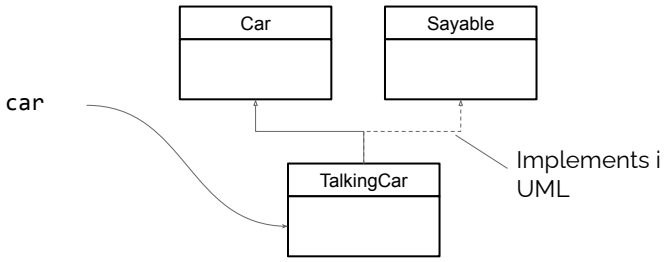
ex4types

Joachim von Hacht

Typomvandling till Gränssnittstyp

```
// Car doesn't implement Sayable
Car car = new Car();
Sayable s = (Sayable) car;   // Compiles, why is this allowed ??? ...

// TalkingCar <: Car
car = new TalkingCar();    // Because could be like this!
Sayable s = (Sayable) car;     // ... sub class may implement!
```



2

Vilket typ som helst kan explicit typomvandlas till en gränssnittstyp!!!

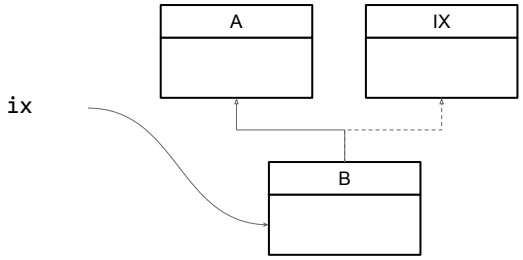
- Trots att objektet kanske inte implementerar gränssnittet d.v.s. inte är en subtyp.
  - Kompilering tillåten eftersom någon subklass kanske implementerar interfacet!
- Om objektet inte implementerar blir det senare ett körningsfel, ClassCastExeption

Typomvandlingar från Gränssnittstyp

```
IX ix = ...;
A a = (A) ix;   // No super sub, but allowed?!?!
ix = new B();   // ... because could have been like this this i.e.

public interface IX {}

public class A {
   // Class has **nothing** to do with IX
}
public class B extends A implements IX{
}
```



3

En variabel av gränssnittstyp kan omvandlas till vilken typ som helst!!!

- Eftersom variabeln kan referera ett objekt som faktisk implementerar gränssnittet.
- Annars körningsfel

Sammanfattning <: Primitiva typer

Typerna inbyggda i språket. Super/sub relationen också inbyggd i språket.

... char <: int <: float <: double ...

boolean (ingen super/sub)

Sammanfattning <: Referenstyper

Typsystemet utbyggbart

Grundregel: Inget super/sub mellan några typer.

Undantag 1: För alla typer T så T <: Object och null <: T
Specialfall 1: Om S <: T så S[] <: T[] (the array loop hole)
Specialfall 2: Typomvandling från/till gränssnitt (interface) alltid tillåtet (oavsett super/sub)

Vi kan införa en super/sub relation:
Om A implements I så A <: I
Om B extends A så B <: A

Exempel <:

```
int <: double           // For values not variables
int[] i NOT <: double[]    // 2. Primitive arrays
Integer i NOT <: Double d  // 3. Class (reference) types
A <: A                     // 4.All types subtype to self
none (except null) <: enum  // 5. enum a class type
Integer[] <: Object[]       // 6. Reference arrays!!!

interface B NOT <: interface A  // 7. Interface types
Box<Integer> NOT <: Box<Object>  // 8. Generic types
B implements A <: interface A   // 9. Implements
B extends A <: class A         // 10. Extends
```

Sammanfattning för super/subtyp-relationen. Rad för rad:

1. int värden är subtyp till double-värden (variabler har ingen <: relation eftersom de har olika roller.
2. Arrayer av primitiv typ har inget <: relation eftersom int variabler inte är subtyp till double variabler. Skulle dessutom leda till hål i typsystemet, se nedean.
3. Referenstyper har inget <: förhållande (samma som Player NOT <: Dog)
4. Alla typer är subtyper till sig själva.
5. enum har inga subtyper alls (någonsin) utom den namnlösa null-typen (null-värdet)
6. Arrayer av referenstyp är subtyper!!! "The array loophole". Tvingar Java att under körning kontrollerar vad vi stoppar in i arrayen, kan leda till ArrayStoreException
7. Gränssnitt har inte någon super/sub-relation.
8. Generiska typer har inget <: förhållande (trots att t.ex. Integer <: Object)
9. En klass som implementerar ett gränssnitt blir subtyp till gränssnittstypen.
10. En klass som ärver blir subtyp till klassen den ärver ifrån.

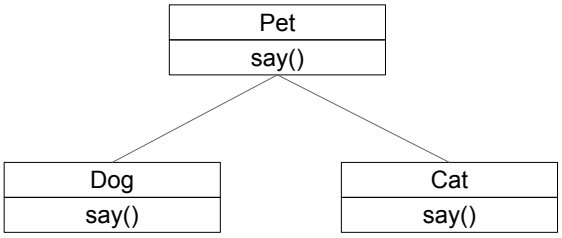
Implicit och Explicit Typomvandling

Om det finns super/sub förhållande:

- Sker, vid behov, automatiskt omvandling från sub till super d.v.s. implicit typomvandling
- Kan man göra en explicit typomvandling m.h.a. typomvandlingsoperatorn.
  - Kan leda till körningsfel.
- Undantag: Man får alltid typomvandla till/från gränssnittstyp

Sammanfattning

Polymorfism



8

Ett sammanfattande begrepp för “att det som händer beror på de inblandade typerna” är
polymorfism. Vi har sett följande

- Vilken metod som körs beror på parametrarnas typer, överlagring/overloading
- Vilken metod som körs beror på objektets typ, överskuggning/override
- Generiska typer.
- Mer i följande kurser.