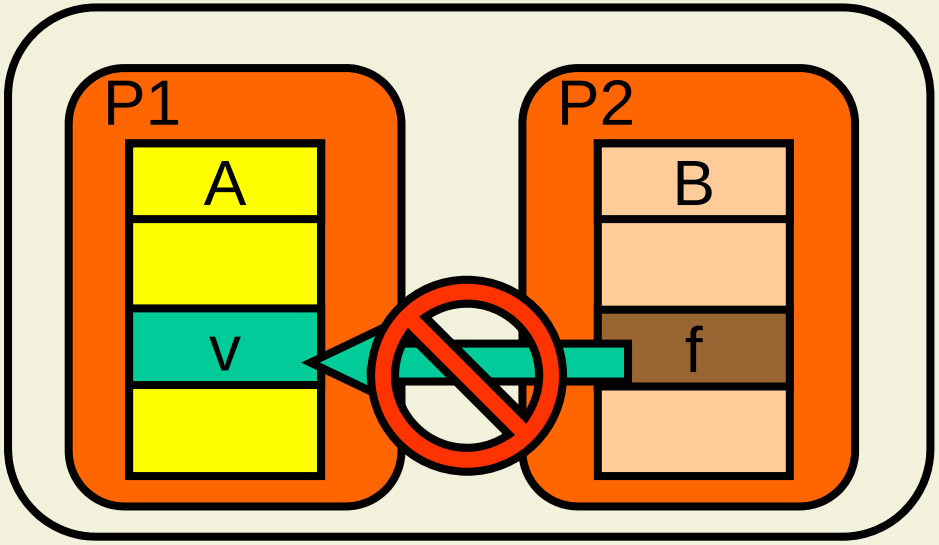


Class member visibility

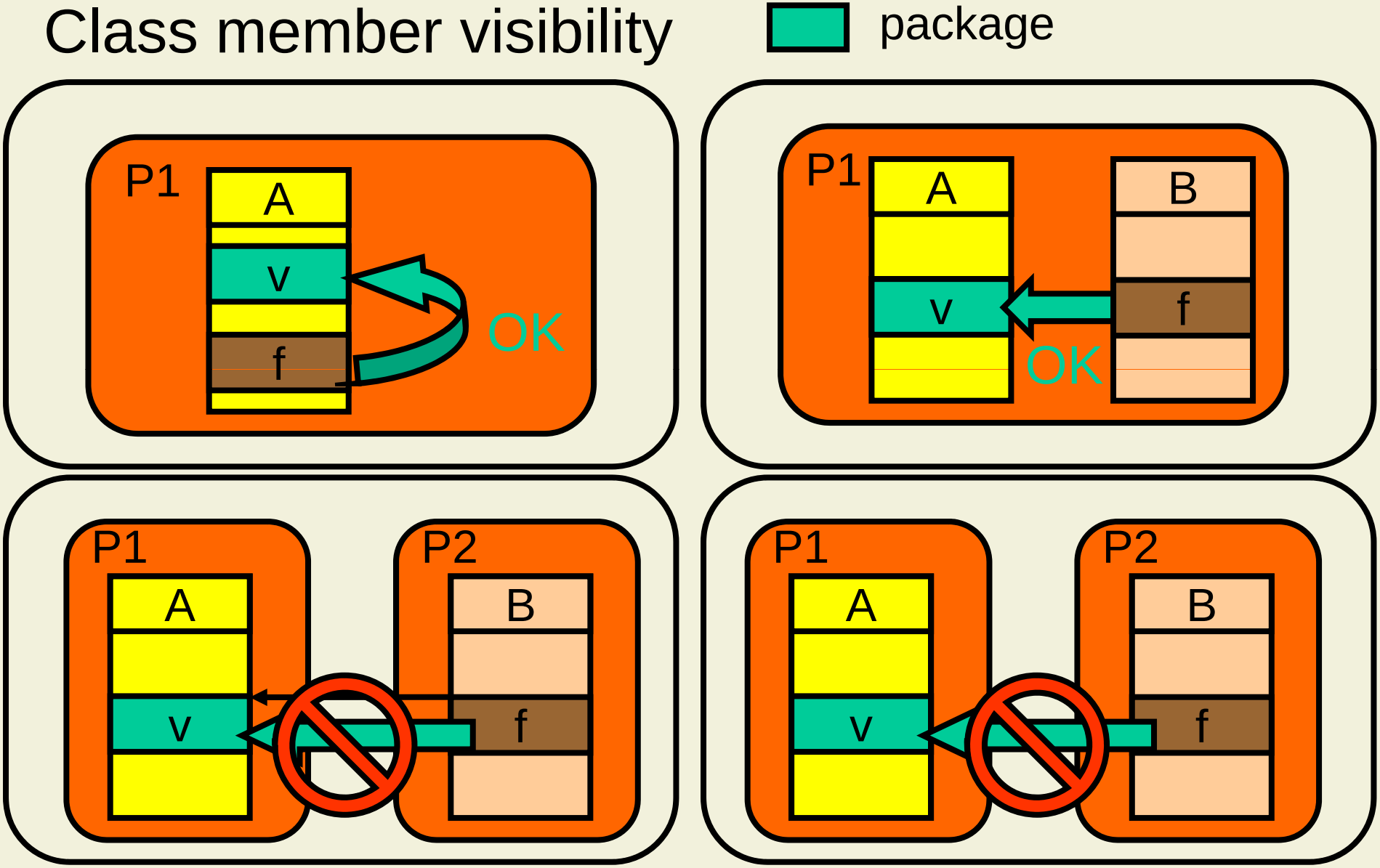




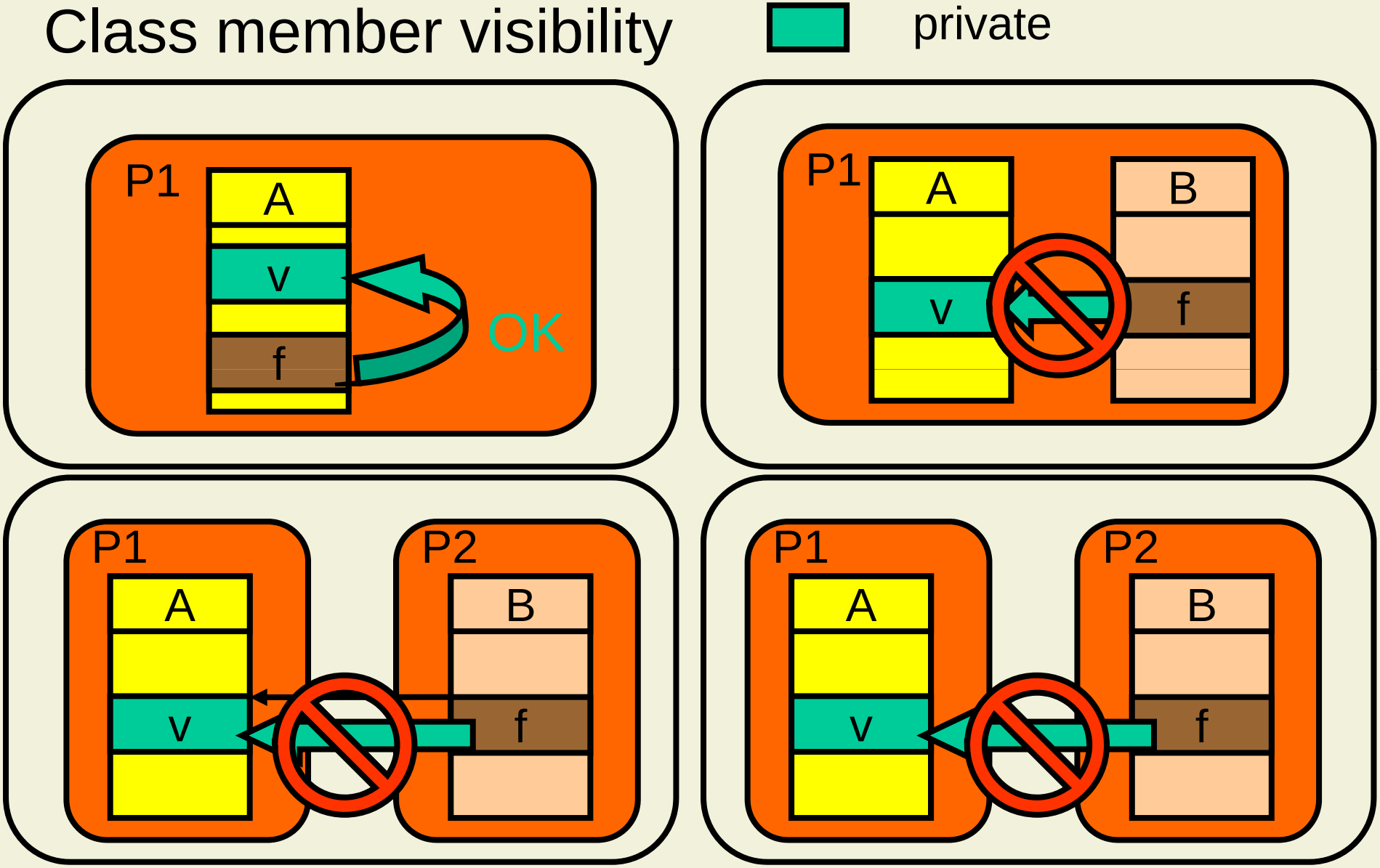




Class member visibility



Class member visibility



Example

```
package p1;
public class A{
public int v;
    void f() {
        int i = v;
    }
}
```

OK

```
package p1;
class B{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

```
package p2;
class B extends A{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

```
package p2;
class B {
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

Example

```
package p1;
public class A{
protected int v;
    void f() {
        int i = v;
    }
}
```

OK

```
package p1;
class B{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

```
package p2;
class B extends A{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

```
package p2;
class B {
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

Example

```
package p1;
public class A{
package int v;
    void f() {
        int i = v;
    }
}
```

OK

```
package p1;
class B{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

OK

```
package p2;
class B extends A{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

```
package p2;
class B {
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

Example

```
package p1;
public class A{
private int v;
    void f() {
        int i = v;
    }
}
```

OK

```
package p1;
class B{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

```
package p2;
class B extends A{
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

```
package p2;
class B {
    void f() {
        A a = new A();
        int b = a.v;
    }
}
```

Error

Class member visibility

Accessible for	public	protected	package	private
the same class	Yes	Yes	Yes	Yes
a class in the same package	Yes	Yes	Yes	No
a subclass in an other package	Yes	Yes	No	No
a non subclass in an other package	Yes	No	No	No

Exercise 1 a

```
public class AList<T> {
  private T[] tarray;
  private int nextidx;

  public AList() {
     // the cast to T[] is unchecked!
     tarray = (T[]) new Object[10];
     nextidx = 0;    }

  public AList( T[] d ) {
     tarray = d;
     nextidx = d.length;    }
```

Exercise 1 b

- getFirstNode has protected visibility
- All classes in this package can call the method
- All subclasses can call the method

Exercise 1 c

The second constructor

```
public AList( T[] d ) {
 tarray = d;
 nextidx = d.length;    }
```

captures the array that is given as parameter.

The modification of the `ia` array is observable!

The output of the main program changes!

Exercise 2 a

```
public class A {
    private int a;
    private int b;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public A(int ta) {
   a = ta;
   b = 0;
    }
    public void increment() {
   ++a;
   ++b;
    }
}
```

OK!

Exercise 2 b

```
public class B {
    public int a;
    public int b;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public B(int ta) {
   a = ta;
   b = 0;
    }
    public void increment() {
   ++a;
   ++b;
    }
}
```

Not OK!

Exercise 2 c

```
public class C {
    private int a;
    private int b;
    public int c;
    /*@ invariant a >= b; @*/
    /*@ requires ta >= 0; @*/
    public C(int ta) {
   a = ta;
   b = 0;
    }
    public void increment() {
   ++a;
   ++b;
    }
}
```

OK!

Exercise 3

a. Method getIDs leaks reference to attribute ids

b.

```
package Attacker;
import System.*;

public class Attack {
  public static void attack(Environment u) {
    Authorization auth = new Authorization();
    u.insertAuthorization(auth);
    int[] data = auth.getIDs();
    data[0] = 666;
  }
}
```

c.

```
public int[] getIDs() {return (int[]) ids.clone()}
```

Exercise 3

d. Method setIDs is **protected**

e.

```
package Attacker;
import System.*;

public class Attack extends Authorization {
  public static void attack(Environment u) {
    Attack att = new Attack();
    u.insertAuthorization(att);
    int[] data = new int[5];
    att.setIDs(data);
    data[0] = 666;
  } }
```

f.

```
protected void setIDs(int[] p) { ids = p; }
```

```
public final class Authorization { ... }
```

Questions?