LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Kontrollskrivning 2, Programmeringsteknik för D/C

2013–11–18, 13.15–16.15

Anvisningar: fyll i omslaget fullständigt, även lösta uppgifter och antalet inlämnade blad. Skriv 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 $14 + 12 + 14 = 40$ 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 ett program för bokning av undervisningssalar håller man reda på bokningarna av en viss sal under en viss dag i ett objekt av klassen `DayBookings`. Man kan bara boka hela timmar 0–23.

```
/** Skapar ett objekt som håller reda på vilka tider som salen room är
    bokade under dagen day. Från början är inga tider bokade */
DayBookings(String room, String day);

/** Tar reda på salen */
String getRoom();

/** Tar reda på dagen */
String getDay();

/** Returnerar true om salen är bokad under timmen hour, false annars */
boolean isBooked(int hour);

/** Undersöker först om salen är ledig under alla timmarna start..finish.
    I så fall bokas alla timmarna och returneras true. I annat fall
    bokas ingenting och returneras false */
boolean book(int start, int finish);
```

   Implementera klassen. Du *ska* använda en vektor `booked` med 24 `boolean`-element där `booked[hour]` är `true` om salen är bokad under timmen `hour`, `false` annars.

2. I en kurs vid en högskola måste studenterna genomföra ett ”färdighetstest”. Frågorna och svaren hanteras i ett webbaserat system. I systemet behöver man en modul som producerar statistik över hur många studenter som har svarat rätt på olika antal frågor. Statistikutskriften kan se ut så här (6 frågor, 250 studenter):

```
Antal rätt  Antal studenter
     0           16
     1           21
     2           39
     3           68
     4           54
     5           40
     6           12
```

Ett svar som en student lämnat in beskrivs av ett objekt av klassen `Answer`. Metoden `getQNbr` används inte förrän i uppgift 3, och klassen har fler metoder som inte alls används här.

`Answer`

```
/** Tar reda på frågans nummer */
int getQNbr();

/** Returnerar resultatet (true om svaret är rätt, false annars) */
boolean isCorrect();
```

En student får lämna in hur många svar som helst på frågorna, och alla svaren sparas. När studenten lämnat in ett korrekt svar på en fråga så blockeras frågan för inlämning (dvs studenten kan inte lämna in fler svar på just den frågan).

Klassen `Student` håller reda på alla svaren som en student har lämnat in. I listan med svar finns både felaktiga och korrekta svar, och svaren kan komma i vilken ordning som helst.

`Student`

```
/** Tar reda på antalet svar som studenten har lämnat in (på alla frågor,
    både felaktiga och korrekta svar räknas) */
int getNbrAnswers();

/** Returnerar svar nummer n. Det måste gälla att 0 <= n < getNbrAnswers */
Answer getAnswer(int n);
```

Färdighetstestet beskrivs av följande klass:

```
public class Test {
    private int nbrQuestions;  // antalet frågor
    private ArrayList<Student> students; // studenterna som deltar

    ... // här finns attribut, konstruktorer och metoder
    ... // som inte används i denna uppgift

    /** Skriver ut statistik över hur många studenter som har svarat rätt
        på 0, 1, 2, ... frågor (se exemplet ovan) */
    public void printStatistics() { ... }
}
```

Implementera metoden `printStatistics`.

3. Fortsättning på uppgift 2. Den (nästan) fullständiga specifikationen av klassen `Student` ser ut så här:

`Student`

```
/** Skapar en student med namnet name */
Student(String name);

/** Tar reda på studentens namn */
String getName();

/** Lägger in svaret a i listan med studentens svar, under förutsättning
    att frågan inte redan är korrekt besvarad. Om svaret lagts in i listan
    returneras true, annars false */
boolean submitAnswer(Answer a);

/** Tar reda på antalet svar som studenten har lämnat in (på alla frågor,
    både felaktiga och korrekta svar räknas) */
int getNbrAnswers();

/** Returnerar svar nummer n. Det måste gälla att 0 <= n < getNbrAnswers */
Answer getAnswer(int n);
```

Implementera klassen. Använd en `ArrayList` för att lagra svaren. Använd en annan `ArrayList` för att hålla reda på numren på de frågor som studenten har besvarat korrekt. Tips: `contains` (se snabbreferensen) är en bra metod.