UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Prøveeksamen i	INF1010 — Objektorientert programmering
Innleveringsfrist:	26. mai 2015 kl. 08.00
Tid for eksamen:	10.15 – 16.15 (6 timer)
Oppgavesettet er på 4 sider.	
Vedlegg:	4 sider
Tillatte hjelpemidler:	Alle

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

Les igjennom hele oppgaveteksten før du begynner å svare. Noter ned uklarheter/spørsmål under gjennomlesingen slik at du har spørmålene klare når faglærer kommer. Tren på å skrive svaret for hånd og noter ned hvor mange minutter du bruker på hver oppgave.

Fristen gjelder dem som ønsker «sensur» av sin besvarelse. Da må den være levert i Devilry innen fristen. Man får tilbakemelding med ca. karakter på rettegruppene i tidsrommet 27.5. - 2.6.

Oppgave 1

Du skal skrive klasse- og interfacedefinisjoner slik at datastrukturen som er tegnet i vedlegg 1 kan opprettes. Du trenger ikke skrive konstruktører og metoder. Heller ikke objektvariable. På tegninga er typen til variablene skrevet over, og navnet skrevet under variabelen (boksen). BStud er forkortelse for BachelorStudent. MStud er forkortelse for MasterStudent. Du skal skrive 8 klasser og grensesnitt slik at følgende tilordninger (gitt tilstanden slik vedlegg 1 viser) blir *lovlige*:

```
personer[0] = new MStud();
personer[4] = new AnsattBStud();
a = (Ansatt) personer[4];
personer[5] = new Person();
```

Følgende tilordninger (etter at de lovlige over er utført) skal være *ulovlige*:

```
aa = new Ansatt();
p = (Professor) personer[4];
personer[0] = new Student();
a = (Ansatt) personer[3];
a = new MStud();
a = (Ansatt) personer[5];
bs = new AnsattBStud();
bs = new MStud();
MStud ms = bs;
```

(Fortsettes på side 2.)

Oppgave 2

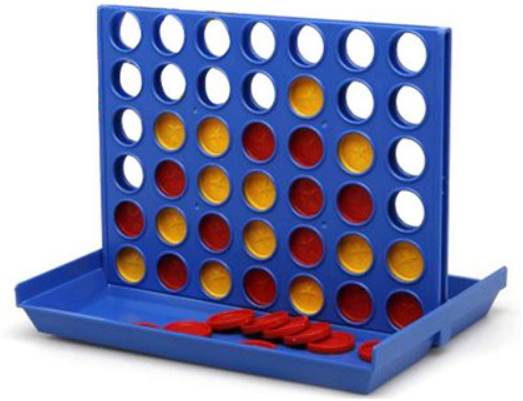
Når programmet i vedlegg 2 kompileres og kjøres slik:

```
javac Oppgave.java
java Oppgave A123 B456 C789
```

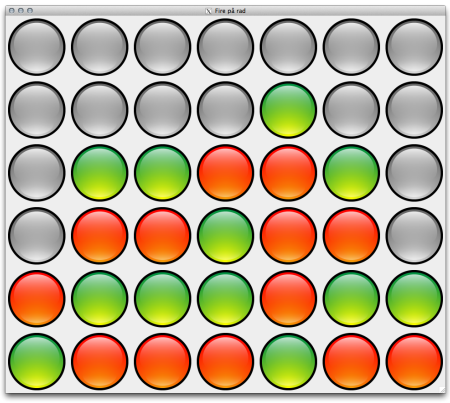
blir det opprettet en datastruktur. Tegn denne datastrukturen når programmet har kommet der det er merket med kommentaren «OPPGAVE 2». Eksempler på hvordan vi tegner varaiable, metoder, konstruktører og objekter framgår av vedlegg 1 og 3.

Oppgave 3

Skriv et program som tegner spillet «Fire på rad» i et vindu på skjermen ved hjelp av java.Swing. Spillet vi skal modellere kan sees på figuren nedenfor:



Spillebrettet består av 6 rader og 7 kolonner med tomme felter (egentlig et hull med plass til en brikke). I det fysiske spillet slippes runde brikker gjennom et hull i kolonnene, slik at de havner nederst i den kolonnen. Det er to spillere som har brikker med hver sin farge og de slipper brikker ned annenhver gang. Et lignende brett laget med programmet du skal lage ser slik ut:



(Fortsettes på side 3.)

I vinduet skal spillerne peker med musa i ei rute i den kolonnen brikken skal plasseres og klikker med musa der. Klikket skal ha som effekt at feltet/ruta nederst i kolonna som er tom markeres med en av fargene. Vi bruker fargene rød og grønn. Ved neste klikk skjer det samme, men med den andre fargen. Hvis det klikkes i en kolonne som er full, skal klikket ikke ha effekt og den samme fargen brukes ved første klikk i en ledig kolonne.

Du skal ikke skrive hele programmet, men fylle ut de delene som er utelatt i vedlegg 4. Legg merke til at det ikke er ruta det klikkes i som skal endres, men den nederste tomme ruta i samme kolonne. For å få til dette trenger du noe ekstra datastruktur som programskissen ikke viser.

For å fange opp museklikk skal du bruke grensesnittet MouseListener. Dette grensesnittet ser slik ut:

```
public interface MouseListener extends EventListener {
    public void mouseClicked(MouseEvent e);
    public void mouseEntered(MouseEvent e);
    public void mouseExited(MouseEvent e);
    public void mousePressed(MouseEvent e);
    public void mouseReleased(MouseEvent e);
}
```

Oppgave 4

Skriv en enkel beholderklasse hvor datastrukturen er nodeobjekter lenket sammen i ei liste og som har settInn- og taUt-metoder slik at beholderen fungerer som en kø (først inn først ut; FIFO). Beholderen kan kun inneholde objekter av klasser som implementerer grensesnittet Comparable<T>, dvs. objekter av samme type som er sammenlignbare med seg selv. En invariant tilstandspåstand for beholderen er at for et vilkårlig par av objekter i beholderen, a og b, skal det være slik at a.compareTo(b) aldri er 0. Mao. at det ikke skal finnes to (eller flere) like objekter i beholderen.

Oppgave 5 (Arv og polymorfi)

Gitt programskissen:

```
abstract class Bil {
  private String regNr;
}

abstract class Varebil extends Bil {}
class Lastebil extends Varebil {}
abstract class Personbil extends Bil {}
class Drosje extends Personbil {}
class Privatbil extends Personbil {}
```

Vi bestemmer at to biler er like, det er samme bil, hvis de har samme registreringsnummer. Skriv de 6 klassene på nytt og gjør endringer og tillegg slik at objekter av de ikke-abstrakte klassene kan lages med f.eks. new Privatbil("IN10101"). Det er et krav at registreringsnummeret skal ha 7 tegn.

(Fortsettes på side 4.)

Hverken mer eller mindre. Gjør også endringer slik at det er mulig å legge bilobjekter inn i beholderen fra forrige oppgave.

Oppgave 6

Mange filer (N) med ord er lest inn i like mange arrayer av type String:

```
String [][] ordfiler = new String[N][500000]
```

Du har en flott datamaskin med 32 prosessorer og du skal se hvor effektiv den er ved å lage et javaprogram som lager en oversikt over lengden på ordene i arrayene. Du er interessert i hvor mange ord som har lengde 1 (en bokstav), hvor mange som har lengde 2 (2-bokstavsord), 3 osv. Ord som har lengde større enn 12 telles sammen med ord med lengde 13 bokstaver. Lengden til en tekststreng returneres av metoden length() som er i klassen String.

I dette programmet skal du så lenge det er flere enn 32 arrayer som ennå ikke er telt opp, ha nøyaktig 32 tråder som går i parallell. Målet er å sitte igjen med en long [ ] som inneholder statsitikk over ordlengdene som om alle ordene lå i samme String-array.

En tråd behandler en String-array om gangen. Når det ikke er flere arrayer igjen å behandle, leverer tråd nr i (i er mellom 0 og 31) sine resultater i long-arrayen histogram.

Når alle trådene har levert, gjøres følgende

```
for (int i=1; i<13; i++)
    System.out.println("Antall ord med lengde "
                       + i + " : " + histogram[i]);

System.out.println("Antall ord med lengde 13 eller mer : "
                   + histogram[13]);
```

Oppgave 7

Innstikksortering i ei lenkeliste:

Skriv en metode void sorter() i beholderen/lenkelista i oppgave 4 som ordner rekkefølgen på objektene slik at minste ligger først. Dette kan f.eks. gjøres ved å ta ut det største objektet fra beholderen og sette dette foran i ei ny lenkeliste. Gjenta på resten av lista. Siden det største ble satt inn foran først, vil det til slutt være bakerst.

Lykke til!

Stein Michael Storleer