Valfri tentamen
EDAB05 Programmering, grundkurs

2026-01-07, 8:00-13:00

Instruktioner

- Skriv din anonymkod + personlig identifierare här: _____
  Om du skriver icke-anonymt ange personnummer + namn i stället.
- Tillåtet hjälpmedel: Snabbreferens för Scala tryckt av institutionen. (Egna utskrifter tillåts ej då de är svåra att kontrollera av tentavakt.)
- **Del A** (uppgift 1) ska besvaras genom att fylla i en tabell i *detta häfte*.
- **Del B** innehåller uppgift 2, 3, ... med svar i form av programkod som du ska skriva på *separata vita papper*. Skriv bara på *ena* sidan av varje inlämnat blad. Skriv din anonymkod + personlig identifierare (eller personnummer + namn om du skriver icke-anonymt) *överst* på *varje* inlämnat blad. Det ska tydligt framgå vilken (del)uppgift du löser.
- Detta häftet ska lämnas in tillsammans med ifyllt omslag och svaren på uppgifterna i del B.
- Innan du går måste du lämna in detta häfte och ifyllt omslag, även om du inte löst några uppgifter. Du får inte lämna salen innan du lämnat in. Du får tidigast lämna in 1 timme efter start.
- Du ska *inte* tentera om du ej är tentamensberättigad. För att vara tentamensberättigad ska du före tentamen vara godkänd på alla obligatoriska moment (laborationer, projekt och muntligt prov).

Upplysningar

- Om du tenterar utan att vara tentamensberättigad annulleras din skrivning utan att bedömas. För att undvika att någon skrivning annulleras av misstag kommer alla som, enligt institutionens noteringar, tenterat utan att vara tentamensberättigade att kontaktas via epost. Felaktigheter i institutionens noteringar kan därefter påtalas inom en månad.
- Tentamen är valfri för tentaberättigade och kan resultera i överbetygen 4 eller 5 om du uppfyller kraven för överbetyg, enligt nedan. Om du är tentaberättigad får du minst betyg 3 oavsett resultat på denna tentamen.
- Tentamen kan maximalt ge 100p, varav del A omfattar 20p och del B omfattar 80p.
- Poänggränser för överbetyg:
  - För överbetyg krävs minst 10p på del A, samt totalt minst 67p för betyg 4 och 83p för betyg 5.
  - Om det blir uppenbart under pågående bedömning att lösningarna inte kommer kunna uppfylla lägsta gräns för överbetyg så kan vidare bedömning av resterande uppgifter komma att avbrytas.
  - Poängangivelser i uppgifterna är preliminära och kan komma att justeras.
- Lösningar och bedömningsriktlinjer läggs ut på kursens hemsida efter tentamen.
- Resultatet läggs in i Ladok när bedömningen är klar. Visning sker på datavetenskaps expedition.

Del A. Uppgift 1. Evaluera uttryck. Totalt max 20p.

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

```
1  sealed trait Tomte:
2    def secret: Set[Julgodis] = Set()
3
4  enum Julgodis:
5    case Gurka, Choklad, Chilli
6
7  case class Troll(override val secret: Set[Julgodis]) extends Tomte:
8    var hack = secret
9
10 object Troll extends Tomte:
11   export Julgodis.*
12   override def secret = Set[Julgodis](Chilli, Chilli, Chilli)
13
14   def apply(styrka: Int = Julafton.tomat): Troll =
15     new Troll((this.secret.toSeq.sortBy(_.ordinal).reverse.drop(1)
16       ++ Seq.fill(styrka)(Chilli)).toSet)
17 end Troll
18
19 object Julafton:
20   private var n = 3
21   var tomtar: Map[Int, Tomte] =
22     (for i <- 1 until n yield i -> Troll(styrka = n + 39)).toMap
23   var a = Troll(1)
24   def tomat = n
25   def godis(): Unit = a.hack = a.hack ++ Julgodis.values.toSet
26   def xxx: Option[Tomte] = if tomtar == null then null else tomtar.get(42)
```

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`, `u2`, ... , 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 = 42.0
2  val u1 = Julafton.tomtar.apply(0).secret.size
3  val u2 = Troll().hack.contains(Troll.Chilli)
4  val u3 = { Julafton.n = 42; Troll.apply().secret.size }
5  val u4 = ((Julafton.tomat + math.Pi) / 3).toInt
6  val u5 = Julafton.tomtar(2).secret.size
7  val u6 = { Julafton.godis(); Julafton.a.hack.size }
8  val u7 = { Julafton.a.secret = Set(); Julafton.a.hack.size; () }
9  val u8 = Julafton.xxx match { case a if a.get == Troll() => 42 case b => 42}
10 val u9 = { Julafton.tomtar = null; Option(Julafton.xxx).getOrElse(None) }
11 val u10 = Julafton.tomtar.size == 0
```

	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`				
`u6`				
`u7`				
`u8`				
`u9`				
`u10`				

Del B. Uppgift 2–4. Totalt max 80p.

Karaoke-app i terminalen

Ditt har fått i uppdrag att hjälpa D-sektionens aktivitetsutskott att skapa en terminalapp som visar karaoke-sångtexter i Linux-terminalen. Karaoke (från japanskans *kara okesutora*, som betyder ”tom orkester”) innebär aktiv underhållning i form av förinspelad musik och sång till text på skärm.

Det finns givna styrtecken som möjliggör positionering och markering (eng. *highlighting*) av text på godtycklig plats i ett terminalfönster. Med hjälp av dessa ska du färdigställa en app som läser en sångtext i ett speciellt filformat och sedan visar några sångtextrader i taget. Orden i texten ska markeras i takt med musiken, så att aktiv teknolog kan sjunga med.

Huvudprogram och exempel

Vid körning av nedan huvudprogram med den episka sångtexten i textfilen `rosa.txt` som programargument, vars början återges i rutan till höger, så visas sångtextrader med markerade ord steg för steg, enligt skärmklippet nedan till höger.

```
@main def Main(path: String): Unit =
  KaraokeTerminal(nRows = 5).play:
    KaraokeTerminal.Lyrics.fromFile(path)
```

```
> scala run . -- rosa.txt
```

Takten styrs av det angivna tempot `152`, med enheten *taktslag per minut*. Då markeras mer och mer av texten i takt med detta tempo. Varje semikolon i textfilen motsvarar ett taktslag. Varje taktslag orsakar en fördröjning om `152/60.0` sekunder, innan nästa markering sker av tecknen fram till nästa taktslag. Notera att semikolon ej ska visas i terminalen, utan enbart orsaka korrekt fördröjning i enlighet med det angivna tempot. Sången börjar med ett intro som ges av blanktecken och 6 taktslag – här alltså två vackra tretakter (vals). Med `nRows = 5` visas endast 5 textrader i taget och nya rader scrollas nedifrån och upp. När texten är slut skrivs en tom rad ut och programmet avslutas.

Början av `rosa.txt`, ”Rosa på bal” (Evert Taube)

```
tempo=152
 ; ; ; ; ; ;
Tänk; att; jag; dan;sar; med; An;;der;sson;;
Lilla; jag;; lilla; jag;;
med Fritiof; An;;der;sson;;;
Tänk; att; bli; upp;bjud;en; av;; en; sån;;
popu;lär;;; per;;;son!;;;
 ; ; ;
Tänk; vil;;ket un;der;bart; liv; de;; ni för;;;
Säj; mej; hur; känns; det; att; va;ra;; charmör;;;
Sjö;man; och; cow;boy; mu;si;ker; art;ist;;;
Det; kan; väl; ald;rig; bli; trist?;;;
 ; ; ;
Nej; al;drig; trist; frö;;ken Ro;;;sa;;;
har; man; som; er; ka;;valjer;;;
 ; ; ;
Vart; jag; än; stä;ller; min; ko;;;sa;;;
al;drig; för;glö;mmer; jag; er;;;
 ; ; ;
Ni; är;; en sång;mö;; från He;li;;kons; berg;;;
Oh,; frö;ken; Rosa;, er; li;nje; er; färg;;;
skul;dran,; pro;fil;en; med; lock;ar;nas; krans,;;;
ö;go;nens; vaaaaaa;;r;ma glans.;;;
```

```
Tänk att jag dansar med Andersson
Lilla jag lilla jag
med Fritiof Andersson
Tänk att bli uppbjuden av en sån
```

Du ska göra klart de påbörjade implementationerna av klasserna i efterföljande uppgifter enligt givna krav och tips. Klassen `KaraokeTerminal` ärver klassen `Terminal`, som i sin har en `MutableMatrix`. Sångtexter läses från fil och tolkas med hjälp av klassen `Lyrics` och visas med metoden `play` i klassen `KaraokeTerminal`.

Uppgift 2. `MutableMatrix`. (10p)

`MutableMatrix` är en generisk, förändringsbar matris som erbjuder indexering och uppdatering på plats.

```
class MutableMatrix[T](val nRows: Int, val nCols: Int, val defaultValue: T):
  require(nRows > 0 && nCols > 0, s"nRows and nCols must be non-zero")

  import collection.mutable.ArrayBuffer

  private val matrix: ArrayBuffer[ArrayBuffer[T]] =
    ArrayBuffer.fill(nRows):
      ArrayBuffer.fill(nCols)(defaultValue)

  def apply(row: Int, col: Int): Option[T] = ??? //6p

  def update(row: Int, col: Int, value: T): Unit = ??? //4p

  def reset(row: Int, col: Int): Unit = update(row, col, defaultValue)

end MutableMatrix
```

Du ska implementera de saknade delarna ovan enligt dessa krav och tips:

- Metoden `apply` ska ge en `Option` som innehåller ett värde som finns på raden `row` i kolumnen `col` om både `row` och `col` är giltiga index, annars ska en tom `Option` ges.
- Metoden `update` ska uppdatera värdet på raden `row` och kolumnen `col` om både `row` och `col` är giltiga index, annars ska inget hända.

Uppgift 3. `Terminal`. (33p)

`Terminal` på sidan 7 hanterar utskrift av positionerad och markerad text i terminalen med speciella styrtecken, enligt vad som är standard i Linuxterminalen. Dessa styrtecken återfinns som teckenvärden i kompanjonsobjektet `Terminal`, med tillhörande förklaring i resp. kommentar. Varje styrtecken föregås av det speciella tecknet `EscapeCode`. Metoden `moveCursor` gör så att nästa utskrift sker på angiven rad och kolumn och hanterar att index i `buffer` är noll-baserade medan terminalpositioner via styrtecken är ett-baserade. Metoden `hideCursor` gömmer terminalens blinkande markör, så att den inte förvirrar visningen av sångtextrader under pågående stegvis markering. Metoden `showCursor` återställer terminalens blinkande markör.

Du ska implementera de saknade delarna och beakta dessa krav och tips:

- `clear` ska rensa terminalfönstret genom att blankställa alla `buffer`-värden och skriva ut styrtecknen `ClearScreen` och `MoveHome` (terminalrensning, efterföljande utskrift överst till vänster).
- `getChar` ska ge ett teckenvärdet som eventuellt finns på raden `row` och kolumnen `col` i `buffer`. Om värde saknas ska teckenattributet i `CharData.Blank` ges.
- Om `row` och `col` är giltiga index så ska `putChar` uppdatera `buffer` med en `CharData`-instans på raden `row` och kolumnen `col` och även skriva ut tecknet med hjälp av `printCharDataAt`. Om `row` eller `col` ej är giltiga index i `buffer` så ska inget hända.
- `get` ska ge ska ge de tecken som finns på rad `row` från kolumnen `fromCol` till (men inte med) kolumnen `untilCol`. Du har nytta av metoden `getChar`.
- `put` ska uppdatera `buffer` och skriva ut tecknen i strängen `text` på rad `row` med start från kolumn `col`. Om `isHighlight` är **`true`** så ska texten vara markerad annars inte. Du har nytta av `putChar`.
- `highlight` ska ånyo skriva ut tecknet på raden `row` och kolumnen `col` i markerat läge, oavsett om det innan utskrift var markerat eller inte. Du har nytta av `putChar` och `getChar`.
- `scroll` ska förflytta alla rader i `buffer` upp ett steg och fylla på med en blankrad underifrån och även visa motsvarande uppflyttade rader i terminalen. Du har nytta av metoderna `update` och `reset` i `MutableMatrix`, samt `printCharDataAt` och `printBlankAt`.

```
class Terminal(val nRows: Int, val nCols: Int):
  import Terminal.*

  private val buffer = MutableMatrix(nRows, nCols, CharData.Blank)

  private def printBlankAt(row: Int, col: Int): Unit =
    moveCursor(row, col)
    print(' ')

  private def printCharDataAt(row: Int, col: Int): Unit =
    moveCursor(row, col)
    val cd = buffer(row, col).get
    if !cd.isHighlight then print(cd.char)
    else print(s"${InvertBackground}${cd.char}${Reset}")

  def clear(): Unit = ??? //5p

  def getChar(row: Int, col: Int): Char = ??? //3p

  def putChar(row: Int, col: Int, c: Char, isHighlight: Boolean): Unit = ??? //7p

  def get(row: Int, fromCol: Int = 0, untilCol: Int = nCols): String = ??? //3p

  def put(text: String, row: Int, col: Int = 0, isHighlight: Boolean): Unit = ??? //4p

  def highlight(row: Int, col: Int): Unit = ??? //2p

  def scroll(): Unit = ??? //9p

object Terminal:
  case class CharData(char: Char, isHighlight: Boolean = false)

  object CharData:
    val Blank = CharData(' ')

  val EscapeCode        = 27.toChar           // specialtecken som föregår styrkoder
  val ClearScreen       = s"${EscapeCode}[2J" // terminalen rensas
  val MoveHome          = s"${EscapeCode}[H"  // nästa utskrift sker överst till vänster
  val InvertBackground  = s"${EscapeCode}[7m" // påbörjar markerad text
  val Reset             = s"${EscapeCode}[0m" // avslutar markerad text

  def width(default: Int = 60) = // ger skärmbredd i antal tecken
    sys.env.get("COLUMNS").flatMap(_.toIntOption).getOrElse(default)

  def moveCursor(row: Int, col: Int): Unit = // nästa utskrift sker vid (row, pos)
    print(s"${EscapeCode}[${row + 1};${col + 1}H")

  def hideCursor(): Unit = print(s"${EscapeCode}[?25l")  // gömmer blinkande markör

  def showCursor(): Unit = print(s"${EscapeCode}[?25h")  // visar blinkande markör
end Terminal
```

Uppgift 4. `KaraokeTerminal`. (37p)

`KaraokeTerminal` visar en sångtext och markerar det som ska sjungas i takt med musiken. Endast `nRows` rader visas i taget och texten scrollas när en rad är helt markerad, se exempel på sidan 4. `Tempo` representerar takten i musiken. Klassen `Lyrics` representerar en sång med tempo och sångtextrader som innehåller tecken som representerar taktslag.

```
1  import scala.compiletime.ops.double
2
3  object KaraokeTerminal:
4    case class Tempo(beatsPerMin: Int):
5      require(beatsPerMin > 0, "beatsPerMin must be more than zero")
6      val secondsPerBeat: Double = 60.0 / beatsPerMin
7      def waitBeat(): Unit = Tempo.waitSeconds(secondsPerBeat)
8
9    object Tempo:
10     def waitSeconds(seconds: Double): Unit = Thread.sleep((seconds * 1000).round)
11     val Default = Tempo(120)
12     val SettingPrefix = "tempo="
13     val Separator = ';'
14
15     def fromString(s: String): Option[Tempo] =
16       if s.startsWith(Tempo.SettingPrefix) then
17         val bpm = s.stripPrefix(Tempo.SettingPrefix)
18           .toIntOption
19           .getOrElse(Tempo.Default.beatsPerMin)
20         if bpm > 0 then Some(Tempo(bpm)) else None
21       else None
22   end Tempo
23
24   case class Lyrics(tempo: Tempo, lines: Vector[String]):
25     def showLine(i: Int): String = lines(i).filterNot(_ == Tempo.Separator)
26
27   object Lyrics:
28     private def loadLines(path: String): Vector[String] =
29       val s = io.Source.fromFile(path, enc = "UTF-8")
30       try s.getLines().toVector finally s.close()
31
32     def fromFile(path: String): Lyrics = ??? //7p
33   end Lyrics
34
35 class KaraokeTerminal(override val nRows: Int)
36 extends Terminal(nRows, Terminal.width()):
37   require(nRows >= 3, "nRows must be at least 3")
38   import KaraokeTerminal.*
39
40   def playLine(row: Int, lineWithBeats: String, tempo: Tempo): Unit = ??? //8p
41
42   def play(lyrics: Lyrics): Unit = ??? //22p
43 end KaraokeTerminal
```

Du ska implementera de saknade delarna ovan och beakta efterföljande krav och tips:

- Tecknet `Tempo.Separator` är ett semikolon som representerar ett taktslag.
- Tecken som representerar taktslag ska inte visas i terminalen.
- `fromFile` ska läsa en textfil med hjälp av `loadLines` och konstruera en instans av klassen `Lyrics`. Du har nytta av `fromString` som tolkar en eventuell inledande temporad som börjar med strängen `Tempo.SettingPrefix`, se exempel på sidan 4. Om temporad saknas ska `Tempo.Default` användas. Sångtextraderna i `lines` ska ej ha temporad.
- `playLine` ska för varje tecken i `lineWithBeats` antingen vänta (om tecknet representerar ett taktslag) eller markera tecknet i motsvarande kolumn. Denna metod förutsätter att sångtextraden redan visas i Terminalen på raden med index `row` i `buffer`.
- `play` ska fungera enligt följande:
  - Först ska skärmen rensas och den blinkande markören ska gömmas.
  - Sedan ska terminalen *initialiseras* genom att, med början på mittersta raden, visa de första sångtextraderna som får plats fram till sista raden. I exempel på sidan 4 visar terminalen med `nRows == 5` rader. Initialt innehåller den två tomma rader och mittenraden (radindex 2) består av 6 blanktecken eftersom första raden i sångtexten innehåller 6 blanktecken. Därefter består den initialiserade terminalen av raderna `Tänk att jag dansar med Andersson` på rad med index 3 samt `Lilla jag lilla jag` på rad med index 4. Exempelskärmbilden på sidan 4 visar läget efter att initialisering och två steg i nedan repetition hunnit göras.
  - Repetera så länge det finns ännu ej visade rader i `lyrics`:
    - Spela upp aktuell rad genom stegvis markering i takt med musiken.
    - Skrolla.
    - Beräkna ny bottenrad som ska bestå av nästa kommande textrad utan taktslagstecken. Om det inte finns fler rader i lyrics så ska bottenraden vara tom.
    - Visa nya bottenraden.
  - Efter att ovan repetition är klar så ska den blinkande markören visas igen och en tom rad ska skrivas ut med `println`.
  - Du har nytta av metoderna `playLine`, `scroll`, `hideCursor`, `showCursor`, `put`, `showLine`.