Tentamen, EDAA10 Programmering i Java

2011–01–10, 14.00–19.00

Anvisningar: Denna tentamen består av 5 uppgifter. Preliminärt ger uppgifterna $4 + 13 + 14 + 9 + 5 = 45$ poäng. För godkänt betyg krävs 20 poäng.

Uppgift 5 är en extrauppgift mest för dig som eftersträvar betyg 5.

Tillåtna hjälpmedel:

- Java-snabbreferens.

När rättningen är klar meddelas detta på kursens hemsida.

Tentamensvillkor: Om du tenterar utan att vara tentamensberättigad annulleras din skrivning. För att undvika att någon skrivning annulleras av misstag anslås vilka som, enligt institutionens noteringar, tenterat utan att vara tentamensberättigade. Namnen anslås (lätt krypterade) senast en vecka efter tentamen på kursens hemsida. Felaktigheter i institutionens noteringar kan därefter påtalas fram till nästa tentamenstillfälle då resterande skrivningar annulleras.

Denna tentamen innehåller ett antal klasser för fågelobservationer. Som bas har vi en *färdigskriven* klass `BirdData` som innehåller de 250 arter som brukar räknas till Sveriges häckande fågelarter (plus en och annan besökare) enligt Sveriges Ornitologiska Förenings databas. Alla dessa fåglar har ett unikt nummer.

`BirdData`, SOF:s databas över Sveriges häckande fågelarter, plus en och annan besökare:

```
static final int NBR_BIRDS = 250;   // Antalet fåglar i databasen

/** Returnerar det unika numret [1..NBR_BIRDS] för fågeln med namnet
    birdName, eller 0 om birdName ej finns i databasen */
static int getNbr(String birdName);

/** Returnerar fågelnamnet kopplat till birdNbr [1..NBR_BIRDS],
    om birdNbr==0 returneras "Övrig fågel" */
static String getName(int birdNbr);
```

Observera att allt i klassen `BirdData` är statiskt, dvs klassen används på samma sätt som standardklassen `Math`.

Uppgift 1

För datumhantering har vi två klasser `Date` och `Period`.

`Date` är *färdigskriven* och har följande specifikation:

```
/** Skapar ett datum med årtalet year, månaden month och dagen day */
Date(int year, int month, int day);

/** Returnerar datumet på formen "2011-01-10" */
String toString();

/** Returnerar true om detta datum är före datumet d, false annars */
boolean before(Date d);

/** Returnerar true om detta datum är efter datumet d, false annars */
boolean after(Date d);
```

Klassen `Period`, som beskriver en period mellan två datum, har följande specifikation:

```
/** Skapar en period mellan (och inklusive) två datum, date1 och date2.
    date1 och date2 kan komma i godtycklig ordning */
Period(Date date1, Date date2);

/** Returnerar första datumet i perioden */
Date getStartDate();

/** Returnerar sista datumet i perioden */
Date getFinishDate();

/** Returnerar true om datumet d är inom perioden, false annars */
boolean contains(Date d);
```

Implementera klassen `Period`.

Uppgift 2

För att hålla reda på fågelobservationer har vi två klasser – klassen `BirdObservation` som beskriver *en* fågelobservation och klassen `BirdObservationList` som beskriver *en lista* med fågelobservationer.

`BirdObservation` är *färdigskriven* och har följande specifikation:

```
/** Skapar en observation av en fågel med nummer birdNbr på platsen location
    vid datumet date */
BirdObservation(int birdNbr, String location, Date date);

/** Returnerar fågelnumret */
int getBirdNbr();

/** Returnerar fågelnamnet */
String getBirdName();

/** Returnerar platsen */
String getLocation();

/** Returnerar datumet */
Date getDate();

/** Returnerar en beskrivande sträng på formen:
    "Fågelnamn (fågelnr) Plats Datum"
    Exempel: "Fiskmås (36) Falsterbo 2010-02-01" */
String toString();
```

`BirdObservationList` har följande specifikation:

```
/** Skapar en tom lista */
BirdObservationList();

/** Lägger till en observation av en fågel med namnet birdName på platsen
    location vid datumet date */
void addObservation(String birdName, String location, Date date);

/** Lägger till en observation av en fågel med numret birdNbr på platsen
    location vid datumet date */
void addObservation(int birdNbr, String location, Date date);

/** Skriver ut alla observationer i listan med en observation per rad */
void printAll();

/** Returnerar en ArrayList med alla observationer som gjorts
    i perioden period */
ArrayList<BirdObservation> getAllInPeriod(Period period);

/** Returnerar en ArrayList med alla fågelobservationer som gjorts
    på platsen location */
ArrayList<BirdObservation> getAllAt(String location);
```

Implementera klassen `BirdObservationList`. Glöm inte att klassen `BirdData` som beskrevs i början på tentamen finns tillgänglig i både denna och de återstående tentamensuppgifterna!

Uppgift 3

Klassen `BirdCount` beskriver en fågelräkning, dvs hur många observationer det finns för varje fågelart.

Klassen har följande specifikation:

```
/** Skapar en fågelräkning för alla observationerna i listan list */
BirdCount(ArrayList<BirdObservation> list);

/** Returnerar antal förekomster av fågeln med numret birdNbr */
int getCount(int birdNbr);

/** Returnerar en lista av SpeciesCount-objekt med alla fågelnamn med minst
    en förekomst */
ArrayList<SpeciesCount> allObserved();

/** Returnerar en lista av SpeciesCount-objekt med alla fågelnamn med minst
    en förekomst, sorterad efter antal förekomster i avtagande ordning */
ArrayList<SpeciesCount> allObservedInCountOrder();
```

Implementera klassen.

I metoderna `allObserved` och `allObservedInCountOrder` ska du returnera en lista med `SpeciesCount`-objekt. `SpeciesCount` är en *färdigskriven* klass med följande specifikation:

```
/** Skapar ett räkneobjekt med count förekomster av fågeln med nummer
    birdNbr */
SpeciesCount(int birdNbr, int count);

/** Returnerar fågelnumret */
int getBirdNbr();

/** Returnerar antal förekomster */
int getCount();
```

Ledning: När man skapar den sorterade `SpeciesCount`-listan i metoden `allObservedInCountOrder` är det enklast att sortera in elementen på rätt plats efterhand som man skapar dem.

Uppgift 4

Skriv ett huvudprogram som testar delar av klasserna ovan genom att det:

- Läser in ett antal fågelobservationer från tangentbordet och lagrar dem. Varje fågelobservation finns på en rad och består av år, månad, dag, plats och fågelnamn åtskilda med mellanslag. Ett negativt årtal avslutar inmatningen.
- Hämtar alla fågelobservationer gjorda i Falsterbo och gör en fågelräkning på dem.
- Skriver ut namnen på de 10 vanligaste fågelnamnen i fågelräkningen (eller färre, om man inte har observerat så många). Namnen ska skrivas ut i avtagande ordning efter antal förekomster.

Uppgift 5

Två listor `obs1` och `obs2` innehåller `SpeciesCount`-objekt, sorterade efter växande `birdNbr`. En del `birdNbr` förekommer i båda listorna. Skriv en metod som returnerar en lista med nya `SpeciesCount`-objekt för dessa `birdNbr`. `count` ska vara = summan av `count` i de båda objekten. Metoden ska ha följande rubrik:

```
private static ArrayList<SpeciesCount> intersection(ArrayList<SpeciesCount> obs1,
        ArrayList<SpeciesCount> obs2);
```

Det är viktigt att metoden är effektiv!