Kontrollskrivning
EDAA45 Programmering, grundkurs

2017-10-24, 14:00-19:00, Kårhusets Gasquesal

Hjälpmedel: Snabbreferens för Scala & Java.

Instruktioner

Du får ett identitetsnummer som du ska ange här, på omslaget och på alla inlämnade blad: _____
Kontrollskrivningen är indelad i tre moment:

- *Moment 1*, ca 2 h 15 min: **Lösning av uppgifterna**. Du löser uppgifterna individuellt; del A direkt i detta häfte och del B på separat papper. Du får endast lämna in *ett* svar per uppgift. Skriv med blyertspenna. Du får inte skriva med rödfärgad penna. Du ska inte lämna in kladdpapper eller anteckningar som inte ingår i dina svar. Du ska skriva ditt identitetsnummer i övre högra hörnet på *alla* inlämnade blad. När du är klar, eller när tiden är ute, lägger du dina svar inklusive detta häfte inuti omslaget och låter skrivningen ligga på din bänk. Är du klar innan sluttiden får du under tystnad ta fram en bok, din telefon, eller liknande.
- *Moment 2*, ca 1 h 15 min: **Parvis kamratbedömning**. Ni sätter er parvis enligt muntliga instruktioner och får ut bedömningsmallen som ni läser igenom noga. Efter ett tag får ni ut två andra personers skrivningar som ni poängsätter enligt anvisningarna i bedömningsmallen. När tiden är slut samlar lärare in alla skrivningarna.
- *Moment 3*, ca 0,5 h: **Bedömning av kamratbedömningen**. Du får nu inspektera din egen skrivning och värdera poängsättningen. Är du inte nöjd med poängsättningen skriver du ett kryss i motsvarande ruta på omslagets baksida och beskriver utförligt på baksidan av omslaget vad i poängsättningen du anser bör korrigeras och varför. Du får *inte* ändra i själva skrivningen, bara skriva på omslagets baksida (och omslagets insida om du behöver extra utrymme). Efter ett tag samlas skrivningen in och du kan därefter lämna lokalen. Om du vill gå i förtid, kontakta skrivningsansvarig.

Kontrollskrivningen motsvarar i omfång en halv ordinarie tentamen och är uppdelad i två delar; del A och del B. Följande poängfördelning gäller:

- Del A omfattar 20% av den maximala poängsumman och innehåller uppgifter med korta svar.
- Del B omfattar 80% av den maximala poängsumman och innehåller uppgifter med svar i form av programkod som du ska skriva på separat papper.
- Maximal poäng: 50
- Om du erhåller `p` poäng på kontrollskrivningen bidrar du med `(p / 10.0).round.toInt` i individuell bonuspoäng inför sammanräkningen av samarbetsbonus.

Den diagnostiska kontrollskrivningen påverkar inte om du blir godkänd eller ej på kursen, men det samlade poängresultatet för din samarbetsgrupp ger möjlighet till *samarbetsbonus* som kan påverka ditt betyg.

Del A. 10 p

Följande kod finns kompilerad utan kompileringsfel och tillgänglig på din *classpath*:

```
1  class A(private val i: Int){
2    def +(a: A): A = {
3      A.x += 1
4      A(i + a.i)
5    }
6    def toInt = i
7  }
8
9  object A {
10   private var x: Int = 0
11   def apply(a: Int): A = new A(i = a)
12   def n = x
13 }
14
15 case class B(xs: Vector[Int]) {
16   def +(a: A): B = B(xs.map(_ + a.toInt))
17 }
```

Du ska fylla i tabellen på nästa sida enligt följande. Antag att du skriver in nedan kod i Scala REPL rad för rad. För varje variabel med namn `u1` ... `u5`, ange statisk **typ** (alltså den typ kompilatorn härleder), samt det **värde** variabeln får efter initialisering, *eller* sätt i stället kryss i rätt kolumn om det blir ett **kompileringsfel** respektive **exekveringsfel**. Vid frånvaro av fel, svara på samma sätt som Scala REPL skriver ut typ respektive värde, enligt exempel `u0` i tabellen.

```
1 val u0 = 41.0 + 1
2 val u1 = 1 / A.x
3 val u2 = (A(1) + new A(2) + A.apply(3)).toInt / A.n
4 val u3 = B((1 to 10).toVector).xs.reverse.distinct.sorted.max
5 val u4 = B(Vector(1,2,3)) + A(-1)
6 val u5 = 1 / (Set(A(1),A(1),A(1)).map(_.toInt).size - 1)
```

	Vid kompileringsfel sätt kryss.	Vid exekveringsfel sätt kryss.	Ange statisk **typ** som kompilatorn härleder om ej kompilerings- eller exekveringsfel.	Ange det **värde** som tilldelas vid exekvering, med samma format som vid utskrift av värdets `toString`, om ej kompilerings- eller exekveringsfel.
`u0`			`Double`	`42.0`
`u1`				
`u2`				
`u3`				
`u4`				
`u5`				

Del B. 40 p

Lönekontoret vid Eslöv University har ökänt krångliga pappersblanketter för tidrapportering som alla labbhandledare måste fylla i innan de kan få ersättning för sitt arbete. Som kodkunnig student har du fått i uppdrag att hjälpa EU att automatisera lönehanteringen för labbhandledare. Ditt efterlängtade program ska skriva ut en lista med utbetalningsrapporter åt EU:s lönekontor så att labbhandledarna kan få sina pengar utan att behöva fylla i de svårbegripliga blanketterna.

Varje kursansvarig lärare åläggs att skapa en indatafil kallad `indata.csv` ur sitt elektroniska kalkylblad med labbhandledarnas arbetade timmar enligt exempel här bredvid. Varje rad i indatafilen representerar ett arbetspass och består av kommaseparerade värden i följande kolumner:

- labbhandledarens namn,
- datum för arbetspasset på formatet `2001-01-01`,
- klockslagen i hela timmar för arbetspassets början och arbetspassets slut med ett streck emellan.

```
Jakob Hök,2017-10-25,15-19
Anders Buhl,2017-10-25,17-19
Erik Grampp,2017-10-25,14-19
Oskar Berg,2017-10-12,13-15
Jakob Hök,2017-10-11,8-10
Oskar Berg,2017-10-12,10-12
Jakob Hök,2017-10-11,13-15
Anders Buhl,2017-10-11,13-15
Anders Buhl,2017-10-13,13-15
```

När ditt program körs med ett filnamn och en timlön som argument enligt nedan, ska det för varje labbhandledare skriva ut en rad med en utbetalningsrapport innehållande den sammanräknade lönen och antalet arbetade timmar. Lagen om anställningsskydd (LAS) kräver även att EU håller reda på hur många *olika* dagar som varje labbhandledare arbetat, vilket därför också ska ingå i varje rapportrad enligt nedan.

```
> scala PrintReports indata.csv 260
Jakob Hök UTBETALAS TOTALT: 2080 SEK för 8 timmar på 2 olika dagar
Anders Buhl UTBETALAS TOTALT: 1560 SEK för 6 timmar på 3 olika dagar
Erik Grampp UTBETALAS TOTALT: 1300 SEK för 5 timmar på 1 olika dagar
Oskar Berg UTBETALAS TOTALT: 1040 SEK för 4 timmar på 1 olika dagar
```

Till din hjälp har en pank labbhandledare skapat en övergripande design av programmet, med vissa delar färdigimplementerade, som du återfinner på nästa sida. Programmets utformning beskrivs nedan:

- Case-klassen `WorkShift` representerar ett arbetspass och är redan färdigimplementerad.
- Klassen `Report` kan bygga upp en sekvens av arbetspass och med hjälp av denna sekvens beräkna den sammanlagda lönen och antalet arbetsdagar. `Report` har följande medlemmar:
  - klassparametern `hourlyWage` anger timlönen i hela kronor,
  - attributet `workShifts` är en sekvens av arbetspass,
  - metoden `add` lägger till ett arbetspass `w` till sekvensen `workShifts`,
  - metoden `hours` summerar alla arbetade timmar för varje arbetspass
  - metoden `salary` räknar ut lönen (färdigimplementerad),
  - metoden `days` räknar antalet unika dagar som förekommer bland arbetspassen,
  - metoden `toString` ger en sträng med data till en utbetalningsrapport (färdigimplementerad).
- Singelobjektet `Reports` kan skapa en nyckel-värde-tabell med labbhandledarnas namn som nycklar och tillhörande rapporter som värden. `Reports` har följande medlemmar:
  - Metoden `apply` läser data från filen `file` med format enligt ovan och skapar en nyckel-värde-tabell som mappar namn till rapporter. För att splittra en inläst rad och skapa namn och arbetspass ska du implementera och använda följande två hjälpmetoder.
  - Metoden `toName` ska plocka fram det namn ur `line` som finns före första kommatecknet.
  - Metoden `toWorkShift` ska skapa ett arbetspass ur `line` genom att ta fram datum och klocklag för arbetspasset som finns efter första och andra kommatecknet.
- Huvudprogrammet `main` i singelobjektet `PrintReports` använder `Reports` för att bygga upp en tabell med raporter och skriver sedan ut den efterfrågade rapportlistan enligt exemplet ovan.

```scala
case class WorkShift(date: String, fromHour: Int, toHour: Int){
  val hours: Int = toHour - fromHour
}

class Report(val hourlyWage: Int){
  private var workShifts      = Vector.empty[WorkShift]
  def add(w: WorkShift): Unit = ???
  def hours: Int              = ???
  def salary: Int             = hours * hourlyWage
  def days: Int               = ???
  override def toString =
    s"TOTALT: $salary SEK för $hours timmar på $days olika dagar"
}

object Reports {
  def toName(line: String): String = ???
  def toWorkShift(line: String): WorkShift = ???
  def apply(file: String, hourlyWage: Int): Map[String, Report] = ???
}

object PrintReports {
  def main(args: Array[String]): Unit = if (args.length == 2) {
    val reports = Reports(file = args(0), hourlyWage = args(1).toInt)
    for (r <- reports) println(r._1 + " UTBETALAS " + r._2)
  } else println("Två argument krävs: <filnamn> <timlön i kr>")
}
```

Krav, tips och förenklingar:

- Ordningen på raderna i indatafilen ska inte spela någon roll.
- Du kan förutsätta att filen `indata.csv` existerar och följer kolumnformatet som beskrivs ovan; du ska alltså inte göra någon speciell felhantering för saknad fil eller felaktigt format.
- Ingen labbhandledare vid EU har exakt samma namn som någon annan EU-labbhandledare.
- Du kan förutsätta att arbetspassen är rimliga: ett arbetspass sträcker sig aldrig över flera dagar och klocklaget för början av ett arbetspass alltid är mindre än klockslaget för arbetspassets slut.
- Uttrycket `line.split(c)` ger en sekvens av typen `Array[String]` där varje sträng skapas ur en uppdelning av `line` i delsträngar vid varje förekomst av separatortecknet `c`. Separatortecknet tas ej med i delsträngarna i resultatsekvensen.
- Uttrycket `scala.io.Source.fromFile(file).getLines.toVector` ger en sekvens av strängar som motsvarar raderna i en textfil med filnamn enligt strängen `file`.
- Uttrycket `s.toInt` ger ett heltal genom omvandling av numeriska tecken i strängen `s`.

Uppgift B1. Report. 11p. (`add` 2p, `hours` 4p, `days` 5p)

Implementera de saknade delarna i klassen `Report` markerade med `???`.

Uppgift B2. Reports. 29p. (`toName` 3p, `toWorkShift` 10p, `apply` 16p)

Implementera de saknade delarna i singelobjektet `Reports` markerade med `???`.