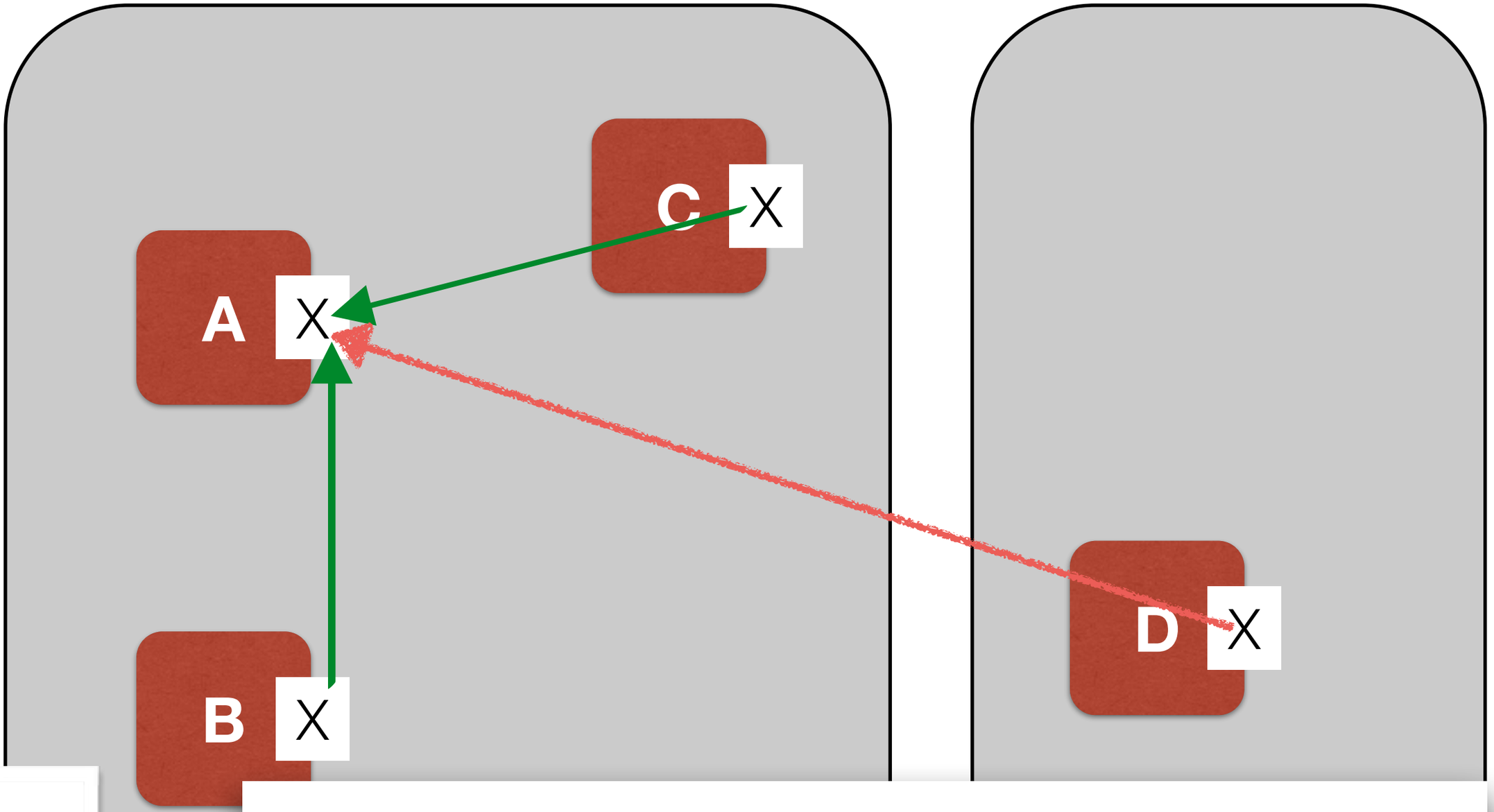
Inkapsling, Ekvivalens och Identitet

Inkapsling?

”Package Scope”



Legend

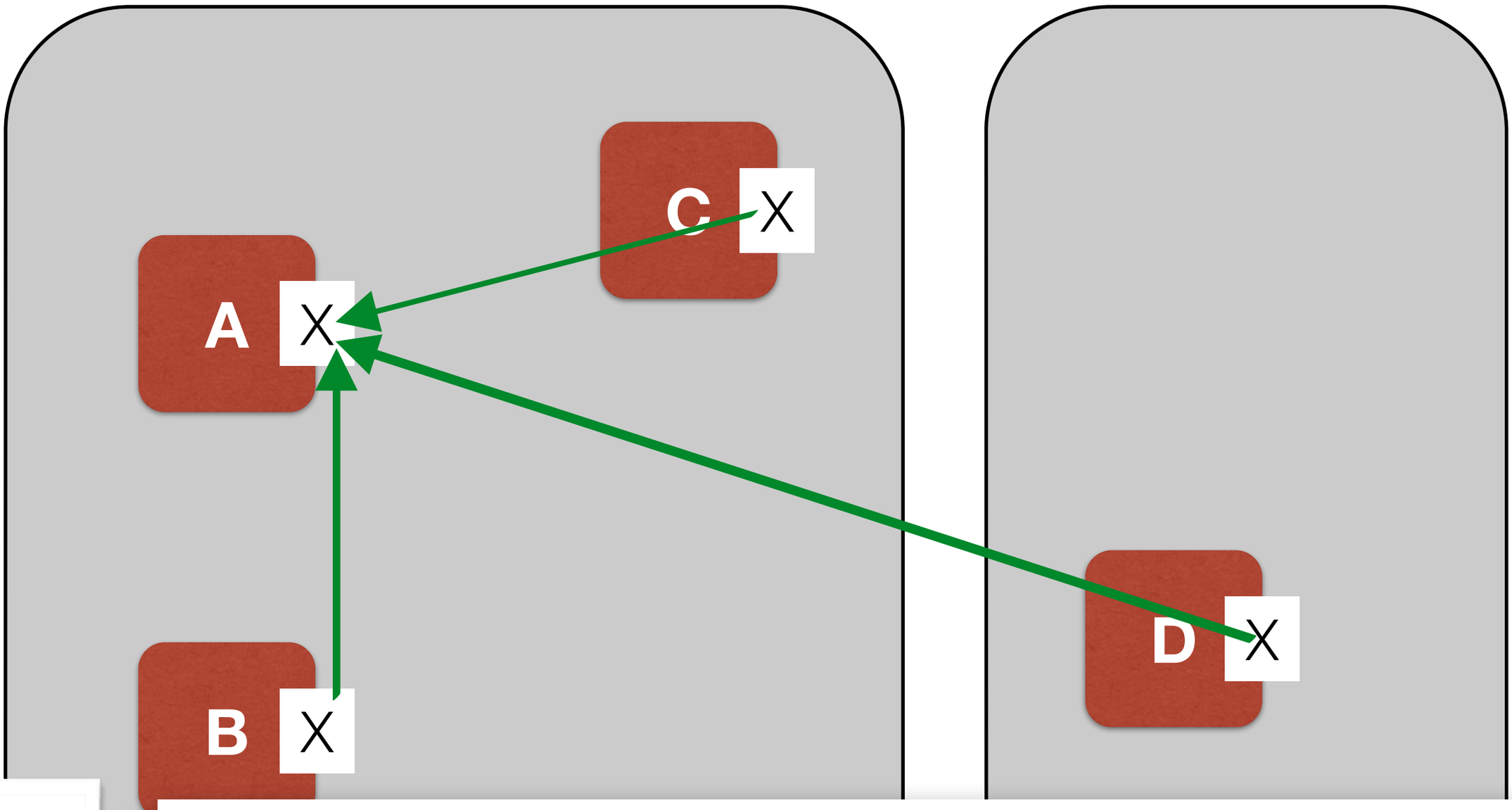
B ärver av A

C och D kan ärva av A — det påverkar ej!

```
class Student extends Person {
    String[] courses;

    void addCourse(final String s) { ... }
```

Åtkomstmodifieraren public



Legend

B ärver av A

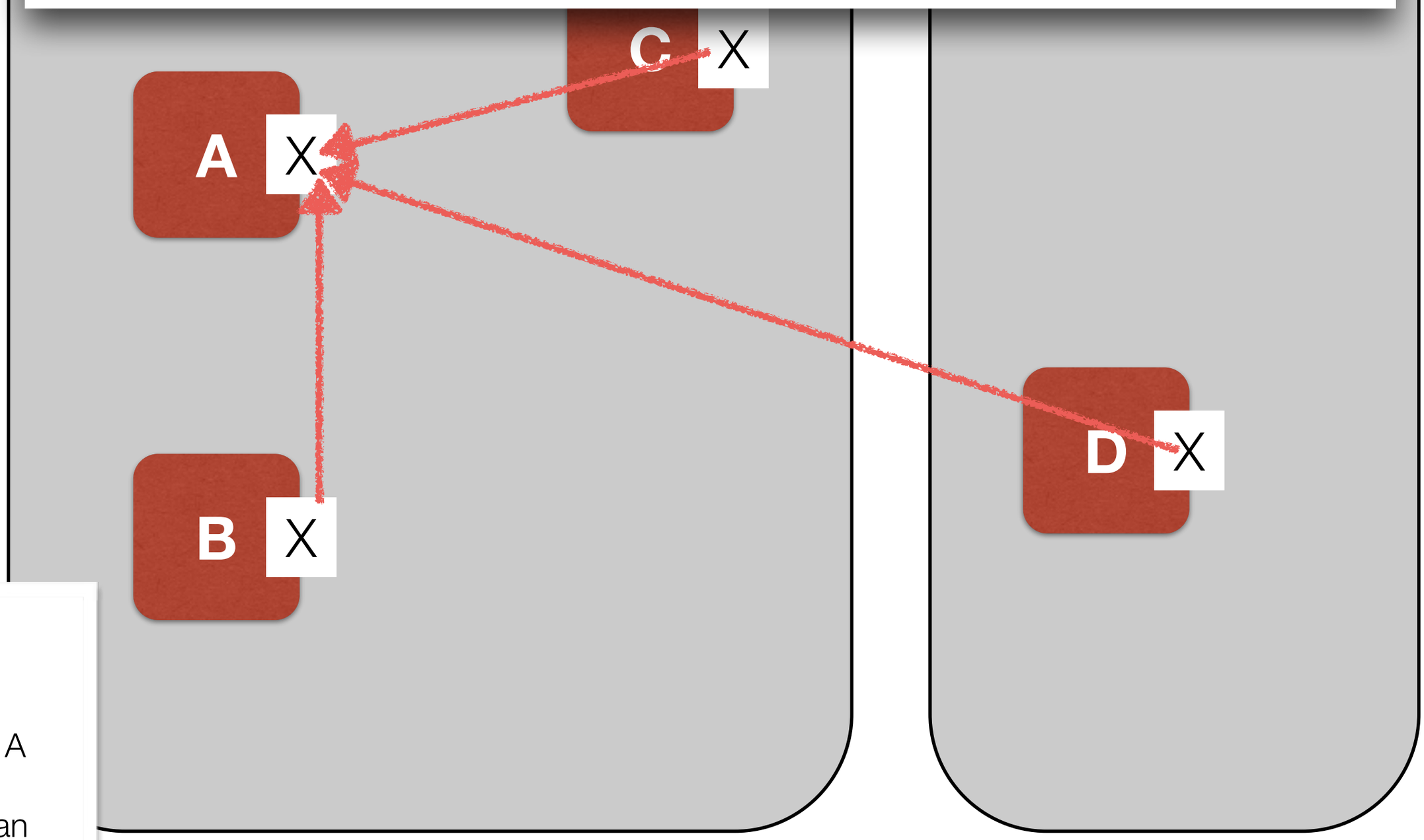
C och D kan ärva av A — det påverkar ej!

```
class Student extends Person {
    public String[] courses;

    public void addCourse(final String s) { ... }
```

```java
class Student extends Person {
    private String[] courses;

    private void addCourse(final String s) { ... }
```



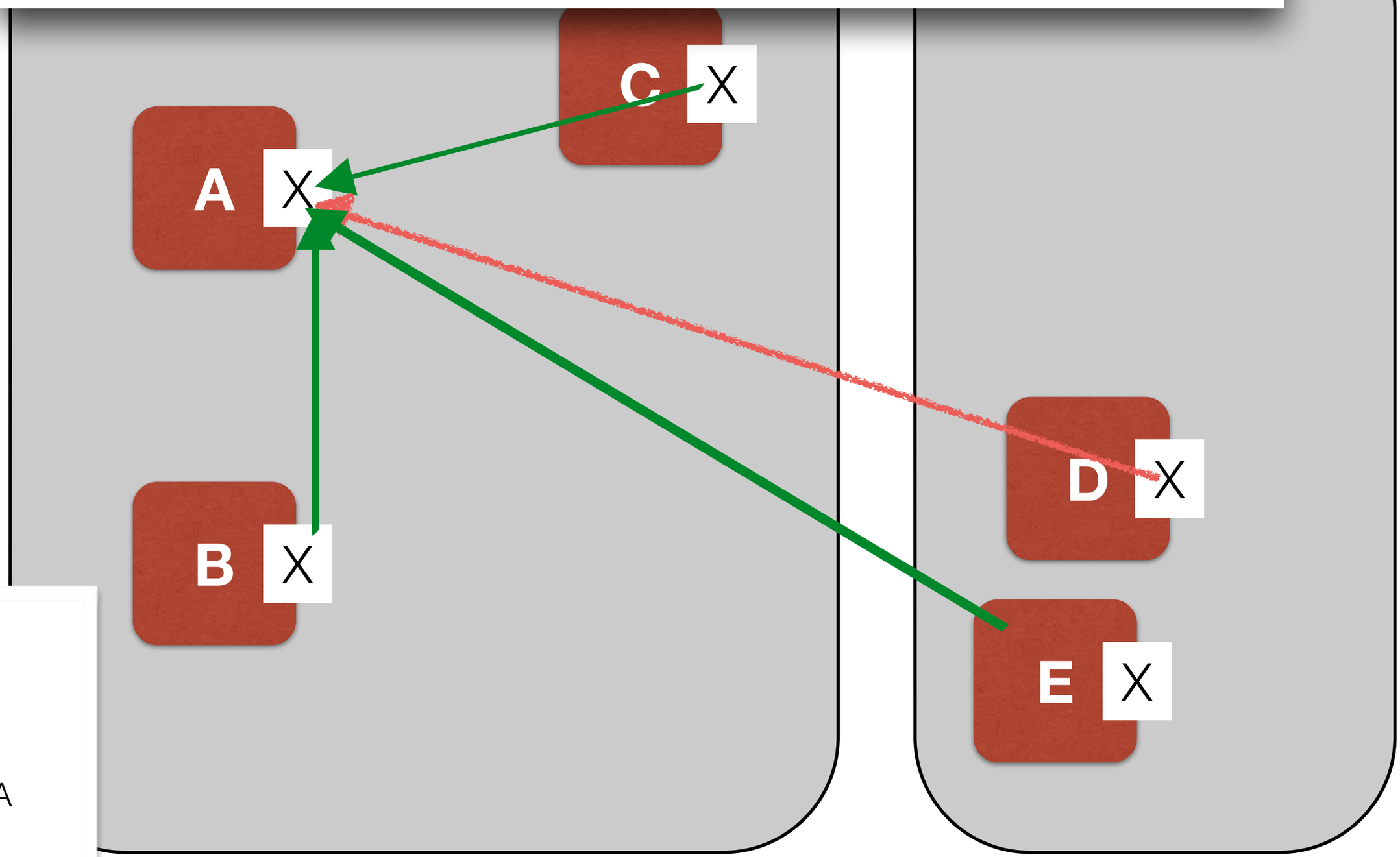
Legend

B ärver av A

C och D kan ärva av A — det påverkar ej!

```java
class Student extends Person {
    protected String[] courses;

    protected void addCourse(final String s) { ... }
```



Legend

B och E
ärver av A

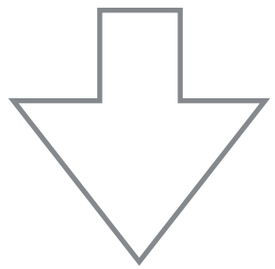
C och D
ärver **EJ** av A

Inkapsling?

```
class Student extends Person {
    private String[] courses;

    public void addCourse(final String s) { ... }

    public String[] getCourses() { return courses; }
```



```
class Student extends Person {
    private String[] courses;

    public void addCourse(final String s) { ... }

    public String[] getCourses() { return courses.clone(); }
```

När är två objekt lika varandra?

UU

Vad betyder ”lika”?

UU

Två definitioner:

Ekvivalent

Måste definieras specifikt för varje typ!

Identiska

Exakt samma objekt

Hur definierar vi likhet för...

- Klassen Personnummer?

  Har datum och fyra sista siffror

- Klassen Person?

  Har förnamn, efternamn, personnummer

- Klassen Student

  Som är en person, men som också har en lista med kurser som hen går

- Klassen String?

- Klassen Integer?

- En länkad lista?

- En graf?

```
class SSN {
    final Calendar birthDate = new java.util.GregorianCalendar();
    final int secretCode[] = new int[4];

    public SSN(final int y, final int m, final int d, final Integer... secret) {
       // KOD FÖR ATT KOLLA ATT PERSONNUMRET ÄR VALITT UTELÄMNAD

       birthDate.set(y, m, d);
       for(int i = 0; i < 4; ++i) secretCode[i] = secret[i];
    }

    public boolean equals(final Object o) {
       if (o instanceof SSN) {
           SSN other = ((SSN) o);
           return birthDate.equals(other.birthDate) &&
                  secretCode.equals(other.secretCode);
        } else {
            return false;
       }
    }

    public String toString() {
        return "SSN(" + birthDate.get(Calendar.YEAR) + ", " +
                        birthDate.get(Calendar.MONTH) + ", " +
                        birthDate.get(Calendar.DATE) + ", " + ... + ")";
    }
}
```

Vad gör .equals här?

```
class Person {
    String firstName;
    String lastName;
    SSN ssn;

    public Person(final String first, final String last, final SSN ssn) {
        this.firstName = first;
        this.lastName  = last;
        this.ssn       = ssn;
    }

    public boolean equals(final Object o) {
        if (o instanceof Person) {
            Person other = ((Person) o);
            return firstName.equals(other.firstName) &&
                   lastName.equals(other.lastName) &&
                   ssn.equals(other.ssn);
        } else {
            return false;
        }
    }

    public String toString() {
        return "Person(" + firstName + ", " + lastName + ", " + ssn + ")";
    }
}
```

```
package foo;

class Student extends Person {
    String[] courses;

    public Student(final String first, final String last, final SSN ssn,
                   final String... courses) {
        super(first, last, ssn);
        this.courses = courses;
    }

    public boolean equals(final Object o) {
        if (super.equals(o) && o instanceof Student) {
            return java.util.Arrays.equals(courses, ((Student)
o).courses);
        } else {
            return false;
        }
    }
}
```

```
private static class Link {
    private Link next;
    private Link prev;
    private Object value;

    public static Link newSentinel() {
        return new Link() { // Anonymous class trick
            public Link find(Object value) { return null; }
        };
    }

    private Link() {
        this.next = this;
        this.prev = this;
    }

    public Link(final Link prev, final Link next, final Object value) {
        this.next  = next;
        this.prev  = prev;
        this.value = value;

        prev.next = this;
        next.prev = this;
    }
```

```
    public Link getNext() { return next; }
    public Link getPrev() { return prev; }

    private void setNext(final Link l) { next = l; }
    private void setPrev(final Link l) { prev = l; }

    public Link find(final Object value) {
        return (this.value.equals(value)) ? this : this.next.find(value);
    }

    public void unlinkNext() {
        this.next.next.prev = this; this.next = this.next.next;
    }

    public void unlink() {
        this.prev.next = this.next; this.next.prev = this.prev;
    }

    public Link insertBefore(final Object v) {
        return new Link(this.prev, this, v);
    }

    public Link insertAfter(final Object v) {
        return new Link(this, this.next, v);
    }
}
```

```
public class List {
    private final Link sentinel = Link.newSentinel();

    public List() {}

    public boolean contains(final Object value) {
        return this.first().find(value) != null;
    }

    public Object get(final int index) {
        Link c = this.first();

        for (int i = 0; i <= index && c != this.last(); c = c.next, ++i) {
            if (index == 0) return c.value;
        }

        return null;
    }

    public int size() {
        int size = 0;

        for (Link c = this.first(); c != this.last(); c = c.next) ++size;

        return size;
    }
```

<Här klipper vi in Link-klassen>

```
public boolean remove(final Object value) {
    final Link l = this.first().find(value);

    if (l == null) {
        return false;
    } else {
        l.unlink();
        return true;
    }
}
```

```
public boolean insertAfter(final Object beforeValue,
                           final Object value) {
    final Link l = this.first().find(beforeValue);

    if (l == null) {
        return false;
    } else {
        l.insertAfter(value);
        return true;
    }
}
```

```
public boolean insertBefore(final Object afterValue,
                            final Object value) {
    final Link l = this.first().find(afterValue);

    if (l == null) {
        return false;
    } else {
        l.insertBefore(value);
        return true;
    }
}
```

UU

```
public void prepend(final Object value) {
    this.sentinel.insertAfter(value);
}

public void append(final Object value) {
    this.sentinel.insertBefore(value);
}

private Link last() {
    return this.sentinel;
}

private Link first() {
    return this.sentinel.next;
}

public String toString() {
    StringBuilder sb = new StringBuilder();

    sb.append("[");
    for (Link c = this.first(); c != this.last(); c = c.getNext()) {
        sb.append(c.value.toString())
        if (c.next != this.last()) sb.append(", ");
    }
   sb.append("]");

    return sb.toString();
}
```

Vad blir utskriften?

```
public static void main(String[] args) {
    List l = new List();
    l.append(new Person("Tom", "Waits", new SSN(1949, 11, 07, 1, 2, 3, 4)));
    Person p = new Person("Tom", "Waits", new SSN(1949, 11, 07, 1, 2, 3, 4));
    System.out.println(l);
    l.remove(p);
    System.out.println(l);
}
```

Varför?

```
~/ioopm
$ javac List.java; and java List
[Person(Tom, Waits, SSN(1949, 11, 7, 1234))]
[Person(Tom, Waits, SSN(1949, 11, 7, 1234))]
```

Nästlade klasser

- Link är en del av klassen List
- Klassens namn blir List.Link
- Privat konstruktor: kan inte skapa Link:ar utanför List

Inre klasser

```
public class List {
    private final Link sentinel = Link.newSentinel();

    public List() {}

    private static class Link {
        private Link next;
```

UU

Slutar kompilera! Varför?

```
public class List { ...


    private class Link {
        private Link next;
        private Link prev;
        private Object value;

        public Link newSentinel() {
            return new Link() { // Anonymous class trick
                public Link find(Object value) { return null; }
            };
        }

        private Link() {
            this.next = this;
            this.prev = this;
        }

        public Link(final Link prev, final Link next, final Object value) {
```

Ekvivalens: två olika objekt som är likadana på alla relevanta ställen.

Dvs. alla relevanta instansvariabler är ekvivalenta.

Identitet: det är inte två olika objekt — det är ett och samma objekt.

Identitet implicerar ekvivalens, men inte tvärtom!