LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Tentamen, Programmeringsteknik för BME, C, D, E, F, I, N och Pi (EDA011/EDA016/EDA017)

2016–08–24, 8.00–13.00

Anvisningar: Preliminärt ger uppgifterna $8 + 10 + 10 + 17 = 45$ poäng. För godkänt betyg krävs 22,5 poäng.

Tillåtet hjälpmedel: Java-snabbreferens.

Lösningsförslag kommer att finnas på kursens hemsida senast dagen efter tentamen.
Resultatet läggs in i Ladok när rättningen är klar och anslås inte på anslagstavlan.

Tentamensvillkor: Om du tenterar utan att vara tentamensberättigad annulleras din skrivning. 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 fram till nästa tentamenstillfälle då resterande skrivningar annulleras.

Uppgifterna handlar om klasser som ingår i ett program för att frivilligarbetare på en festival ska kunna anmäla vilka tider de önskar arbeta. Festivalarrangörerna lägger in ett antal arbetspass som består av tid och uppgift. En webbsida för festivalen skapas där dessa tider visas. Varje arbetare har sedan möjlighet att gå in på festivalens webbsida och anmäla sig genom att fylla i sitt namn samt kryssa i de tider och arbetsuppgifter som passar. Festivalarrangörerna kan se vilka pass som är schemalagda och som saknar tider etc. Klasserna i tentan antas användas av de klasser som hanterar webbsidan (vilka är utanför uppgiften).

I tentamen hanterar vi endast hela timmar och arbetarnas namn antas vara unika. Du kan utgå från att inga arbetspass passerar midnatt. Start- och sluttid tillhör med andra ord alltid samma dygn.

Här är ett exempel på en tidsenkät med sju möjliga arbetspass och fyra arbetare. Ett av arbetspassen är ännu inte schemalagt.

	Städa toaletter 17-19	Vakta entrén 17-20	Plocka skräp 18-19	Plocka skräp 19-20	Städa toaletter 19-21	Vakta entren 20-22	Plocka skräp 22-23
Ingert	X			X			
Jessica		X					X
Joey			X				
Kim						X	

1. Klassen `WorkPeriod` beskriver ett arbetspass.

`WorkPeriod`

```
/** Skapar att arbetspass för uppgiften task, starttiden hour
    och längden length timmar. */
WorkPeriod(String task, int hour, int length)

/** Returnerar true om arbetspasset helt eller delvis kolliderar med
    arbetspasset wp, annars false. */
boolean collidesWith(WorkPeriod wp);

/** Jämför detta arbetspass med arbetspasset wp.
    Returnerar 0 om detta objekt och wp anses lika,
    ett negativt tal om detta objekt är mindre än wp och ett
    positivt tal om detta objekt är större än wp.* /
int compareTo(WorkPeriod wp);

/** Returnerar arbetspasset som en sträng enligt följande exempel:
    "Vakta entrén 17-19" */
String toString();
```

Implementera klassen `WorkPeriod` enligt följande anvisningar:

- När två `WorkPeriod`-objekt jämförs med `compareTo` jämför man i första hand på starttid och i andra hand på uppgift. Sluttiden och längden ska helt bortses från. Ett `WorkPeriod`-objekt med starttid 17 anses alltid mindre än ett med starttid 18. Om de båda starttiderna är lika ska uppgiftsbeskrivningen avgöra ordningen (bokstavsordning).

2. Klassen `WorkPeriodList` håller reda på arbetspass i en lista.

`WorkPeriodList`

```
/** Skapar en tom lista för arbetspass. */
WorkPeriodList();

/** Lägger till ett arbetspass för uppgiften task, starttiden hour
    och längden length timmar. */
void add(String task, int hour, int length);

/** Returnerar en vektor som innehåller arbetspassen.
    Arbetspassen ska vara sorterade efter starttid i första hand
    och uppgift i andra hand. */
WorkPeriod[] toSortedArray();
```

Implementera klassen `WorkPeriodList`.

3. Klassen `Worker` håller reda på en persons namn och på vilka av tiderna personen ska arbeta.

`Worker`

```
/** Skapar en arbetare med namnet name. Vektorn times innehåller alla
    arbetspass som finns (oavsett om de är schemalagda på någon annan
    eller ej). */
Worker(String name, WorkPeriod[] times);

/** Returnerar arbetarens namn. */
String getName();

/** Markerar att arbetaren nu är schemalagd på passet
    med nummer nbr (0,1,2 ...). */
void schedule(int nbr);

/** Returnerar true om personen är schemalagd för att arbeta på passet
    med nummer nbr, annars false. */
boolean isScheduled(int nbr);

/** Returnerar true om arbetaren kan arbeta på passet med nummer nbr.
    En arbetare kan arbeta ett pass om inga av arbetarens redan schemalagda
    pass kolliderar med passet nbr. */
boolean canWork(int nbr);
```

Implementera klassen `Worker`.

Ledning:

- Använd en vektor med element av typen `boolean` för att hålla reda på vilka av arbetspassen personen ska arbeta. Ett `true` i första elementet betyder att arbetaren är schemalagd första arbetspasset etc. Arbetspassen numreras från 0 och uppåt. Då ett Worker-objekt skapas är alla elementen false (inga tider ännu markerade som schemalagda) och kan senare bara ändras genom metoden `schedule`.
- För samtliga metoder i `Worker` får man anta att parametern `nbr` är korrekt (d.v.s. mellan 0 och totala antalet arbetspass–1).

4. Klassen `TimePlanner` håller reda på arbetspass och de personer som ska arbeta.

   Man kan inte ändra på de möjliga arbetspassen efter det att `TimePlanner`-objektet skapats. Däremot kan man lägga till personer och låta dem välja arbetspass. I klassen finns också en metod som tar reda på vilka arbetspass som är lediga.

   (I klassen bör det egentligen finnas fler publika metoder, t.ex. för att spara informationen på fil. Men det bortser vi från i den här uppgiften.)

   `TimePlanner`

```
/** Skapar en tidsplanerare. Vektorn times innehåller alla arbetspass. */
TimePlanner(WorkPeriod[] times);

/** Lägger till en arbetare med namnet name förutsatt att
    det inte redan finns en sådan.
    Returnerar true om arbetaren lagts till, annars false. */
boolean addWorker(String name);

/** Schemalägger arbetaren med namnet name på passet nbr (0,1,2 ...)
    förutsatt att arbetaren finns, att arbetspasset är ledigt och
    att det inte kolliderar med några andra schemalagda pass för arbetaren
    (annars händer ingenting). */
void scheduleWorker(String name, int nbr);

/** Returnerar en lista med alla arbetspass som saknar schemalagd arbetare.
    Om alla arbetspass är schemalagda returneras en tom lista. */
ArrayList<WorkPeriod> availableTimes();
```

   Implementera klassen `TimePlanner`.