Bedömningsmall för kontrollskrivning EDAA45 Programmering, grundkurs

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

Hjälpmedel: Snabbreferenser för Scala och Java.

Instruktioner för kamraträttning

- Läs igenom hela denna bedömningsmall innan bedömningen påbörjas. Sätt dig in i mönsterlösningens angreppssätt och funktion, samt hur poängsättningen ska ske.
- För varje del: läs noga lösningen du ska bedöma och sätt dig in i det specifika angreppssättet.
- Ge maxpoäng om koden är korrekt, läsbar och lika bra som mönsterlösningen även om den är annorlunda. Det finns många olika sätt att lösa varje del, varav många är lika bra som mönsterlösningen. En lösning kan vara lika bra som mönsterlösningen, även om den är längre eller kortare eller har ett annat angreppssätt.
- Ge noll poäng om lösningen saknas, är oläsbar, obegriplig eller helt fel.
- Markering av poängavdrag (negativa) och uppgiftspoäng (inringad, positiv):
  - Markeringar ska göras med en penna med avvikande färg, gärna *röd*.
  - *Stryk under* de delar av en rad som är felaktiga eller bara delvis korrekta.
  - Om något saknas, *markera med en pil* var det som saknas borde finnas och skriv en kommentar om vad som saknas.
  - Markera poängavdrag i *högerkanten* med *minuspoäng* enligt bedömningsriktlinjerna i detta häfte.
  - När du markerat relevanta avdrag för en hel uppgift, beräkna och skriv total uppgiftspoäng *inringad* i en cirkel, så att det går lätt att skilja uppgiftspoäng från poängavdrag. Uppgiftspoängen ska inte vara negativ utan ligga i intervallet [*0, Maxpoäng*].
  - Om det efter studier av de markerade poängavdrag du gjort i skrivningen och dessa riktlinjer, inte är uppenbart varför du ej givit maxpoäng skriv då en kort förklaring i kanten.
  - Om en lösning är oläslig eller helt obegriplig, skriv *”oläsligt”* eller *”obegripligt”* i kanten.
- Avsluta med att göra en *kvalitativ helhetsbedömning* av hela skrivningen och justera poäng för vissa delar om du tycker att poängsumman för hela uppgiften inte överensstämmer med din helhetsbedömning. Om du ändrar i din poängsättning, stryk över de gamla poängen med ett kryss och skriv nya poäng bredvid.
- Summera poängen och fyll i summan på omslagets framsida.
- Ange era identitetsnummer i fälten för rättare på omslagets framsida.
- Kamraträttningar med invändningar kommer att bedömas igen i efterhand av lärare. Även de kamratbedömningar som godkänns utan invändningar kommer att stickprovsbedömas i efterhand av lärare.

Del A. Totalt max 10 p

Efter init. av:	Vid kompileringsfel sätt kryss.	Vid exekveringsfel sätt kryss.	Ange statisk **typ** som kompilatorn härleder om ej kompilerings- eller körtidsfel.	Ange det **värde** som tilldelas vid exekvering, med samma format som vid utskrift av värdets `toString`, om ej kompilerings- eller körtidsfel.
`u1`	**X**			
`u2`			`Int`	`3`
`u3`			`Int`	`10`
`u4`			`B`	`B(Vector(0,1,2))`
`u5`		**X**		

Rätta en rad i taget enligt nedan och sätt minuspoäng vid felaktigt svar i marginalen, t.ex. -1 eller -2. Räkna ut totala poängen och resultatet inringat längst ner på sidan direkt i skrivningshäftet. Varje rad kan ge max 2p.

- Typerna och värdena ska skrivas som det står i tabellen ovan för full poäng. Dock får svaret skilja sig vad gäller i betydelselös användning av blanktecken och radbrytningar. Om värdet är *nästan helt rätt*, så när som på obetydliga detaljer i `toString`-representationen, till exempel `10` i stället för `10.0` för ett värde av `Double`-typ, ges inget avdrag.
- Om rätt svar är ett kryss i någon av kolumnerna för kompilerings- eller exekveringsfel, men man kryssat för inkorrekt typ av fel, ges -1 i avdrag.
- Om man helgarderat med två kryss på samma rad ges -2 i avdrag.
- Om man svarat med fel typ men rätt värde ges -1 i avdrag.
- Om man svarat med rätt typ men fel värde ges -1 i avdrag.
- Om både värde och typ är fel ges -2 i avdrag.
- Om man svarat med typ/värde *och* kryss i någon av kolumnerna för kompilerings- eller exekveringsfel på samma rad ges -2 i avdrag.

Del B. Totalt max 40 p

Allmänna bedömningsregler som gäller alla svar i del B:

- Olika formateringsvarianter som skiljer sig från mönsterlösning som ej påverkar funktionen, t.ex. extra (klammer)parentespar eller radbrytningar, ger inga avdrag.
- Felaktigt matchade (klammer)parentespar, eller glömda (klammer)parenteser ger -1 i avdrag.
- Dokumentationskommentarer och importsatser i specifikationer behöver ej upprepas i lösningen. Om ytterligare importsatser behövs men helt saknas ska -1 i avdrag ges, men smärre felaktigheter i själva sökvägen ger inga avdrag. Om en fullständig sökväg anges direkt på plats (i stället för import), ges inget avdrag om sökvägen har smärre felaktigheter.
- Felaktigt avskriven metodsignatur från specifikationen ger -1 i avdrag.
- Om nödvändigt semikolon saknas (t.ex. mellan flera satser på samma rad) ges -1 i avdrag.
- Obetydliga fel i likhet med föregående två punkter där det framgår tydligt vad som egentligen avses ger -1 i avdrag.
- Deklaration av `val` när det måste vara `var` för att koden ska fungera ger -2 i avdrag.
- Deklaration av `var` när det skulle kunna vara en `val` och fungera lika bra ger -1 i avdrag
- Tillägg av element i samling på fel sätt, t.ex. `+:` i stället för `:+` eller liknande, ger -1 i avdrag.
- Användning av metoden `length` på samling som ej är en sekvens, t.ex `Set` el. `Map`, ger -1 i avdrag.
- Enkla misstag som är lätta att åtgärda ger -1 eller -2 i avdrag, beroende på hur allvarligt felet är.
- Allvarliga misstag eller stora ofullständigheter ger från -3 upp till *–maxpoäng* i avdrag, beroende på hur mycket av koden som måste skrivas om för att det ska fungera.
- Om samma typ av fel förekommer i samma deluppgift (implementerad medlem) mer än en gång görs bara avdrag en gång vid första förekomsten. Skriv *”samma fel igen”* i kanten för att indikera att avdraget redan är gjort.
- Totala poängen för en uppgift kan inte bli negativ; om alla avdrag för en uppgift blir mer än maxpoängen ges 0 poäng.

Uppgift B1. Report. 11p. (2 + 4 + 5)

Uppgift B2. Reports. 29p. (3 + 10 + 16)

Endast de poängmarkerade delarna ingår i bedömningen:

```
1  case class WorkShift(date: String, fromHour: Int, toHour: Int){
2    val hours: Int = toHour - fromHour
3  }
4
5  class Report(val hourlyWage: Int){
6    private var workShifts      = Vector.empty[WorkShift]
7
8    def add(w: WorkShift): Unit = workShifts :+= w                        // 2p
9    def hours: Int              = workShifts.map(_.hours).sum             // 4p
10   def days: Int               = workShifts.map(_.date).distinct.length  // 5p
11
12   def salary: Int             = hours * hourlyWage
13   override def toString =
14     s"TOTALT: $salary SEK för $hours timmar på $days olika dagar"
15 }
16
17 object Reports {
18   def toName(line: String): String = line.split(',')(0)            // 3p
19
20   def toWorkShift(line: String): WorkShift = {                      // sum 10p
21     val xs = line.split(',')                                             // 2p
22     val hrs = xs(2).split('-').map(_.toInt)                              // 4p
23     WorkShift(xs(1), hrs(0), hrs(1))                                     // 4p
24   }
25
26   def apply(file: String, hourlyWage: Int): Map[String, Report] = { // sum 16p
27     val lines = scala.io.Source.fromFile(file).getLines.toVector        // 1p
28     val names = lines.map(toName)                                        // 3p
29     val result = names.map(n => n -> new Report(hourlyWage)).toMap       // 5p
30     for (line <- lines) result(toName(line)).add(toWorkShift(line))      // 6p
31     result                                                               // 1p
32   }
33 }
34
35 object PrintReports {
36   def main(args: Array[String]): Unit = if (args.length == 2) {
37     val reports = Reports(file = args(0), hourlyWage = args(1).toInt)
38     for (r <- reports) println(r._1 + " UTBETALAS " + r._2)
39   } else println("Två argument krävs: <filnamn> <timlön i kr>")
40 }
```

Specifika bedömningsriktlinjer för varje implementerad medlem:

- `add`
  - det går lika bra utan förkortad tilldelning:
    `workShifts = workShifts :+ w`
- `hours`
  - i stället för `map` går det lika bra med **`for yield`**, men då behövs parenteser t.ex.:
    `(for (w <- workShifts) yield w.hours).sum`
    eller tilldelning till extra **`val`** t.ex.:
    `{ val hs = for (w <- workShifts) yield w.hours; hs.sum }`
    eller en imperativ lösning, t.ex.:
    `{ var sum = 0; workshifts.foreach(sum += _.hours); sum }`
- `days`
  - i stället för `length` går det lika bra med `size` på en sekvens
  - i stället för `map` går det lika bra med **`for yield`**, t.ex.:
    `(for (w <- workShifts) yield w.date).distinct.size`
  - ok med `toSet.size` men `toSet.length` ger -1 i avdrag
- `toName`
  - betyder samma sak som detta (tydligare?) uttryck:
    `line.split(',').apply(0)`
  - i stället för `split` går det lika bra att använda t.ex.:
    `line.takeWhile(_ != ',')`
  - användning av `split` med strängargument `","` fungerar i detta fall lika bra (men man ska vara försiktig med strängargument till `split` då strängen tolkas som ett `java.util.regex.Pattern` där vissa tecken, t.ex. `[].*?\|`, har speciella betydelser)
- `toWorkShift`
  - det finns många lösningar; gör avdrag ungefär efter motsvarande funktionalitet som inte fungerar beroende på hur allvarligt felet är:
    - uppdelning av `line` med kommaseparator, 2p
    - uppdelning av arbetspass med streckseparator, 2p
    - omvandling till heltal av början och slut, 2p
    - indexeringar i motsvarande strängsekvenser `xs` och `hrs` eller annan logik som åstadkommer motsvarande, 3p
    - konstruktion av `WorkShift` 1p
  - användning av `split` med strängargument `"-"` fungerar i detta fall lika bra (men man ska vara försiktig med strängargument till `split` då strängen tolkas som ett `java.util.regex.Pattern` där vissa tecken, t.ex. `[].*?\|`, har speciella betydelser)
- `apply`
  - det finns många lösningar; gör avdrag ungefär efter motsvarande funktionalitet som inte fungerar beroende på hur allvarligt felet är:
    - inläsning (uttrycket är givet i uppgiften) motsvarar 1p
    - att ta fram alla namn ur inläst strängsekvens motsvarar 3p
    - att skapa nyckel-värde-tabellen inklusive konstruktion av tomma rapporter motsvarar 5p
    - att behandla varje indatarad och registrera rätt saker i rätt rapport motsvarar 6p inklusive att skapa oföränderlig resultattabell vilket motsvarar 1p av dessa 6p
    - att returnera referens till tabell av rätt typ motsvarar 1p
  - en annan variant som fungerar lika bra: skapa lokal förändringsbar tabell och lägg till efterhand om namn finns, annars skapa mappning från namn till ny rapport, och avsluta med `.toMap` för att returnera en oföränderlig tabell.