EDAA45 Programmering, grundkurs

Läsvecka 7: Arv

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

7 Arv

TODO: Begrepp att förklara

Tänk igenom ordningen:

- OO, arv, supertyp, subtyp, bastyp, polymorfism, ...

Medlemmar och arv

Olika sorters medlemmar i **Scala**:

- `def`
- `val`
- `lazy val`
- `var`

Olika sorters medlemmar i **Java**:

- variabel
- metod

Variabler kan vara instansvariabler eller klassvariabler (nyckelord `static`)

Medlemmar och arv

Olika sorters medlemmar i **Scala**:

- `def`
- `val`
- `lazy val`
- `var`

Olika sorters medlemmar i **Java**:

- variabel
- metod

Variabler kan vara instansvariabler eller klassvariabler (nyckelord `static`)

- Vid arv kan man överskugga (eng. ***`override`***) en medlem, så att medlemmen med samma namn i en subtyp får sin egen speciella skepnad.
- När man konstruerar ett objektorienterat språk gäller det att man definierar sunda överskuggningsregler vid arv.

Regler för `override` i Scala.

En medlem M1 i en supertyp får ersättas av en medlem M2 i en subtyp, givet reglerna:

1. M1 och M2 ska ha samma namn och typerna ska matcha.
2. **`def`** får bytas ut mot: **`def`**, **`val`**, **`var`**, **`lazy val`**
3. **`val`** får bytas ut mot: **`val`**, och om M1 är abstrakt mot en **`lazy val`**.
4. **`var`** får bara bytas ut mot en **`var`**.
5. **`lazy val`** får bara bytas ut mot en **`lazy val`**.
6. Om en medlem i en supertyp är abstrakt *behöver* man inte använda nyckelordet **`override`** i subtypen. (Men det är bra att göra det ändå så att kompilatorn hjälper dig att kolla att du verkligen byter ut något.)
7. Om en medlem i en supertyp är konkret *måste* man använda nyckelordet **`override`** i subtypen, annars ges kompileringsfel.
8. M1 får inte vara **`final`**.
9. M1 får inte vara **`private`** eller **`private[this]`**, men kan vara **`private[X]`** om M2 också är **`private[X]`**, eller **`private[Y]`** om X innehåller Y.
10. Om M1 är **`protected`** måste även M2 vara det.

När använda en trait och när använda en klass som supertyp?

TODO

Använd en trait som supertyp om:

1. Du är osäker på vilket som är bäst. (Du kan alltid ändra till en klass senare.)
2. etc
3. etc

Använd en klass om:

1. Du vill ge supertypen en parameter vid konstruktion.
2. etc
3. etc

Designexempel: Klassen ???

TODO: