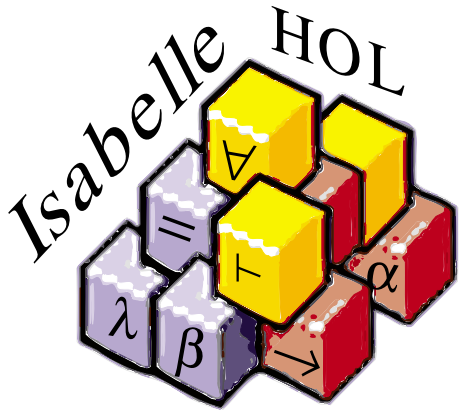
Modeling the Java Package/Access Concept in Isabelle



Norbert Schirmer
Software & Systems Engineering
TUM

Contents

- ***Motivation and Background***
- ***Model***
  - Packages and Access Modifiers
  - Members, Inheritance, Overriding, Accessibility
  - Different static and dynamic flavor
- ***Runtime Access Properties***
- ***Conclusions***

Motivation

Why studying Java Packages and Access Modifiers (private, protected, public)?

Language Design

- Information hiding
- Fine grained control over inheritance and overriding
- Competing modularization principles: modules vs. classes
- Package boundaries vs. inheritance

Java Security

- Runtime security based on bare language features
- Protection of private data like PINs

Clarification

- Java Language Specification (JLS) ambiguous and incomplete
- Inconsistency between language specification and JVM

Java Source Language Model

Main Theories

Language Features

- ⊕ Classes, Interfaces, Arrays
- ⊕ Classinitialization
- ⊕ Abrupt Completion (exceptions, break, continue, return)
- ⊕ Package/Access-Concept

- ⊖ Dynamic Class Loading
- ⊖ Threads
- ⊖ Garbage Collection & Finalization

Foundations

Basic Definitions

tname **Name of a type (interface or class)**
pname **Name of a package**

Packages (Qualified Typename)

record $qtname = pid::pname\ tid::tname$

Access Modifier

datatype $acc\text{-}modi = Private \mid Package \mid Protected \mid Public$
with $Private < Package < Protected < Public$

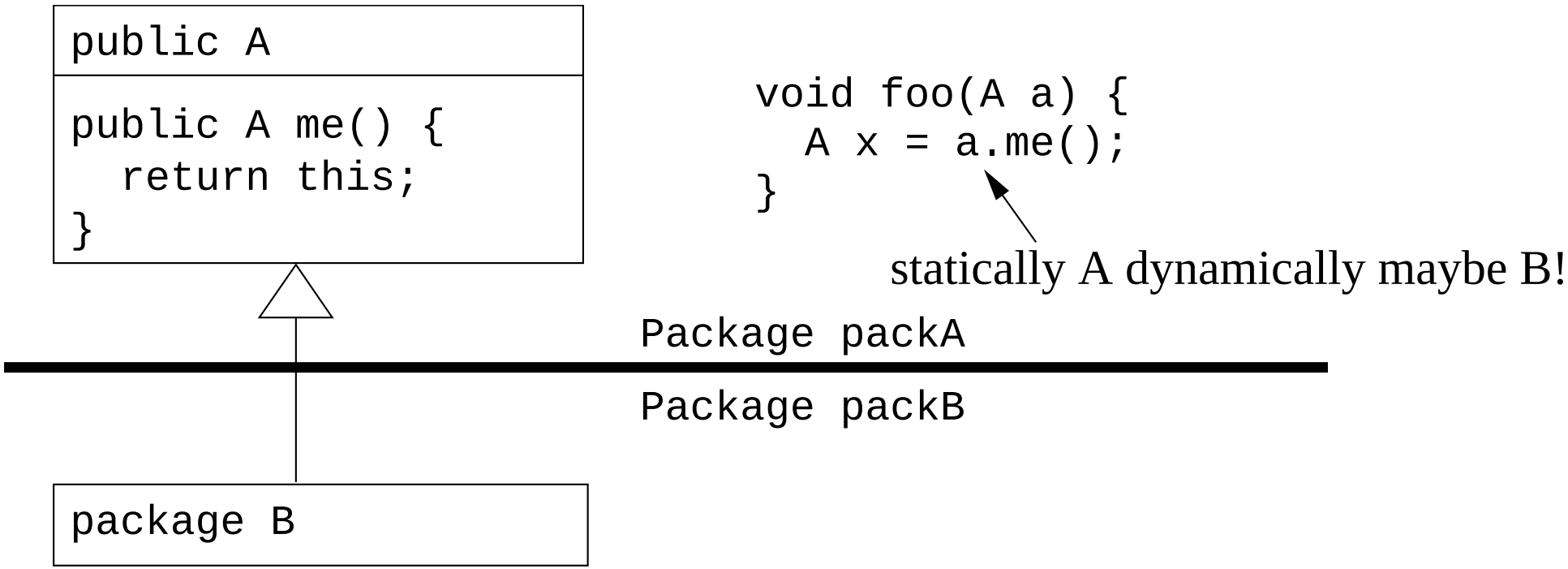
Attached to:

- types (classes, interfaces)
- members (fields, methods)

Accessibility of Types

type	$G \vdash$ *type accessible-in pack*
PrimT P	*True*
Iface I	*pid I* $=$ *pack* $\vee$ *is-public G I*
Class C	*pid C* $=$ *pack* $\vee$ *is-public G C*
Array T	$G \vdash$ *T accessible-in pack*

Accessibility of types is only *statically* guaranteed by the type system:

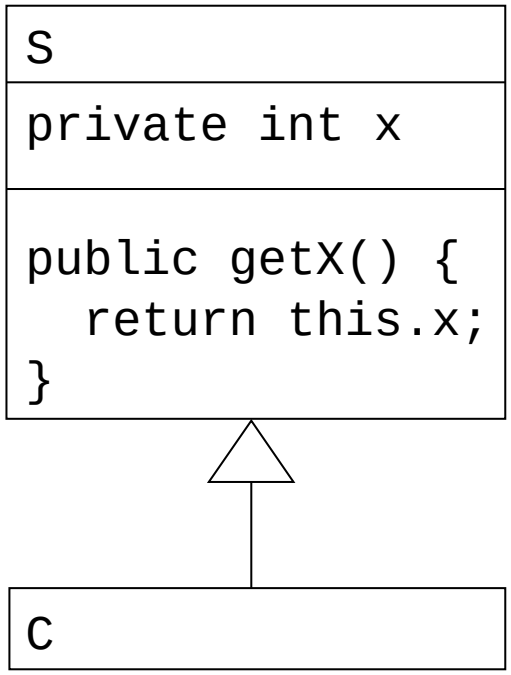


Members of a Class

$$\frac{G \vdash \mathit{mbr}\ m\ \mathit{declared\text{-}in}\ C \quad \mathit{declclass}\ m = C}{G \vdash m\ \mathit{member\text{-}of}\ C}\ (\textsc{Immediate})$$

$$\frac{G \vdash m\ \mathit{member\text{-}of}\ S \qquad G \vdash C \prec_{C1} S \quad G \vdash \mathit{Class}\ S\ \mathit{accessible\text{-}in}\ \mathit{pid}\ C \quad G \vdash \mathit{memberid}\ m\ \mathit{undeclared\text{-}in}\ C \quad G \vdash m\ \mathit{inheritable\text{-}in}\ \mathit{pid}\ C}{G \vdash m\ \mathit{member\text{-}of}\ C}\ (\textsc{Inherited})$$

$\mathit{accmodi}\ m$	$G \vdash m\ \mathit{inheritable\text{-}in}\ \mathit{pack}$
Private	*False*
Package	$\mathit{pid}\ (\mathit{declclass}\ m) = \mathit{pack}$
Protected	*True*
Public	*True*



Examples:

- $G \vdash$ `x` *member-of* `S`
- $\neg\ G \vdash$ `x` *member-of* `C`
- $G \vdash$ `getX` *member-of* `C`
- $G \vdash$ `x` *member-in* `C`

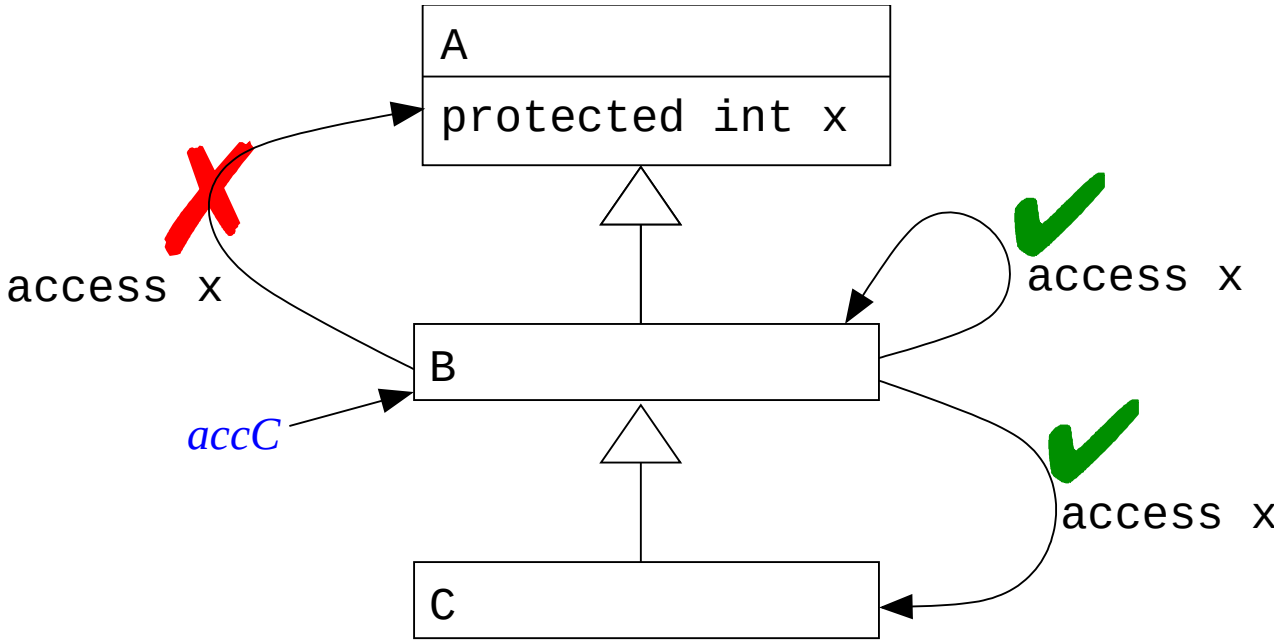
Runtime:

$$G \vdash m\ \mathit{member\text{-}in}\ C \equiv \exists\, \mathit{provC}.\ G \vdash C \preceq_C \mathit{provC} \wedge G \vdash m\ \mathit{member\text{-}of}\ \mathit{provC}$$

Member permits access

accmodi m	$G \vdash m$ *in* C *permits-acc-to* $accC$
Private	*declclass* $m = accC$
Package	*pid* (*declclass* m) = *pid* $accC$
Protected	*pid* (*declclass* m) = *pid* $accC \vee$ $G \vdash accC \prec_C$ *declclass* $m \wedge G \vdash C \preceq_C accC$ *"Same package and in subclasses"*
Public	*True*

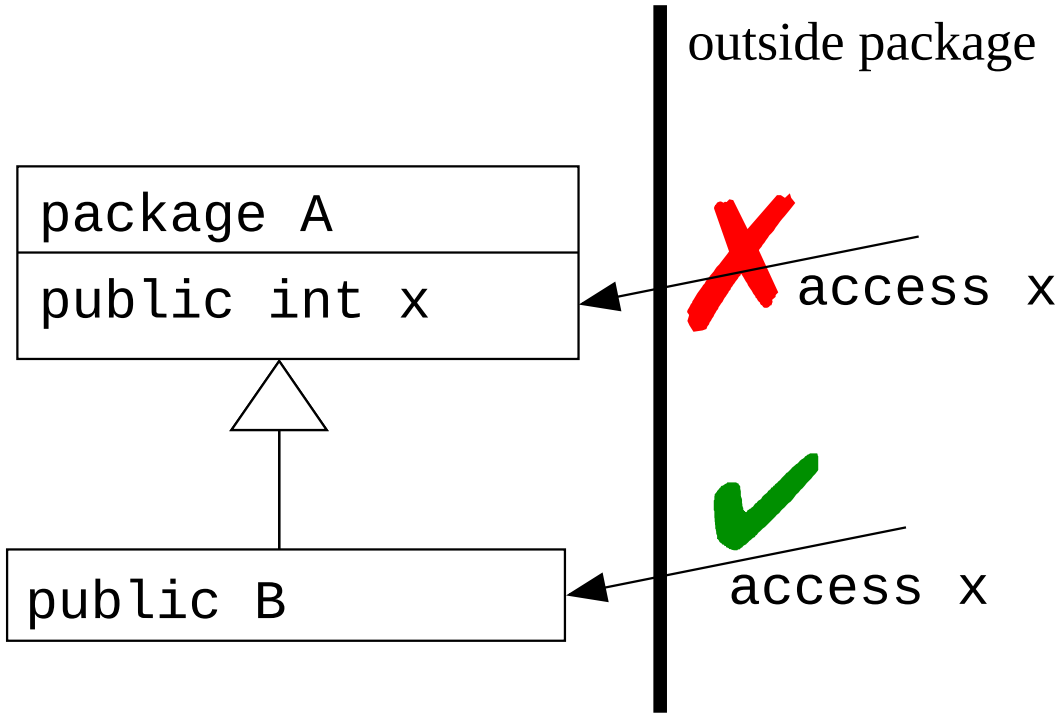
Example:



Accessible Member

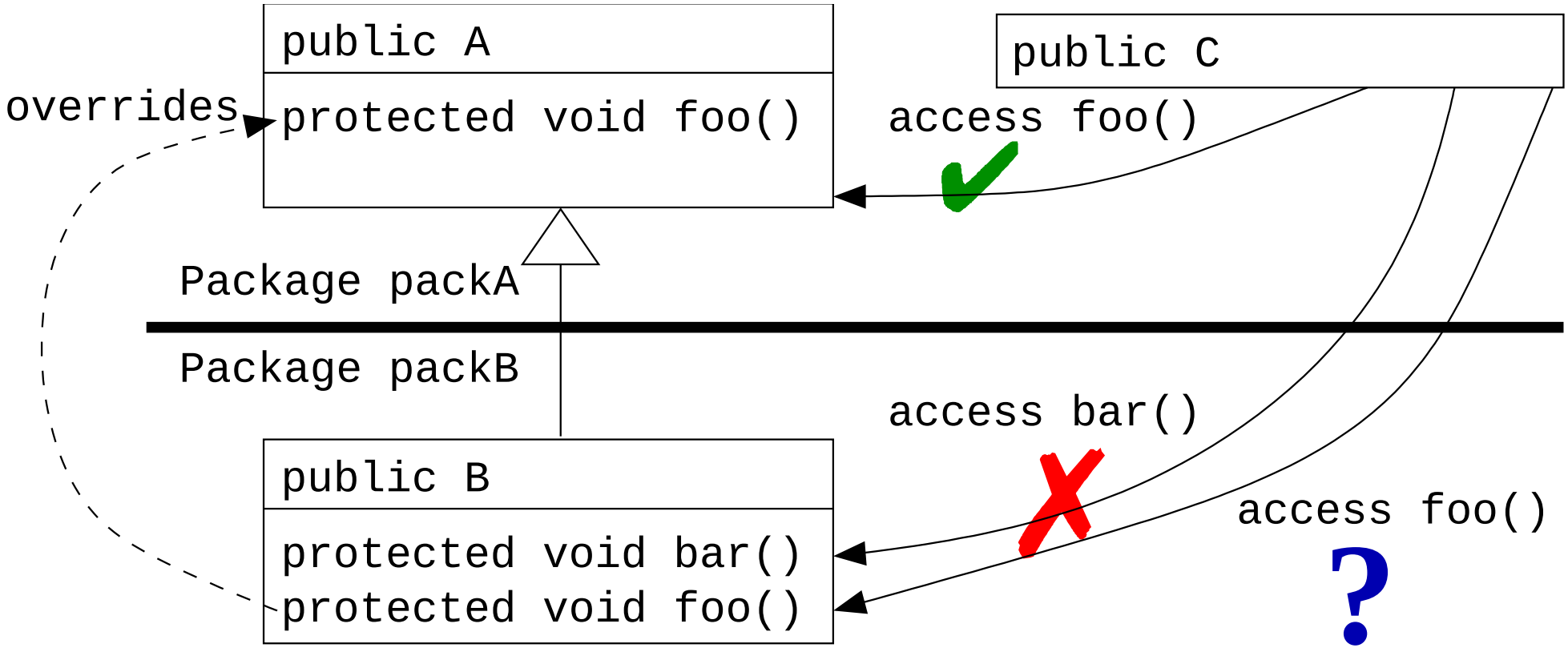
$$\frac{G \vdash m \; member\text{-}of \; C \quad G \vdash m \; in \; C \; permits\text{-}acc\text{-}to \; accC \quad G \vdash Class \; C \; accessible\text{-}in \; pid \; accC}{G \vdash m \; of \; C \; accessible\text{-}from \; accC}$$

Example:



JavaCard:
A package-visible class that is extended by a public class cannot define any public or protected methods

Overriding and Accessibility



Until now: `C` **can only access** `protected` **members declared in package** `packA` **and in subclasses of** `C`**!**

Java: Overriding is stronger then information hiding.

Accessible Member (revised)

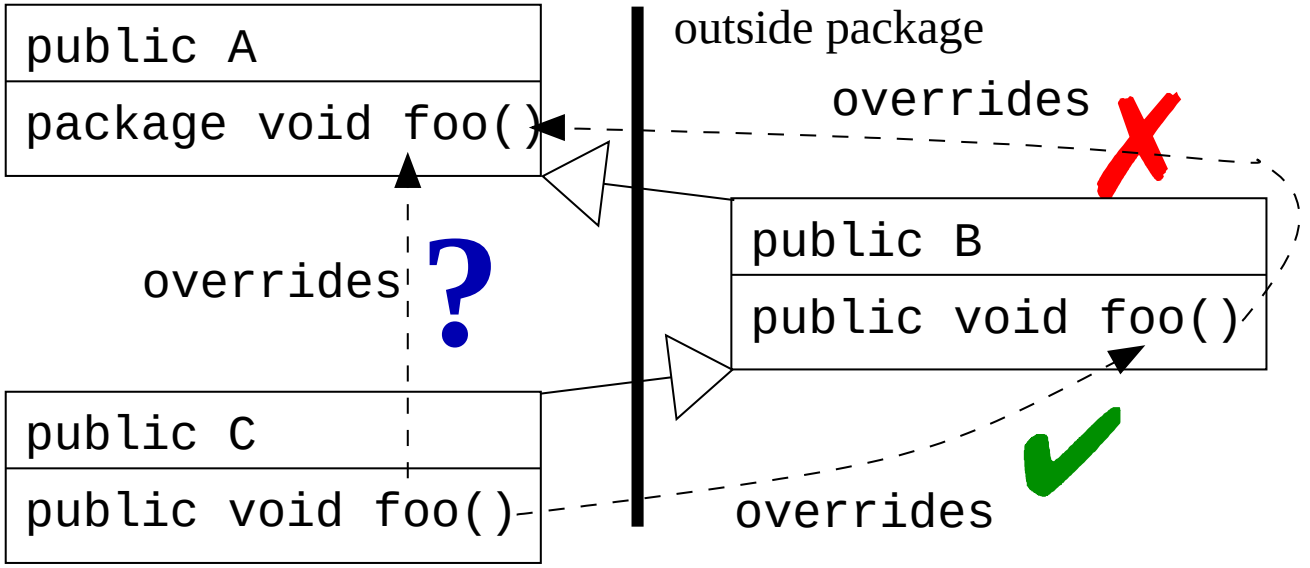
$$\frac{\begin{array}{c} G \vdash m \ member\text{-}of \ C \\ G \vdash m \ in \ C \ permits\text{-}acc\text{-}to \ accC \\ G \vdash Class \ C \ accessible\text{-}in \ pid \ accC \end{array}}{G \vdash m \ of \ C \ accessible\text{-}from \ accC} \ (\textsc{Immediate})$$

$$\frac{\begin{array}{c} G \vdash new \ overrides_S \ old \\ is\text{-}method \ new \\ G \vdash new \ member\text{-}of \ C \\ G \vdash C \prec_C S \\ G \vdash old \ of \ S \ accessible\text{-}from \ accC \\ G \vdash Class \ C \ accessible\text{-}in \ pid \ accC \end{array}}{G \vdash new \ of \ C \ accessible\text{-}from \ accC} \ (\textsc{Overriding})$$

new overrides old?

Preconditions for overriding:

$$msig\ new = msig\ old \qquad \neg\ is\text{-}static\ new$$
$$G \vdash declclass\ new \prec_C declclass\ old$$
$$G \vdash Method\ old\ inheritable\text{-}in\ pid\ (declclass\ new)$$



Static No

$$\frac{G \vdash declclass\ new \prec_{C1} S \qquad G \vdash Method\ old\ member\text{-}of\ S}{G \vdash new\ overrides_S\ old}$$

Dynamic Yes

$$\frac{resTy\ new = resTy\ old \qquad accmodi\ new \neq Private \qquad \neg\ is\text{-}static\ old}{G \vdash new\ overrides\ old}$$

Dynamic Accessibility

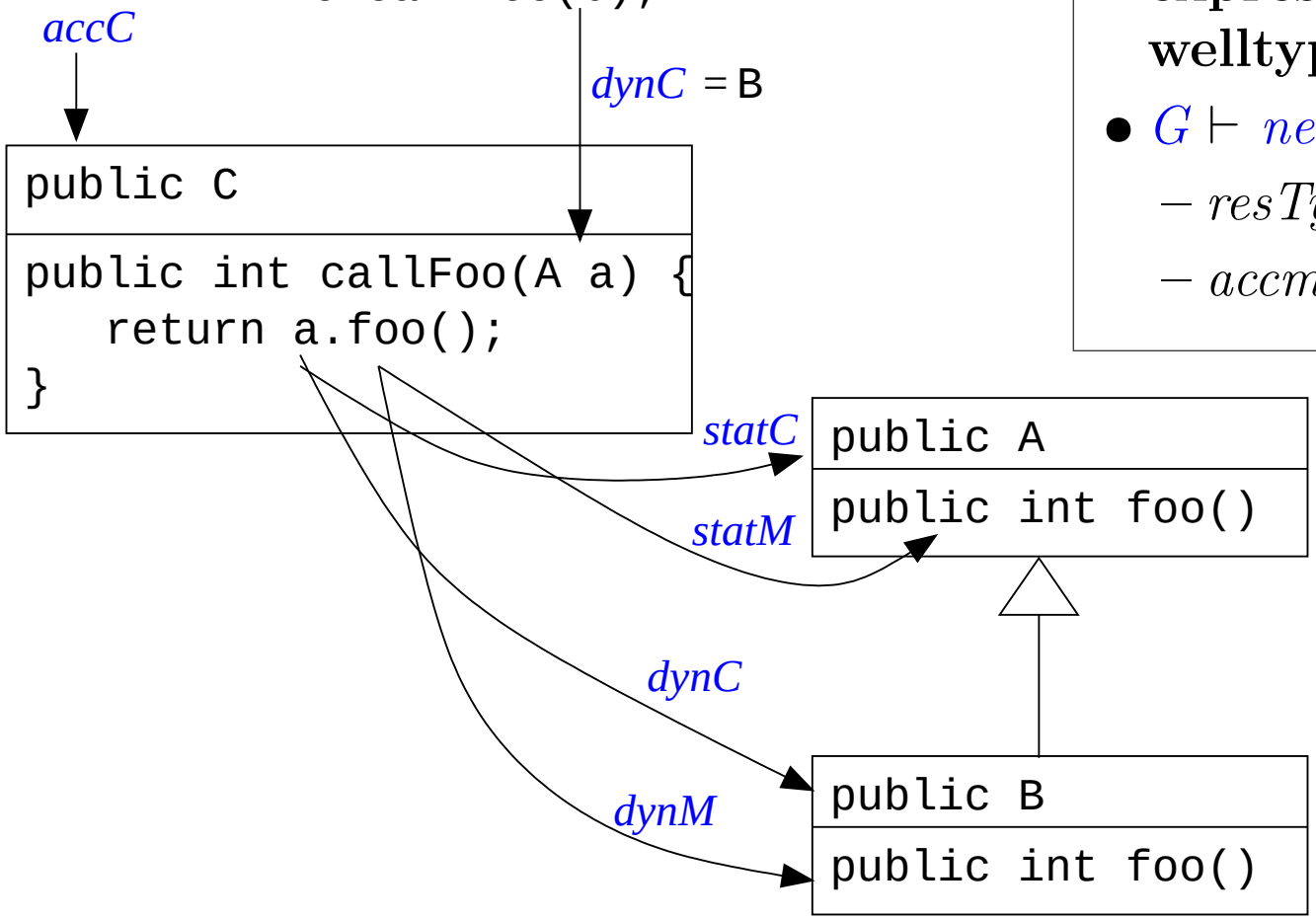
$$\frac{\begin{array}{c} G \vdash m \ \mathit{member\text{-}in}\ C \\ \sout{G \vdash \mathit{Class}\ C\ \mathit{accessible\text{-}in}\ \mathit{pid}\ \mathit{accC}} \\ G \vdash m \ \mathit{in}\ C \ \mathit{permits\text{-}acc\text{-}to}\ \mathit{accC} \end{array}}{G \vdash m \ \mathit{in}\ C \ \mathit{dyn\text{-}accessible\text{-}from}\ \mathit{accC}}\ (\textsc{Immediate})$$

$$\frac{\begin{array}{c} G \vdash \mathit{new}\ \mathit{member\text{-}in}\ C \\ \sout{G \vdash \mathit{Class}\ C\ \mathit{accessible\text{-}in}\ \mathit{pid}\ \mathit{accC}} \\ \mathit{is\text{-}method}\ \mathit{new} \\ G \vdash \mathit{new}\ \mathit{overrides}_{\not\$}\ \mathit{old} \\ G \vdash C \prec_C S \\ G \vdash \mathit{old}\ \mathit{in}\ S\ \mathit{dyn}\text{-}\mathit{accessible\text{-}from}\ \mathit{accC} \end{array}}{G \vdash \mathit{new}\ \mathit{in}\ C \ \mathit{dyn\text{-}accessible\text{-}from}\ \mathit{accC}}\ (\textsc{Overriding})$$

Access: Static to Dynamic

Theorem: $G \vdash statM \ of \ statC \ accessible\text{-}from \ accC \wedge wf\text{-}prog \ G \overset{Type\ Safety}{\Longrightarrow} G \vdash dynM \ in \ dynC \ dyn\text{-}accessible\text{-}from \ accC$

```
B b = new B();
C c = new C();
c.callFoo(b);
```



wf-prog G

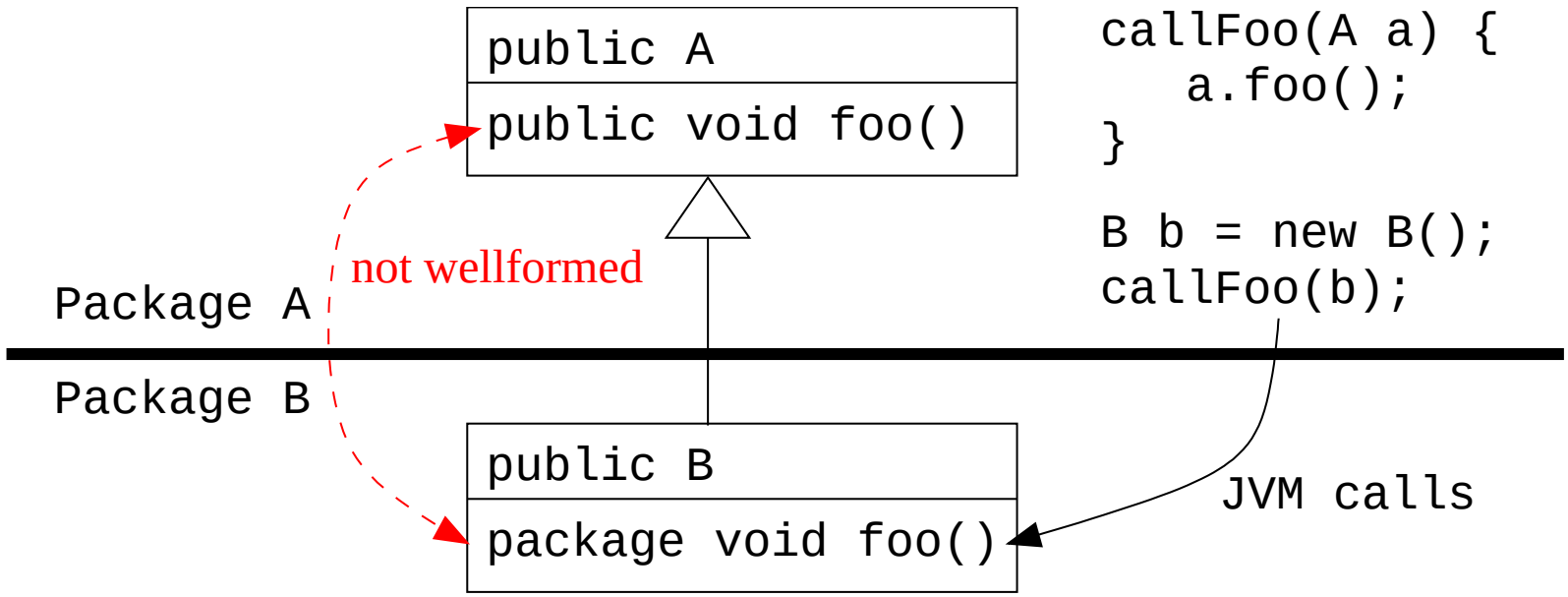
- **expressions and statements welltyped**
- $G \vdash new \ overrides_S \ old$
  - $resTy \ new = resTy \ old$
  - $accmodi \ old \leq accmodi \ new$

Properties of Dynamic Access

Lemma:
$$G \vdash m \text{ in } C \text{ dyn-accessible-from } accC \wedge accmodi\ m = Private \Longrightarrow accC = C$$

Lemma:
$$G \vdash m \text{ in } C \text{ dyn-accessible-from } accC \wedge accmodi\ m = Package \wedge wf\text{-}prog\ G \Longrightarrow pid\ accC = pid\ declclass\ m$$

But JVM does not check the wellformedness condition $accmodi$ `A.foo()` $\leq$ $accmodi$ `B.foo()`:



Conclusions

- ***Complicated semantics of package/access concept***
  - Subtle interaction of inheritance, overriding and accessibility
  - Different static and dynamic behavior

For clearer future language design:

- ***Separation of different concepts***
  - Modules
  - Subtyping
  - Implementation Inheritance

References

- T. Jensen, D. Le Métayer and T. Thorn, Security and Dynamic Class Loading in Java: A Formalisation, *IEEE International Conference on Computer Languages*, 1998, 4–15.
- James Gosling, Bill Joy, Guy L. Steele Jr. and Gilad Bracha, *The Java Language Specification, Second Edition*, Addison Wesley, 2000.

The End

Static Overriding

$$\frac{\begin{array}{c} msig\ new = msig\ old \qquad \neg\ is\text{-}static\ new \\ G \vdash declclass\ new \prec_C declclass\ old \\ G \vdash Method\ old\ declared\text{-}in\ declclass\ old \\ G \vdash Method\ new\ declared\text{-}in\ declclass\ new \\ G \vdash Method\ old\ inheritable\text{-}in\ pid\ (declclass\ new) \\ G \vdash declclass\ new \prec_{C1} S \\ G \vdash Method\ old\ member\text{-}of\ S \end{array}}{G \vdash new\ overrides_S\ old}\ (\textsc{Direct})$$

$$\frac{\begin{array}{c} G \vdash new\ overrides_S\ inter \\ G \vdash inter\ overrides_S\ old \end{array}}{G \vdash new\ overrides_S\ old}\ (\textsc{Indirect})$$

Dynamic Overriding

$$\frac{\begin{array}{c} msig\ new = msig\ old \qquad \neg\ is\text{-}static\ new \\ \neg\ is\text{-}static\ old \\ G \vdash declclass\ new \prec_C declclass\ old \\ G \vdash Method\ old\ declared\text{-}in\ declclass\ old \\ G \vdash Method\ new\ declared\text{-}in\ declclass\ new \\ G \vdash Method\ old\ inheritable\text{-}in\ pid\ (declclass\ new) \\ resTy\ new = resTy\ old \\ accmodi\ new \neq Private \end{array}}{G \vdash new\ overrides\ old}\ (\textsc{Direct})$$

$$\frac{\begin{array}{c} G \vdash new\ overrides\ inter \\ G \vdash inter\ overrides\ old \end{array}}{G \vdash new\ overrides\ old}\ (\textsc{Indirect})$$