EDAA45 Programmering, grundkurs

Läsvecka 3: Funktioner, objekt

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

3 Funktioner, objekt

- Kursombud
- Funktioner
- Objekt
- Funktioner är objekt
- Rekursion

Kursombud

Fastställa kursombud

- Glädjande nog är det många intresserade!
- Instruktioner från studierådet:
  min 2 max 4 D-are
  min 2 max 4 W-are
- Vi lottar med lite lajvkodning inspirerat av:

```
1 scala> val kursombud = Vector("Kim Finkodare", "Robin Schnellhacker")
2 scala> scala.util.Random.shuffle(kursombud).take(1)
```

Funktioner

Deklarera funktioner

Exempel på deklaration av två olika funktioner:

- En parameter, och sedan två parametrar:

```
scala> def öka(a: Int): Int = a + 1
scala> def öka(a: Int, b: Int) = a + b
```

- Båda ovan funktioner kan finnas samtidigt: De är **olika** funktioner, eftersom parameterlististorna är olika med avseende på antal och typ. Trots att de har samma namn är det två olika funktioner, och kompilatorn kan skilja dem åt med hjälp av parameterlistorna.
- Detta kallas **överlagring** (eng. *overloading*) av funktioner.

Funktioner med default-argument

- Vi kan ofta åstadkomma något som liknar överlagring, men med en enda funktion, om vi i stället använder **default-argument**:

```
scala> def inc(a: Int, b: Int = 1) = a + b
inc: (a: Int, b: Int)Int

scala> inc(42, 2)
res0: Int = 44

scala> inc(42, 1)
res1: Int = 43

scala> inc(42)
res2: Int = 43
```

- Om argumentet utelämnas och det finns ett default-argumentet, så är det default-argumentet som appliceras på parametern.

Aktiveringspost

Lokala funktioner

Värdeanrop och namnanrop

Uppdelad parameterlista

Skapa din egen kontrollstruktur

Funktioner med namngivna argument

Funktioner är äkta värden i Scala

- En funktioner är ett äkta värde; vi kan till exempel tilldela en variabel ett funktionsvärde:
- Funktioner har en typ precis som alla värden:

Anonyma funktioner

Applicera funktioner på element i samlingar

Stegade funktioner, ”Curry-funktioner”

Objekt

Objekt som modul

- Ett **`object`** användas ofta för att samla **medlemmar** som hör ihop och ge dem en egen **namnrymd**.
- Medlemmarna kan vara t.ex.:
  - **`val`**
  - **`var`**
  - **`def`**
- Ett sådant objekt kallas även för **modul**.[1]

[1] Även paket som skapas med **`package`** har en egen namnrymd och är därmed också en slags modul. Objekt kan alltså i Scala användas som ett alternativ till paket; en skillnad är att objekt kan ha tillstånd och att objekt inte skapar underkataloger vid kompilering.
en.wikipedia.org/wiki/Modular_programming

Vad är ett tillstånd?

Lata variabler och fördröjd evaluering

Vad är egentligen skillnaden mellan `val`, `lazy val`, `var`, `def`?

En funktion som finns inuti ett objekt är en **metod**.

Funktioner är objekt

Programmeringsparadigm

Funktioner är äkta objekt i Scala

Scala visar hur man kan **förena** (eng. *unify*) objekt-orientering och funktionsprogrammering på ett elegant & pragmatiskt sätt:

En funktion är ett objekt som har en `apply`-metod.

Rekursion

Rekursiva funktioner

Rekursiva datastrukturer