Kontrollskrivning
EDAA45 Programmering, grundkurs

2024-10-29, 14:00-19:00

Hjälpmedel: Snabbreferens för Scala & Java.

Instruktioner

Du får en lapp med en **identitetskod** som du ska skriva här: _____
Identitetskoden ska även skrivas på omslaget och på alla inlämnade blad. Ha legitimation redo.
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. Skriv endast på ena sidan av varje blad. Du får inte skriva med rödfärgad penna. Du ska inte lämna in kladdpapper eller anteckningar som inte ingår i dina svar. Du ska skriva din identitetskod 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, vänta under tystnad tills den individuella delen är över. Du får inte lämna lokalen. När tiden är ute samlas skrivningarna in.
- *Moment 2*, ca 1 h 15 min: **Kamratbedömning**. Du får utdelat en bedömningsmall som du läser igenom noga. Efter ett tag får du en annan persons skrivning som du poängsätter enligt anvisningarna i bedömningsmallen. Du blir också tilldelad en eller två andra studenter som diskussionspartner och ni ska hjälpa varandra att göra så bra bedömningar som möjligt. När tiden är slut samlas alla skrivningarna in.
- *Moment 3*, ca 0,5 h: **Kontrollera bedömningen**. 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.
- 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. Tolka uttryck. Totalt max 10 p.

Följande kod finns i en ensam `.scala`-fil i aktuell katalog och kompileras utan kompileringsfel när du startar REPL med kommandot `scala repl .`

```
class Mutant(private var dna: Array[Char] = Array()):
  private var sista = 'a'
  val första = 'g'

  def reset(): Mutant =
    dna = Array(första, 'u', 'r', 'k', sista)
    this

  def update(c: Char): Char =
    dna(dna.length - 1) = c
    dna = dna.reverse
    dna(dna.length - 1)

  def show = dna.mkString
end Mutant

object OuterSpace:
  val alien1 = new Mutant
  val alien2 = Mutant("tomat".toArray)
  var alien3 = alien1
end OuterSpace
```

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.

```
val u0 = 42.0
val u1 = OuterSpace.alien1.sista
val u2 = OuterSpace.alien1.reset().update('x')
val u3 = OuterSpace.alien3.show.take(3)
val u4 = {import OuterSpace.*; alien3 = Mutant(); alien3.update('y')}
val u5 = OuterSpace.alien2.reset().show.headOption
```

	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. Skriva kod. Totalt max 40 p.

Bakgrund, huvudprogram, indata, utdata

Du ska göra klart ett program som skriver ut information om vilka laborationer som studenter klarat av i sin första programmeringskurs. Följande fil med namnet `students.csv` utgör indata till programmet.

```
förnamn,efternamn,id,labbar
Oddput,Clementin,od3928cl-s,-+++
Gusten,Grodslukare,gu7612gr-s,****
Nelly,Raps,na4636fl-s,****
Sammy,Namne,sa2182na-s,
Kim,Kardasjian,ki8971ka-s
Sammy,Namne,sa2183na-s,*****
Kringla,Två Efternamn,kr8456tv-s,**+-
Anna,Panna,an1324pa-s,*****
Svenne,Banansson,sv9876ba-s,*****+
Päron,Banansson,pä4321fö-s,****
```

Filen ovan innehåller kolumner som är separerade med kommatecken med följande innebörd:

- Första raden innehåller kolumnrubriker.
- Efterföljande rader innehåller data för varje student i 3 eller 4 kolumner med förnamn, efternamn, id, samt en eventuell avslutande kolumn med noll eller flera labbmarkeringar för varje laboration. Det finns en laboration per läsvecka och totalt 7 laborationer.
- Labbmarkeringar med tecknen `*+-` representerar i tur och ordning *godkänd*, *kompletteras*, *frånvaro*. Alla andra tecken eller inga tecken representerar saknad markering.

Följande huvudprogram är givet:

```
1  @main def countApproved(studentFile: String, toWeek: Int): Unit =
2    import Register.*
3    val spaces: String = " " * (NbrOfLabs - "MARK".length)
4    println(s"ID         G MARK$spaces NAME")
5    val reg: Register = fromFile(studentFile)
6    val sorted = reg.students.sortBy(_.count(Mark.Approved)).reverse
7    for s <- sorted do println(s.showStudent())
8    val n =  reg.countAllApproved(toWeek)
9    println(s"\n$n studenter är godkända på allt t.o.m läsvecka $toWeek")
```

Följande utskrift ska ske vid körning av huvudprogrammet med `scala run . -- students.csv 4`

```
ID         G MARK    NAME
an1324pa-s 5 *****.. Panna, Anna
sa2183na-s 5 *****.. Namne, Sammy
pä4321fö-s 4 ****... Banansson, Päron
na4636fl-s 4 ****... Raps, Nelly
gu7612gr-s 4 ****... Grodslukare, Gusten
sv9876ba-s 3 ****+.. Banansson, Svenne
kr8456tv-s 2 **+-... Två Efternamn, Kringla
od3928cl-s 1 -+++... Clementin, Oddput
ki8971ka-s 0 ....... Kardasjian, Kim
sa2182na-s 0 ....... Namne, Sammy

5 studenter är godkända på allt t.o.m läsvecka 4
```

Kolumnen `G` ska visa totalt antal godkända laborationer för varje student. Raderna ska vara sorterade i fallande ordning enligt kolumnen `G`. Den avslutande raden ska visa totalt antalet studenter som är godkända på alla laborationer fram till och med vecka `toWeek`.

Uppgifter

Du ska implementera delarna markerade med `???` enligt efterföljande krav och tips. Poängvärdet anges i kommentarer i koden efter `???`. Varje inlämnad lösning ska börja med den implementerade medlemmens huvud så som det står på raden med `???` fram till och med `=`. Du ska inte skriva av hela klassen, objektet eller enumerationen utan bara lämna in de medlemmar som saknar implementation.

Uppgift `Mark`

Den uppräknade datatypen `Mark` beskriver en labbmarkering som i tur och ordning innebär *saknad* markering, *frånvaro* på labben, labb som ska *kompletteras*, och *godkänd* labb. Metoden `toChar` ger en teckenrepresentation av markeringen.

```
1  enum Mark:
2    case Empty, Absent, Incomplete, Approved
3    def toChar: Char = Mark.chars(ordinal)
4
5  object Mark:
6    val chars: Seq[Char] = Vector('.', '-', '+', '*')
7
8    def fromChar(c: Char): Mark = ??? //6p
9  end Mark
```

Krav och tips:

- `fromChar`:
  - Givet ett tecken ges motsvarande uppräknat värde i **`enum`** `Mark`, i tur och ordning enligt `chars`.
  - Om tecknet saknas i `chars` så ska `Mark.Empty` ges.
  - Du har nytta av samlingsmetoden `indexOf(v)` som ger `-1` om linjärsökningen ej hittar `v`.
  - Du har nytta av kompanjonsmedlemmen `values` som är tillgänglig för alla **`enum`**.

Uppgift `Student`

Case-klassen `Student` beskriver en student med namnet `name` som är ett strängpar med förnamn följt av efternamn, en unik `id`, samt en sekvens med labbmarkeringar.

```
1  case class Student(name: (String, String), id: String, marks: Seq[Mark]):
2    def isApproved(labIndex: Int): Boolean = ??? //4p
3
4    def isAllApproved(toWeek: Int): Boolean = ??? //6p
5
6    def count(m: Mark): Int = ??? //4p
7
8    def showName(lastNameFirst: Boolean = true): String = ??? //3p
9
10   def showStudent(lastNameFirst: Boolean = true): String =
11     val ms = marks.map(_.toChar).mkString
12     s"$id ${count(Mark.Approved)} $ms ${showName(lastNameFirst)}"
```

Krav och tips:

- `isApproved`:
  - Ska ge sant endast om tecknet i `marks` med index `labIndex` motsvarar godkänd laboration.
  - Ska ge falskt (alltså inte orsaka undantag) om `labIndex` är utanför tillgängliga index i `marks`.
- `isAllApproved`:
  - Ska ge sant om alla labbmarkeringar i `marks` från vecka 1 fram till och med vecka `toWeek` är godkända, annars falskt.

- `count`:
  - Ska ge antalet laborationsmarkeringar i `marks` som motsvarar `m`.
- `showName`:
  - Ska ge en sträng som visar namnet.
  - Om `lastNameFirst` är sann så ska efternamnet visas först följt av ett komma och ett blanktecken innan förnamnet, annars så ska förnamnet visas först följt av ett blanktecken och efternamnet.

Uppgift `Register`

```
1  case class Register(students: Vector[Student]):
2    def countAllApproved(toWeek: Int): Int = students.count(_.isAllApproved(toWeek))
3
4  object Register:
5    val NbrOfLabs = 7
6
7    def fromStringToMarks(s: String): Seq[Mark] = ??? //5p
8
9    def fromLineToStudent(line: String, delim: String): Student = ??? //12p
10
11   def fromFile(fileName: String): Register =
12     val lines = scala.io.Source.fromFile(fileName, "UTF-8").getLines.toVector
13     Register(lines.drop(1).map(line => fromLineToStudent(line, ",")))
14 end Register
```

Krav och tips:

- `fromStringToMarks`:
  - Ska ge en sekvens av längd `NbrOfLabs` oavsett längden på `s`.
  - Om `s` är kortare än `NbrOfLabs` så ska resultatet fyllas ut med tomma markeringar.
  - Om `s` är längre än `NbrOfLabs` så ska överskjutande markeringar ignoreras.
- `fromLineToStudent`:
  - Ska skapa en studentinstans givet en indata-rad enligt beskrivningen på sidan 4.
  - Om en eller fler markeringar saknas ska tomma markeringar ges för motsvarande labb. Notera hur saknade markeringar hanteras i körningen av huvudprogrammet på sidan 4.
  - Du har nytta av strängmetoden `s.split(d)` som ger en `Array[String]` med de inre strängdelar som finns i `s` mellan varje förekomst av delsträngen `d`. Exempel:

    ```
    scala> val xs = "hejsansvejsjsanjs".split("js")
    val xs: Array[String] = Array(he, ansve, "", an)
    ```

  - Ska kasta ett undantag av typen `IllegalArgumentException` med felmeddelandet *felaktigt antal kolumner* om `line` inte har 3 eller 4 kolumner (se indata på sidan 4).
  - Ska kasta ett undantag av typen `IllegalArgumentException` med felmeddelandet *tomma kolumner* om någon av kolumn 1 till 3 i `line` är tom (se indata på sidan 4).
  - Du har nytta av metoden `require`. Exempel:

    ```
    scala> :type require(true, "pang!")
    Unit

    scala> require(true, "pang!")

    scala> require(false, "pang!")
    java.lang.IllegalArgumentException: requirement failed: pang!
    ```