Föreläsning 15:
Utvidgning av klasser

Christian.Soderberg@cs.lth.se

18 mars 2013

Tidigare

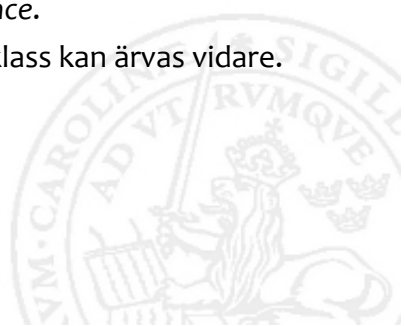
- Matriser.
- Uppdelning av program med hjälp av:
  - underprogram
  - klasser och objekt

Utvidgning av klasser

- Välj en klass som gör nästan det du vill.
- Deklarera en ny klass som utvidgar (`extends`) den valda klassen.
- Definiera de extra attribut och operationer som behövs.

Utvidgning av klasser

- Den klass vi utvidgar kallas *superklass*, den nya klassen kallas *subklass*.
- Man brukar säga att subklassen *ärver*, eller *utvidgar* superklassen.
- Det engelska ordet för arv är *inheritance*.
- Vi kan ärva i flera nivåer, dvs en ärvd klass kan ärvas vidare.



Skydd av attribut och operationer

- Objekt av den nya klassen kommer att innehålla alla ärvda attribut och operationer.
- Vi kan inte använda de attribut och operationer som är `private`-deklarerade i superklassen.
- Attribut och operationer kan skyddas i fyra nivåer:
  - `private`: kan bara användas i klassen själv.
  - `protected`: kan bara användas av klassen, och dess subklasser.
  - paket-skydd (skrivs inte ut): kan användas av alla i samma 'paket', dvs i den katalog som klassen skrivs i.
  - `public`: kan användas av alla.
- Ärvda attribut och operationer som är `private`-deklarerade kan inte användas, men de finns i objekten (och beskriver en del av objektens tillstånd)!

Att ge ärvda attribut startvärden

- Vi ger ärvda attribut sina startvärden genom att anropa konstruktorn i superklassen:

```
public GeometricTurtle(GraphicsWindow w, ...) {
    super(w,x,y);
    ...
}
```

- Om vi skall anropa superklassens konstruktor så måste vi göra det innan vi gör något annat i vår konstruktor.

Förhållande mellan klasser

- Vi kan ofta beskriva förhållandet mellan två klasser som:
  - *är-en*, eller
  - *har-en*.
- En `RandomTurtle` *är-en* `GeometricTurtle`, men den *har-ett* `Random`-objekt.
- Arv beskrivs alltså med *är-en*.

Arv i flera nivåer

- En klass kan ärvas av flera klasser.
- En klass kan bara ärva från en klass åt gången:
  - däremot indirekt från flera.

Överlagring

- Vi kan i en klass ha flera underprogram/operationer med samma namn, men olika signatur – det kallas *överlagring*.
- Klasser som ärver andra klasser kan införa operationer som överlagrar ärvda operationer.
- Det uppstår aldrig några problem när vi anropar en överlagrad operation, eftersom kompilatorn alltid kan avgöra vilken operation som avses.

Överskuggning

- Det är även tillåtet att i en klass införa en operation som har samma signatur som en ärvd operation – denna operation kommer att *överskugga* den ärvda operationen.
- Överskuggning är en vanlig och mycket användbar princip. Den gör det möjligt att ha besläktade klasser, med olika beteenden.
- Vi kan alltså ärva för att:
  - lägga till nya operationer,
  - modifiera gamla operationer.

Överskuggning och `super`

- Vi kan inifrån en klass anropa en överskuggad operation genom att anropa operationen på `super` (istället för på `this`).
- Vi använder bara det reserverade ordet `super` i två fall:
  - När vi vill anropa konstruktorn i superklassen (kan bara göras i konstruktorn, och då först i konstruktorn).
  - När vi vill anropa en överskuggad operation.

Överlagring

- Vi kan i en klass ha flera metoder med samma namn, men olika signatur – det kallas *överlagring*.
- Klasser som utvidgar andra klasser kan införa metoder som överlagrar ärvda metoder.
- Det uppstår aldrig några problem när vi anropar en överlagrad metod, eftersom kompilatorn alltid kan avgöra vilken metod som avses.

Överskuggning

- Det är även tillåtet att i en klass införa en metod som har samma signatur som en ärvd metod – denna metod kommer att *överskugga* (ersätta) den ärvda metod.
- Vi kan bara överskugga metoder som vi kan 'se' (alltså inte `private`-deklarerade metoder).
- Överskuggning är en vanlig och mycket användbar teknik. Den gör det möjligt att ha besläktade klasser, med olika beteenden.
- Vi kan alltså ärva för att:
  - lägga till nya metoder,
  - modifiera befintliga metoder.

Överskuggning och `super`

- Vi kan inifrån en klass anropa en överskuggad metod genom att anropa metoden på `super` (istället för på `this`).
- Vi använder bara det reserverade ordet `super` i två fall:
  - När vi vill anropa konstruktorn i superklassen (kan bara göras i konstruktorn, och då först i konstruktorn).
  - När vi vill anropa en överskuggad metod.