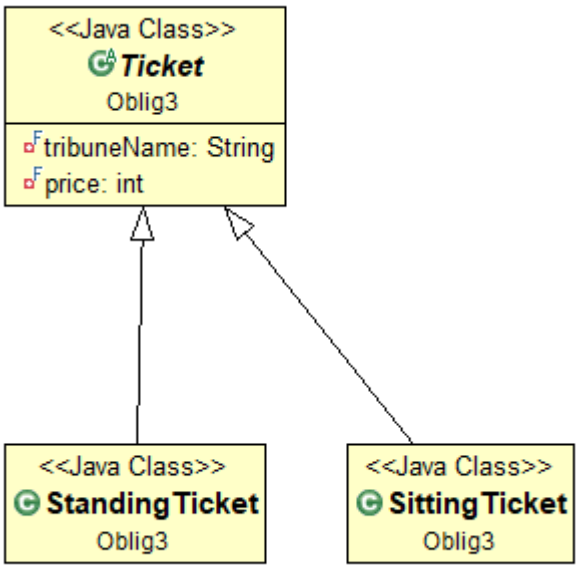
Arv og polymorfi II

Du skal jobbe med et system for salg av billetter til et sportsarrangement med ulike tribune- og billett-typer.

Vi begynner med billettene. Det selges to typer billetter, ståplassbilletter og sitteplassbilletter. Dette er modellert som vist i figur 1. Operasjoner er ikke vist i diagrammet. Disse klassene får dere ferdig programmert: Ticket.java (Se vedlegg).



Figur 1: Klassediagram for billetter

Nå går vi over til å se på tribunene. Man har ståtribuner og sittetribuner. Alle tribuner har et navn, en kapasitet og en pris. Sittetribuner har i tillegg et antall seterader. Vi antar at alle radene er like lange, slik at antall seter pr. rad er kapasiteten delt på antall rader.

For ståtribunene trenger man bare å holde rede på hvor mange plasser som er solgt, mens på sittetribunene må man vite hvor mange som er solgt på hver rad. Vi antar at plassene selges i stigende rekkefølge innenfor hver rad. I tillegg antar vi at programmet, ikke kjøperen, bestemmer hvilken rad som skal benyttes, noe du vil få bruk for i oppgave c.

Noen sittetribuner er VIP-tribuner. Der vil vi også holde rede på navnet på de som skal sitte på de ulike plassene.

Oppgave A:

Tribune-klassene skal ha følgende objekt-variabler:

Klassen Tribune:

```
private final String tribuneName;
private final int capacity;
private final int price;
```

Klassen Stand:

```
private int noSoldTickets;
```

Klassen Sit:

```
private int[] noBusy;
```

Klassen VIP:

```
private String[][] spectator; //tilskuere: antall rader * antall plasser pr rad
```

Tegn et klassediagram tilsvarende figur 1, for tribune-klassene. Les hele oppgaveteksten før du tegner, og oppdater også figuren underveis dersom du finner ut at det trengs. Du skal totalt ha 4, eventuelt 5, tribune-klasser.

I det følgende antar vi at alle objekt-variabler er private. Konstruktører og tilgangsmetoder må du programmere etter hvert som du trenger dem.

Oppgave B:

For alle tribunetyper skal vi ha metodene findNumberOfSoldTickets() og findIncome(). Inntekten er prisen multiplisert med antall solgte billetter. Programmer dette.

(Tips: Begge de nevnte metodene skal eksistere i klassen Tribune.)

Oppgave C:

Nå skal du programmere billettkjøp. Dette skal gjøres ved å lage to metoder (i tribune-klassene) som begge heter buyTicket(), og som returnerer en tabell av billett-objekter. Den første metoden tar som parameter antall billetter som ønskes, den andre tar som parameter en tabell av tekststrenger

(navnet på de som skal ha billettene). For alle tribunetyper unntatt VIP-tribuner, skal metode to gi samme resultat som metode en, mens for VIP-tribuner skal metode en returnere null (ingen får kjøpe billett på VIP-tribuner uten å oppgi navn). For alle tribunetyper gjelder at hvis man ønsker seg flere billetter enn det som kan effektueres, får man ingen. På sittetribuner begrenses dette av at alle billetter som selges i en bestilling, skal være på samme rad. På ståtribuner er det bare den totale kapasiteten, og hvor mange som hittil er solgt, som setter begrensninger.

(Tips: Begge buyTicket()-metodene skal eksistere i klassen Tribune.)

Oppgave D:

Lag et lite testprogram der du oppretter to ståtribuner, en vanlig sittetribune og en VIP-tribune. Samle tribunene i en tabell av typen Tribune[].

Kjøp billetter på alle tribunene, og skriv ut billettene.

For hver tribune skal du også skrive ut tribunenavn, kapasitet, antall solgte billetter og inntekten. (Du kan gjerne la en toString()-metode i klassen Tribune gjøre jobben.)

Oppgave E:

Vis hvordan vi sorterer Tribune-tabellen etter inntekt ved å bruke en metode i klassen Arrays.

```
package Oblig3;

/**
 * Klassen Ticket med subklasser
 * Denne blir delt ut sammen med øvingen
 */
abstract class Ticket {
  private final String tribuneName;
  private final int price;

  /**
   * Konstruktør:
   * Tribunenavn må oppgis. Ingen krav til pris.
   */
  public Ticket(String tribuneName, int price) {
    if (tribuneName == null || tribuneName.trim().equals("")) {
      throw new IllegalArgumentException("Tribunenavn må oppgis.");
    }
    this.tribuneName = tribuneName.trim();
    this.price = price;
  }

  public String getTribune() {
    return tribuneName;
  }

  public int getPris() {
    return price;
  }

  public String toString() {
    return "Tribune: "+tribuneName + " Pris: "+price;
  }
}

/**
 * STandTicket.
 */
class StandingTicket extends Ticket {
  public StandingTicket(String tribuneName, int price) {
    super(tribuneName, price);
  }
}

/**
 * Sitteplassbilletter. Rad og plass på raden må oppgis.
 */
class SittingTicket extends Ticket {
  private final int row;
  private final int place;

  public SittingTicket(String tribuneName, int price, int row, int place) {
    super(tribuneName, price);
    if (row < 0 || place < 0) {
      throw new IllegalArgumentException("Verken rad eller plass kan være negativ.\n"
                                                                + "Oppgitte verdier: " + row +
", " + place);
    }
    this.row = row;
    this.place = place;
```

```
  }

  public int getRad() {
    return row;
  }

  public int getPlass() {
    return place;
  }

  public String toString() {
    String res = super.toString();
    res += " Rad: "+ row + " Plass: " + place;
    return res;
  }
}
```