LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Kontrollskrivning 2, Programmeringsteknik för D/C

2011–11–14, 13.15–16.15

Anvisningar: fyll i omslaget fullständigt, även lösta uppgifter och antalet inlämnade blad. Skriv dina lösningar på rutpapper, bara på en sida av varje papper. Lämna bara in dina lösningar och omslaget, inte skrivningen och inte några kladdpapper. Tillåtet hjälpmedel: Java-snabbreferens.

Preliminärt ger uppgifterna $15 + 15 + 10 = 40$ poäng.

Jag meddelar på kurshemsidan när rättningen är klar och sätter upp en lista med poäng på anslagstavlan. Senare blir det visning av skrivningen.

---

1. I en programmeringskurs har man samlat in uppgifter om kursdeltagarnas förkunskaper. Förkunskaperna har graderats från 1 (har aldrig programmerat) till 4 (har skrivit många och stora program).

   När kursen är avslutad vill man undersöka om förkunskaperna har någon inverkan på slutbetyget (som kan vara 0, 3, 4 eller 5). Man redovisar detta i en tabell med följande utseende:

   ```
       0   3   4   5
     ---------------
   1 |  6   8   2   4
   2 |  3   3   2   2
   3 |  4   8   8  10
   4 |  4  10  11  15
   ```

   Här ser man till exempel att 30 studenter hade förkunskaperna ”3”. Av dessa fick 4 betyget 0, 8 betyget 3, 8 betyget 4 och 10 betyget 5.

   Studenternas resultat finns i en fil med namnet `result.txt`. Varje rad i filen innehåller förkunskaper (1–4) och betyg (0, 3, 4 eller 5) för en student. Raderna är inte sorterade på något sätt. Början av filen kan ha följande utseende:

   ```
   2 3
   4 5
   2 0
   ...
   ```

   Skriv ett program som läser in filen `result.txt` och skriver ut betygsfördelningen enligt exemplet. Anvisningar:

   - Det står i snabbreferensen hur man läser från en fil.
   - Om filen inte kan öppnas ska man få en felutskrift, och så ska programmet avslutas.
   - Registreringen av de 16 antalen görs lämpligen i en matris.
   - Du ska bara skriva ut det ”inre” av tabellen (6 8 2 4 / 3 3 ...).
   - Du behöver inte skriva talen i raka kolumner.

2. I ett kortspel beskrivs korten av en klass `Card` (samma som använts i laboration 7):

```
/** konstanter för färgerna */
static final int SPADES = ...;
static final int HEARTS = SPADES + 1;
static final int DIAMONDS = SPADES + 2;
static final int CLUBS = SPADES + 3;

/** Skapar ett spelkort med färgen suit (SPADES, HEARTS, DIAMONDS, CLUBS)
    och valören rank (1-13) */
Card(int suit, int rank);

/** Tar reda på färgen */
int getSuit();

/** Tar reda på valören */
int getRank();
```

I spelet delar man ut kort till spelarna. En spelares kort beskrivs av klassen `Hand`:

```
public class Hand {
    private Card[] cards;
    private int n;

    /** Skapar en hand utan några kort */
    public Hand() {
        cards = new Card[13];
        n = 0;
    }

    /** Lägger kortet c sist i handen */
    public void addCard(Card c) {
        cards[n] = c;
        n++;
    }

    /** Tar reda på antalet kort i handen */
    public int getNbrCards() {
        // ... implementera
    }

    /** Tar reda på det bästa kortet (null om det inte finns några kort i
        handen). Spader är den bästa färgen, sedan hjärter följt av ruter
        och klöver. Ett kort av en viss färg är bättre än alla kort av
        sämre färger (till exempel är spader 2 bättre än hjärter kung).
        Inom en färg är kortet med valören 13 bäst, sedan 12, ..., 2, 1. */
    public Card getBestCard() {
        // ... implementera
    }
}
```

Implementera metoderna `getNbrCards` och `getBestCard`.

3. I ett studentregister lagras uppgifter om studenternas tentor. En student beskrivs av följande klass (man lagrar många fler uppgifter om studenterna, till exempel namn, adress och personnummer, men det bryr vi oss inte om i denna uppgift):

```
/** Skapar en student utan några tentor */
Student();

/** Lägger till en tentamen i kursen course. Tentamen gjordes datumet date
    och resultatet var result (betyget 0, 3, 4 eller 5) */
void addExam(Course course, String date, int result);

/** Tar reda på hur många tentor som studenten har gjort */
int getNbrExams();

/** Tar reda på hur många poäng som studenten har klarat totalt. En kurs är
    avklarad om betyget är 3, 4 eller 5. Du får förutsätta att man bara
    klarar en kurs en gång */
double getTotalCredits();
```

Implementera klassen. Du *ska* använda en `ArrayList` med `Exam`-objekt. Klassen `Exam`, som beskriver en tenta, är färdigskriven:

```
/** Skapar en tentamen i kursen course med datumet date och resultatet
    result */
Exam(Course course, String date, int result);

/** Tar reda på kursen */
Course getCourse();

/** Tar reda på resultatet */
int getResult();
```

Klassen `Course` beskriver en kurs (också denna klass är färdigskriven):

```
/** Tar reda på kurskoden (till exempel EDA016) */
String getCode();

/** Tar reda på kursens poäng (till exempel 7.5) */
double getCredits();
```

Så här kan det se ut för en student (studenten har tenterat tre olika kurser och klarat 12,5 poäng):