Ugeseddel 5

29. september - 6. oktober

- Læs afsnit 8.3 (side 333 - 366) i lærebogen.
- Løs opgave 8.1 samt de to opgaver på de næste sider.

Ekstraopgave 2

Betragt nedenstående programskitse:

```
class PersonList {
    PersonList() {}

    PersonList(String fileName) { ... }

    void add(Person p) {
        p.nextPerson = firstPerson;
        firstPerson = p;
    }

    void toFile(String fileName) { ... }

    String toString() { ... }

    Person firstPerson;
}

class Person {
    Person(String name) {
         this.name = name;
    }

    String name;
    Person nextPerson;
}
```

fortsættes

```
public class Test {
    public static void main(String[] args) {
        PersonList list1 = new PersonList();
        list1.add(new Person("Hansen"));
        list1.add(new Person("Jensen"));
        list1.add(new Person("Nielsen"));
        list1.toFile("personer");
        PersonList list2 = new PersonList("personer");
        System.out.println(list2);
    }
}
```

Færdiggør programmet ved anvendelse af **objekt-serialisering**, således at en kørsel resulterer i følgende udskrift:

```
Nielsen
Jensen
Hansen
```

Ekstraopgave 3

Nedenfor er vist et udkast til en samling Java-klasser, der er beregnet til katalogisering af dyr.
Klassen `Animal` er en fælles overklasse for de to klasser `Carnivore` og `Herbivore`,
der repræsenterer henholdsvis mængden af kødædende og planteædende dyr.

```
class Animal {
    Animal(String n) {
        /* Kode ikke medtaget. Se spørgsmål 1 */
    }

    static void printAnimals() {
        /* Kode ikke medtaget. Se spørgsmål 3 */
    }

    static Animal firstAnimal;

    String name;
    Animal next;
    public String toString() { return name; }
}

class Carnivore extends Animal {
    Carnivore(String n, int m) {
        /* Kode ikke medtaget */
    }

    int meatNeeded;
}

class Herbivore extends Animal {
    Herbivore(String n, int g) {
        /* Kode ikke medtaget. Se spørgsmål 2 */
    }

    int grassNeeded;
}
```

fortsættes

Hvert `Animal`-objekt er forsynet med en tekststreng `name`, der angiver dyrets navn. Referencen `next` benyttes til opbevaring af `Animal`-objekterne i en envejsliste, idet den statiske reference `firstAnimal` peger på det forreste objekt i denne liste, og `next` angiver objektets efterfølger i listen.

> **Spørgsmål 1** Programmér en konstruktør for klassen `Animal`, som tildeler `name` en værdi og indsætter det genererede objekt forrest i envejslisten.

For hvert planteædende dyr (`Herbivore`-objekt) angives dets daglige fødebehov ved hjælp af attributten `grassNeeded`.

> **Spørgsmål 2** Programmér en konstruktør for klassen `Herbivore`, som tildeler værdier til `name` og `grassNeeded`, og som indsætter objektet i envejslisten ved hjælp af konstruktøren for `Animal`.
>
> **Spørgsmål 3** Programmér metoden `printAnimals`, der gennemløber envejslisten af dyr og udskriver deres navne.
>
> **Spørgsmål 4** Programmér en metode `printHerbivores`, der kun udskriver navnene på de af dyrene i envejslisten, der er planteædere (`Herbivore`-objekter).

fortsættes

Klassen `Herbivore` benyttes nu som overklasse for klassen `Giraffe`, der repræsenterer mængden af giraffer. Attributten `neckLength` angiver halslængden for en giraf.

```
class Giraffe extends Herbivore {
    Giraffe(String n, int g, double nL) {
        /* Kode ikke medtaget */
    }

    double neckLength;
}
```

Spørgsmål 5 Lad der være givet følgende erklæringer:

```
Animal a;
Herbivore h;
Carnivore c;
Giraffe g1, g2;
```

Antag at objekterne `a`, `h`, `c`, `g1` og `g2` eksisterer. Hvilke af følgende sætninger vil da give fejl under oversættelsen?

(1) `a = g1;`
(2) `a = h;`
(3) `g1 = a;`
(4) `g1 = g2;`
(5) `g1 = (Giraffe) h;`
(6) `g1 = (Carnivore) h;`
(7) `g1 = (Giraffe) c;`