

# EDAA45 Programmering, grundkurs

## Läsvecka 4: Datastrukturer

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

## 4 Datastrukturer

- Vad är en datastruktur?
- Tupler
- Klasser
- Case-klasser
- Samlingar

# Vad är en datastruktur?

# Vad är en datastruktur?

- En datastruktur är en struktur för organisering av data som...
  - kan innehålla **många** element,
  - kan refereras till med **ett** enda namn, och
  - ger möjlighet att komma åt de enskilda elementen.
- En **samling** (eng. *collection*) är en datastruktur som kan innehålla många element av **samma typ**.
- Exempel på **färdiga samlingar** i Scalas standardbibliotek där elementen är organiserade på olika vis så att samlingen får olika egenskaper som passar **olika användningsområden**:
  - `scala.collection.immutable.Vector`
  - `Array`
  - `List`
  - `Set`
  - `Map`

# Olika sätt att skapa datastrukturer

## ■ Tupler

- samla  $n$  st datavärden i element **\_1**, **\_2**, ... **\_n**
- elementen kan vara av **olika** typ

## ■ Klasser

- samlar data i **attribut** med (väl valda!) namn
- attributen kan vara av **olika** typ
- definierar även **metoder** som använder attributen (operationer på data)

## ■ Färdiga samlingar

- speciella klasser som samlar data i element av **samma** typ
- finns ofta *många* färdiga **bra-att-ha-metoder**
- Exempel: `scala.collection.immutable.Vector`

## ■ Egenutvecklade samlingar

- → Föreläsning

# Denna vecka: Förstå datastrukturer

- Läs teori
- Gör övning data
- Gör lab ???

# Tupler

# Vad är en tupel?

`("hej", 42, math.Pi)` är en 3-tupel med typ:  
`(String, Int, Double)`



# Klasser

# Vad är en klass?

# Case-klasser

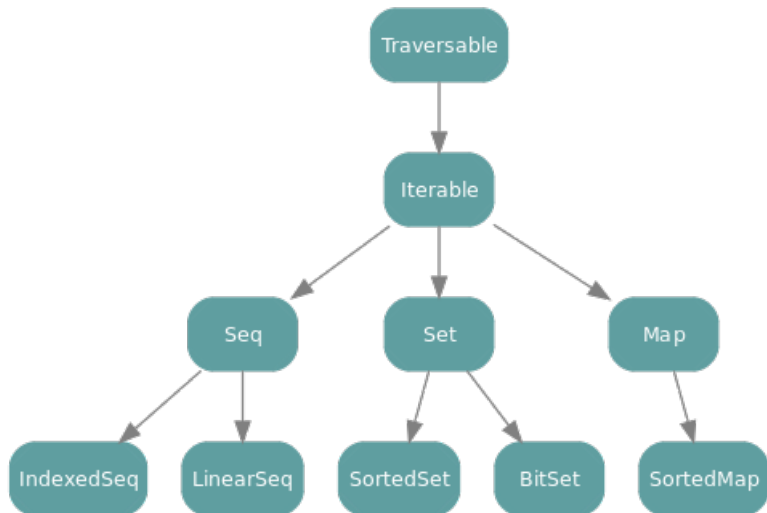
# Vad är en case-klass?

# Samlingar

# Vad är en samling?

En **samling** (eng. *collection*) är en datastruktur som kan innehålla många element av **samma typ**.

# Hierarki av samlingar i scala.collection

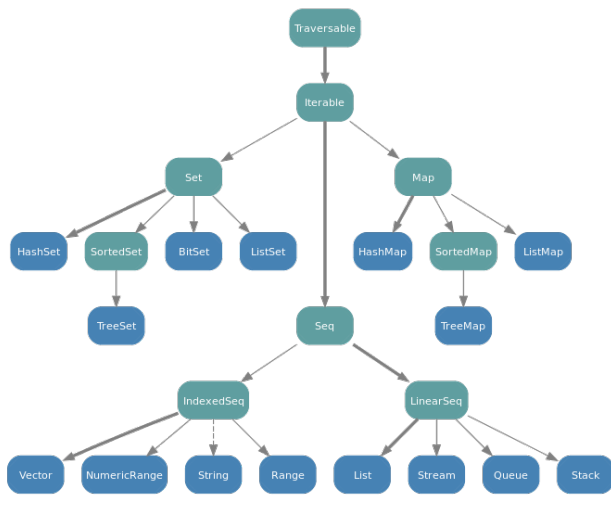


Läs mer om Scalas samlingar här:

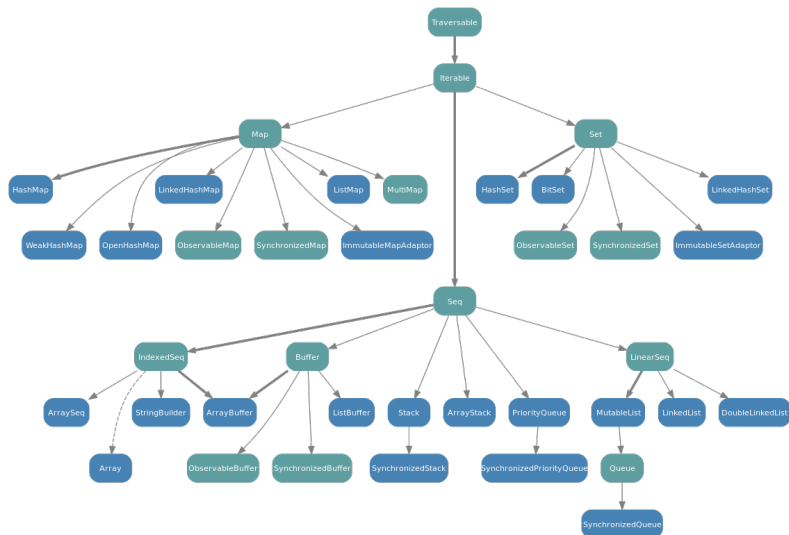
[http://docs.scala-lang.org/overviews/  
collections/overview](http://docs.scala-lang.org/overviews/collections/overview)



# scala.collection.immutable



# scala.collection.mutable



# Vector eller List???

[http://stackoverflow.com/questions/6928327/](http://stackoverflow.com/questions/6928327/when-should-i-choose-vector-in-scala)

when-should-i-choose-vector-in-scala

”

- We only need to transform sequences by operations like map, filter, fold etc: basically it does not matter, we should program our algorithm generically and might even benefit from accepting parallel sequences. For sequential operations List is probably a bit faster. But you should benchmark it if you have to optimize.
- We need a lot of random access and different updates, so we should use vector, list will be prohibitively slow.
- We operate on lists in a classical functional way, building them by prepending and iterating by recursive decomposition: use list, vector will be slower by a factor 10-100 or more.
- We have an performance critical algorithm that is basically imperative and does a lot of random access on a list, something like in place quick-sort: use an imperative data structure, e.g. ArrayBuffer, locally and copy your data from and to it.