Vecka 13. Repetition

Programmering, grundkurs (pgk)

Björn Regnell

Datavetenskap, LTH, Lunds universitet
https://cs.lth.se/pgk

EDAA45, Lp1-2, HT2024

Kompilerad den 14 maj 2025

13 Repetition

- Repetition på begäran
- Tentatips
- Utblick och avslutning

Repetition på begäran

På begäran 2024

Grumligt

1. När är det bra/dåligt att använda anonyma funktioner? w03
2. Klasser och kompanjonsobjekt: vad passar bäst var? w05
3. Hur göra felhantering med `Option` och `Try`? w06
4. Skillnaden mellan sats & uttryck, tex **`if`**, **`for`**? w01

Nyfiken-på

1. Flertrådad programmering
2. Fönsterhantering i introprog under huven
3. Generiska typgränser **`<: >:`**

Repetition: Tumregler/tips vid val av abstraktion

Extensionsmetod, singelobjekt, case-klass, klass, trait, eller enum?

- Om du vill lägga till en metod på befintlig typ utan behov av nya attribut etc., använd `extension`.
- Använd `object` om du behöver samla metoder (och variabler) i en modul som bara finns i en upplaga. Du får lokal namnrymd och punktnotation på köpet.
- Behöver du modellera **oföränderlig data**, använd en `case class` eller `enum`.
- Om du vill ha uppräknade värden som du vill kunna iterera över och matcha på i förseglad struktur, med värden i egen namnrymd, använd `enum`.
- Med `case class` och `enum` får du även innehållslikhet och en massa annat godis på köpet!
- Behöver du **förändringsbart tillstånd** (eng. *mutable state*) använd en vanlig `class`. Det normala är att det föränderliga tillståndet (de attribut som är föränderliga) är `private` eller `protected` och att all uppdatering och avläsning av tillståndet sker indirekt genom metoder (getters/setters/...).
- Behöver du en abstrakt bastyp använd en `trait`, speciellt om du vill ha möjlighet till inmixning. Om du vill förhindra inmixning eller underlätta användning från Java, använd `abstract class`.

Repetition: Tips om val av samling

Det är ofta enklare med oföränderliga samlingar med oföränderliga element och skapa nya samlingar vid förändring. Men för vissa algoritmer blir det enklare eller effektivare om du ändrar på plats i förändringsbar samling.

- Behöver du hantera värden i sekvens?
  - Om du klarar dig utan förändring av innehållet efter konstruktion: **`val`**-referens till `Vector`
  - Om du behöver ändra innehåll men **inte** antal element: **`val`**-referens till `Array`
  - Om du behöver ändra innehåll **och** antal element: **`var`**-referens till `Vector` och t.ex. metoden `patch`, eller **`val`**-referens till `ArrayBuffer` och t.ex. metoden `insert`
- Behöver du hantera värden `x` som ska vara unika?
  - Oföränderlig: `Set`
  - Förändringsbar: **`val`**-referens till `scala.collection.mutable.Set`
- Behöver du leta upp värden `x:Int` utifrån en nyckel av t.ex. String?
  - Oföränderlig: `Map[String, Int]`
  - Förändringsbar: **`val`**-referens till `scala.collection.mutable.Map[String, Int]`

Tentatips

Före tentan:

1. Repetera övningar och labbar i kompendiet.
2. Läs igenom föreläsningsanteckningar.
3. Studera **snabbref** **mycket noga** så att du vet vad som är givet och var det står, så att du kan hitta det du behöver snabbt.
4. Skapa och **memorera** en personlig **checklista** med programmeringsfel du brukar göra, som även inkluderar småfel, så som glömda parenteser och semikolon, och annat som en kompilator/IDE normalt hittar.
5. Tänk igenom hur du ska disponera dina 5 timmar på tentan.
6. Gör minst en extenta som om det vore **skarpt läge**:
    1. Avsätt 5 ostörda timmar (stäng av telefon, dator etc).
    2. Inga hjälpmedel. Bara snabbref.
    3. Förbered dryck och tilltugg.

På tentan:

1. Läs igenom **hela** tentan först.
   Varför? Förstå helheten. Delarna hänger ihop.
2. Notera och begrunda specifika begrepp och definitioner.
   Varför? Begreppen är avgörande för förståelsen av uppgiften.
3. Notera förenklingar, antaganden och specialfall.
   Varför? Uppgiften blir mkt enklare om du inte behöver hantera dessa.
4. **Fråga** tentamensansvarig om du inte förstår uppgiften – speciellt om det finns misstänkta felaktigheter eller förmodat oavsiktliga oklarheter.
   Varför? Det är inte lätt att konstruera en ”perfekt” tenta.
   Du får fråga vad du vill, men det är inte säkert du får svar...
5. Läs specifikationskommentarerna och metodsignaturerna i alla givna klass-specifikationer **mycket noga**.
   Varför? Det är ett vanligt misstag att förbise de ledtrådar som ges där.
6. Återskapa din memorerade personliga checklista för vanliga fel som du brukar göra och avsätt tid till att gå igenom den på tentan. Varje fix plockar poäng!
7. Lämna in ett försök även om du vet att lösningen inte är fullständig. Det gäller att plocka så många poäng det går. En ofullständig lösning kan ändå ge poäng.
8. Om du har svårigheter kan det bli kamp mot klockan. Försök hålla huvudet kallt och prioritera utifrån var du kan plocka flest poäng. Ge inte upp! Ta en kort äta-dricka-paus för att få mer energi!

Planeringstips

Exempel på saker som du kan lägga in tid för i din julpluggkalender:

1. Ta reda på vad just **du** behöver träna på!
2. Välja ut övningar att repetera.
3. Repetera övning X, Y, Z, ... Både läsa och skriva kod. Fundera på typ och värde.
4. Välja ut labbar att repetera.
5. Repetera labb X, Y, Z, ... Lär dig ”trick” och ”mönster”.
6. Träna på att skriva program med papper och penna.
7. Gör så många extentor du orkar, simulera ”skarp läge”.
8. Gör en checklista med vanliga fel och misstag som du brukar göra.
9. Läsa igenom alla de extentor som du väljer att inte göra ”i fiktivt skarpt läge” och studera generella mönster och typiska trick.

Tentans struktur

- Del A 20%:
  Evaluera uttryck där du ska **ange typ och värde**
  - Testar förståelse av variabler, uttryck, samlingar, algoritmer, arv, etc.
  - Det är bra/nödvändigt att anteckna delsteg och variablers värden, då det är mycket svårt att tänka ut svaren direkt i huvudet.
- Del B 80%:
  Skriva kod som uppfyller **krav och design**
  - Testar att du själv kan skapa kod med delar som samverkar
  - Testar förmåga att gå från indata-utdata till algoritm
    givet: ledtrådar, design, ev. skiss på lösning, ev. pseudokod etc.
- Blanka inlämningar ger 0 poäng; det är alltid bättre att försöka än att lämna in blankt. Lämna inte in kladdpapper eller dubbla lösningar.

Vad kommer på tentan?

- Grundläggande begrepp och det som tränas på grundövningar och labbar är basen för att bli godkänd.
- Begrepp, föreläsningsbilder och övningar som är markerade **”fördjupning”** krävs ej för att klara tentan men ökar förståelsen och hjälper dig att nå högre betyg.
- Det är helt ok på tentan om du väljer en **enkel lösning med basala begrepp som fungerar bra**, i stället för en kortare/elegantare/mer avancerad lösning.
- Extra-övningarna i läsvecka 12 ingår ej på tentan.

Utblick och avslutning

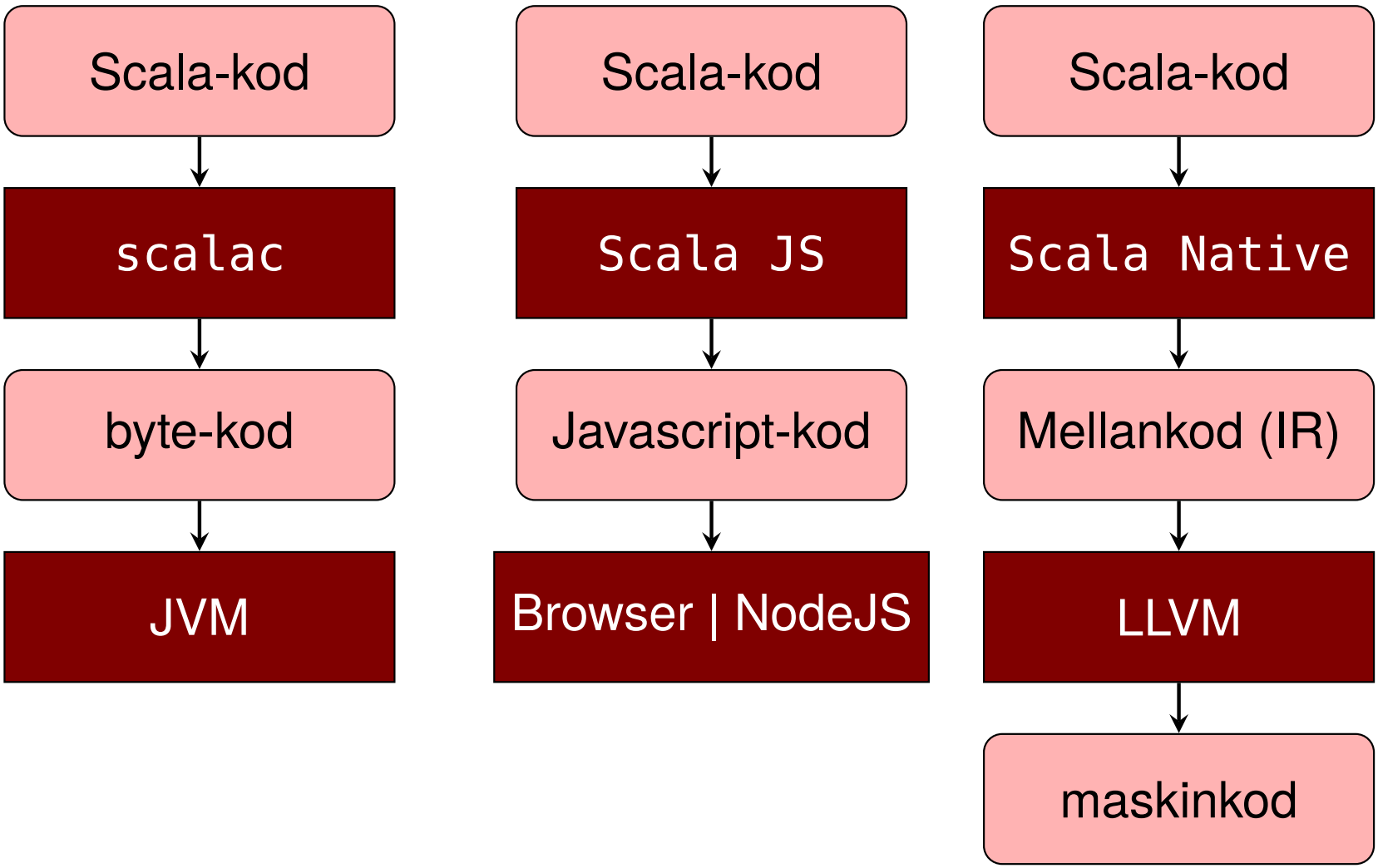
Scala då, nu och i framtiden

- Scala 1.0 (2003) första pre-release
- Scala 2.0-2.9 (2006-2011) pionjärer: Twitter, LinkedIn, The Guardian, ...
- Scala 2.10 (2013) brett genombrott, viktiga språkutvidgningar
- Scala 2.11 (2014) allmän industriell spridning, stabilitet, prestanda,
- Scala 2.12 (2016) fokus på prestanda, snabbare bytekod med lambda i JVM Java 8
- Scala 2.13 (2019) fokus på standardbiblioteket och `scala.collection`, migreringsverktyg för Scala 3
- Scala 3 (2021): **stort tekniksprång** med många nya språkkonstruktioner

Läs mer om historik här:

`https://en.wikipedia.org/wiki/Scala_(programming_language)`

Scala på JVM, Scala JS, Scala Native



Hur håller jag mig uppdaterad om Scalas utveckling?

- Officiell blog: `https://www.scala-lang.org/blog/`
- Scala-nyheter: `http://scalatimes.com/`
- User-forum: `https://users.scala-lang.org/`
- Tjatt: `https://discord.gg/h9452YPJ`
- Scala Center: `https://scala.epfl.ch/`
- Scala-bibliotek: `https://index.scala-lang.org/`
- Contributors: `https://contributors.scala-lang.org/`
- Scala-språkets pågående förbättring: `https://docs.scala-lang.org/sips`

CEQ – Course Experience Questionnaire

- Görs på hela LTH på samma sätt. Alla får länkar via mejl.
- Snälla fyll i CEQ! Jag är **mycket tacksam** för all konstruktiv feedback! Hög svarsfrekvens är viktigt för att kunna dra slutsatser om variationen i svaren och signifikansen i sammanställningen.
- Del 1: Generella påståenden, alla med 5-gradig skala:
  tar helt avstånd ... instämmer helt
- Del 2: **Fritextfrågor**:
  ”Vad tycker du var det bästa med den här kursen?”
  ”Vad tycker du främst behöver förbättras?”
- Mer om CEQ här: `https://www.ceq.lth.se/`
- **Fördel** med CEQ: Samma alla kurser alla år medger jämförelse över tid.
- **Begränsning**: Saknar frågor kopplat till specifika kursmoment.

Kursspecifik utvärdering om specifika kursmoment

- Jag vill gärna att **alla** gör den LTH-gemensamma, anonyma kursutvärderingsenkäten CEQ. Dina fritext-kommentarer om vad som är det bästa med kursen och vad som främst behöver förbättra emottages mycket tacksamt i CEQ-utvärderingen!
- Jag kommer att komplettera CEQ med en **kursspecifik utvärdering** av specifika kursmoment i denna kurs och jag är därför **mycket tacksam** om alla fyller enkäten när länk kommer via anslag i Canvas.
- Jag behandlar dina svar **konfidentiellt**, men sparar din email så att jag kan återkomma om jag mot förmodan undrar något mer.
- Din input är **mycket värdefull** vid framtida kursutveckling!

Intresserad av att arbeta som handledare?

- Vi har ständigt behov av nya handledare i våra kurser
- Det är lärorikt att jobba som handledare
- Kontakta `bjorn.regnell@cs.lth.se`
  eller annan kursansvarig i den kurs du vill jobba
- Jag påbörjar intervjuer av kandidater pgk+dod **tidig vår**

`https://cs.lth.se/utbildning/`

”Arbeta som Övningsassistent”

Ett stort TACK...

- ... till alla **handledare** som jobbat hårt för att ni ska lära er så mycket som möjligt!
- ... till alla **studenter** som gått kursen för:
  - ... att ni kämpat så hårt!
  - ... att ni ställt massor med frågor!
  - ... att det har varit så hög närvaro på föreläsningarna!
  - ... att ni hjälp till med värdefull återkoppling!
  - ... att ni är så konstruktiva och verkligen vill lära er!

Ett stort TACK...

- ... till alla **handledare** som jobbat hårt för att ni ska lära er så mycket som möjligt!
- ... till alla **studenter** som gått kursen för:
  - ... att ni kämpat så hårt!
  - ... att ni ställt massor med frågor!
  - ... att det har varit så hög närvaro på föreläsningarna!
  - ... att ni hjälp till med värdefull återkoppling!
  - ... att ni är så konstruktiva och verkligen vill lära er!

Ett stort LYCKA TILL på vägen till att bli en kompetent och innovativ systemutvecklare!

Hoppas att pgk-kursen varit givande!