IN1010

Sensorveiledning og løsningsforslag

Ny og utsatt prøve 11. august 2026

Løsningsforslag

Her er et løsnings*forslag* til konteeksamen i august 2026. Det finnes mange andre fullgode løsninger til oppgavene.

Sensorveiledning

Generelt gjelder at studenten får full uttelling for det enkelte delspørsmålet om

- koden er riktig,
- den gjør det oppgaven ber om og
- den er oversiktlig og lettlest.

Effektivitet er ikke viktig i dette emnet.

Rene skrivefeil ignoreres vanligvis, men feil som kan tyde på manglende forståelse (for eksempel kode som ikke passer inn eller svarer på oppgaven) skal gi trekk.

Dette ignoreres:

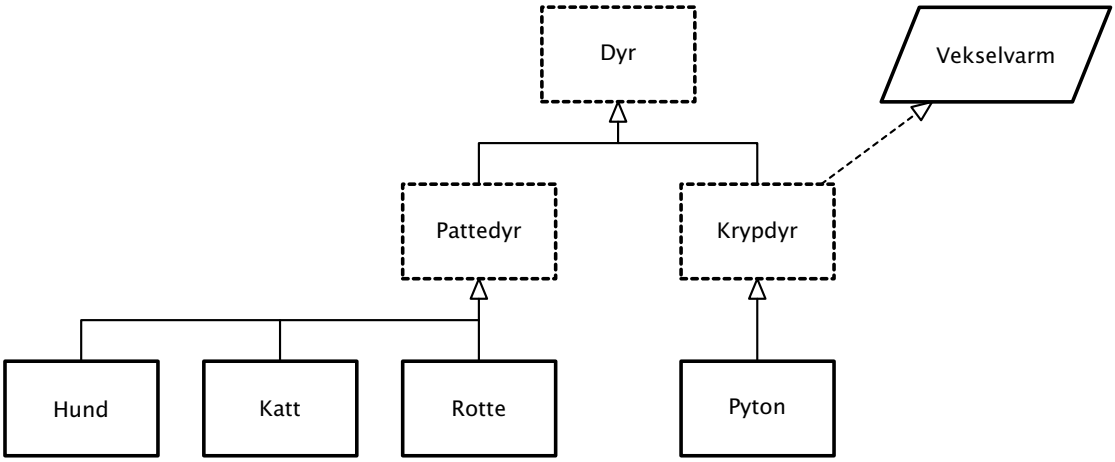
- koden mangler enkelte «**;**» , «**{**» eller «**}**» om koden ellers gir mening
- småfeil i navn

Studentene har fått beskjed om at importspesifikasjoner, kommentarer eller tilgangsmodifikatorer (**public**/**private**/**protected**) ikke er viktig på eksamen.

Feil i kode kopiert mellom ulike deloppgaver trekkes kun én gang.

Se også «Generelt om oppgaven» i oppgavesettet.

Oppgave 1 — Klassestruktur (6 poeng)



Det viktigste her er å vise at man har skjønt hvordan de ulike klassene henger sammen, dvs hvilke klasser som er subklasser av hverandre.

- Det trekkes 2 poeng om pilene ikke har retning eller har gal retning.
- 1 poeng trekk for ikke å angi riktig hvilke klasser som skal være abstrakte.
- Det gis −1 poeng om Vekselvarm ikke er koplet til Krypdyr og enda −1 poeng om interfacet mangler.

Oppgave 2 — Java-programmering (10 poeng)

```
abstract class Dyr {
    String navn;

    Dyr (String id) {
        navn = id;
    }
}

abstract class Pattedyr extends Dyr {
    Pattedyr (String id) {
        super(id);
    }
}

interface Vekselvarm {
    double idealtemperatur ();
}

abstract class Krypdyr extends Dyr implements Vekselvarm {
    double temperatur;

    Krypdyr (String id, double temp) {
        super(id);
        temperatur = temp;
    }

    @Override
    public double idealtemperatur () {
        return temperatur;
    }
```

```
}

class Hund extends Pattedyr {
    Hund (String id) {
        super(id);
    }
}

class Katt extends Pattedyr {
    Katt (String id) {
        super(id);
    }
}

class Rotte extends Pattedyr {
    Rotte (String id) {
        super(id);
    }
}

class Pyton extends Krypdyr {
    Pyton (String id, double temp) {
        super(id,temp);
    }
}
```

Hovedformålet med denne oppgaven er å sjekke om studenten kan skrive klasse- og subklassedeklarasjoner i Java. Det gis trekk om

- det er feil i selve deklarasjonene
- det ikke er samsvar mellom Java-koden og syntaksdiagrammet i oppgave 1
- instansvariablene mangler eller er galt deklarert
- konstruktørene ikke initierer instansvariablene eller de brukes galt (dvs bruker ikke **super**())

Siden det ikke står noe om det i oppgaveteksten, kan idealtemperaturer for krypdyr deklareres enten som heltall eller flyt-tall.

Oppgave 3 — Redefinisjon (3 poeng)

```
abstract class Dyr {
        :
    @Override
    public String toString () {
        return navn;
    }
}
```

Dette er en meget enkel oppgave som sjekker om studentene kan redefinere en bibliotekmetode.

- ½ poeng trekk om svaret mangler @Override.
- ½ poeng trekk om de mangler **public**.

Oppgave 4 — Listestruktur (8 poeng)

```
class Bur {
    private DyrNode første = null;
```

```
    private class DyrNode {
        Dyr dyr;
        DyrNode neste;

        DyrNode (Dyr d) {
            dyr = d;  neste = null;
        }
    }

    void settInn (Dyr d) {
        DyrNode dn = new DyrNode(d);
        dn.neste = første;
        første = dn;
    }
}
```

Her skal studentene vise at de behersker programmering av lister med interne noder.

- Bruk av interne noder er viktig her, så det trekkes 4 poeng om koden ikke benytter slike noder.
- Det trekkes 2 poeng om innsetting *først* i listen ikke er riktig.
- Trekk 1 poeng om det brukes dobbeltlinket liste (for da er det sannsynlig at koden er rått kopiert fra en tidligere eksamen).

Det er helt OK om koden bruker en sistepeker i tillegg til førstepekeren.

Oppgave 5 — Iterator (7 poeng)

```
class Bur implements Iterable<Dyr> {
        :
    @Override
    public Iterator<Dyr> iterator () {
        return new BurIterator();
    }

    private class BurIterator implements Iterator<Dyr> {
        DyrNode denne = første;

        @Override
        public boolean hasNext () {
            return denne != null;
        }

        @Override
        public Dyr next () {
            Dyr d = denne.dyr;
            denne = denne.neste;
            return d;
        }
    }
}
```

Denne oppgaven er en sjakk på om studentene har forstått hva en iterator er og hvorledes den programmeres i Java.

Oppgave 6 — Listegjennomsøking (8 poeng)

```
class Bur implements Iterable<Dyr> {
        :
    int antallHunder () {
        int ant = 0;
        for (Dyr dx: this) {
            if (dx instanceof Hund) ant++;
        }
        return ant;
    }

    int antallKatter () {
        int ant = 0;
        for (Dyr dx: this) {
            if (dx instanceof Katt) ant++;
        }
        return ant;
    }

    int antallRotter () {
        int ant = 0;
        for (Dyr dx: this) {
            if (dx instanceof Rotte) ant++;
        }
        return ant;
    }

    int antallPyton () {
        int ant = 0;
        for (Dyr dx: this) {
            if (dx instanceof Pyton) ant++;
        }
        return ant;
    }

    // Hjelpemetode for oppgave 7
    int totalantall () {
        return antallHunder() + antallKatter() +
               antallRotter() + antallPyton();
    }
}
```

Her sjekkes om studentene kan lete gjennom en lenkeliste og sjekke på objekters klasse.

- Løsningsforslaget benytter iteratoren deklarert i forrige oppgave, men det er helt greit om det i stedet brukes en peker som løper gjennom listen.
- Det er det samme om det brukes **instanceof** eller virtuelle metoder.

Oppgave 7 — Polymorfi (13 poeng)

```
class Hund extends Pattedyr {
        :
    @Override
    boolean settInnIBur (Bur bur) {
        if (bur.antallPyton() > 0) return false;
        if (bur.antallRotter() > 0) return false;
        if (bur.totalantall() >= 4) return false;
```

```
        bur.settInn(this);
        return true;
    }
}

class Katt extends Pattedyr {
        :
    @Override
    boolean settInnIBur (Bur bur) {
        if (bur.antallPyton() > 0) return false;
        if (bur.antallRotter() > 0) return false;
        if (bur.totalantall() >= 4) return false;
        bur.settInn(this);
        return true;
    }
}

class Rotte extends Pattedyr {
        :
    @Override
    boolean settInnIBur (Bur bur) {
        if (bur.antallPyton() + bur.antallHunder() +
            bur.antallKatter() > 0) return false;
        if (bur.antallRotter() >= 10) return false;
        bur.settInn(this);
        return true;
    }
}

class Pyton extends Krypdyr {
        :
    @Override
    boolean settInnIBur (Bur bur) {
        if (bur.totalantall() > 0) return false;
        bur.settInn(this);
        return true;
    }
}
```

Denne oppgaven er en sjekk på om studentene kan skrive virtuelle metoder, men også om de kan programmere logiske tester basert på noen spesifikasjoner.

- Det trekkes 8 poeng om oppgaven løses uten å redefinere `settInnIBur` i de ulike dyreklassene.
- Det trekkes 1–4 poeng for logikkfeil i testingen.

Oppgave 8 — Hovedprogram (6 poeng)

```
class Dyrepensjonat {
    static Bur[] alleBurene;

    static void feil (String melding) {
        System.out.println("Dyrepensjonat: " + melding);
        System.exit(9);
    }

    public static void main (String[] args) {
        if (args.length != 2)
```

```
            feil("Bruk: java Dyrepensjonat n fil");
        alleBurene = new Bur[Integer.parseInt(args[0])];
    }
}
```

Her må studentene vise at de kan håndtere det å skrive et hovedprogram (med en main-metode) med argumenter. Dessuten skal de kunne opprette arrayer.

- Det trekkes 2 poeng for manglende kode for å sjekke antallet argumenter.
- Feilaktige eller manglende kode for å opprette arrayen gir 2 poeng trekk.
- Å glemme metoden feil gir 2 poeng trekk. Det gis ikke trekk for å glemme å angi at den må være **static**.
- Trekk 1 poeng om argument ikke konverteres fra String til **int**.

Oppgave 9 — Innlesing fra fil (11 poeng)

```
class Dyrepensjonat {
        :
    // Hjelpemetode
    private static void plasser (Dyr d) {
        for (int ix = 0;  ix < alleBurene.length;  ix++) {
            if (alleBurene[ix] == null)
                alleBurene[ix] = new Bur();
            if (d.settInnIBur(alleBurene[ix])) return;
        }
        feil("Det var ikke plass til " + d);
    }

    static void lesAlleDyrene (String filnavn) {
        Scanner s = null;
        try {
            s = new Scanner(new File(filnavn));
        } catch (FileNotFoundException e) {
            feil("Kan ikke lese " + filnavn);
        }

        while (s.hasNextLine()) {
            String linje = s.nextLine();
            String[] data = linje.split(";");
            if (data[0].equals("Hund"))
                plasser(new Hund(data[1]));
            else if (data[0].equals("Katt"))
                plasser(new Katt(data[1]));
            else if (data[0].equals("Rotte"))
                plasser(new Rotte(data[1]));
            else if (data[0].equals("Pyton"))
                plasser(new Pyton(data[1],Double.parseDouble(data[2])));
            else
                feil("Datafeil: " + linje);
        }
        s.close();
    }
}
```

I denne oppgaven skal studentene demonstrere åpning av tekstfil og lesing linje for linje. De skal også kunne håndtere ulike typer data i filen, og de skal vise enkel håndtering av en array.

- Det gis inntil 4 poeng trekk for manglende eller feilaktig åpning av filen, inkludert testing på feil.

- Trekk 1 poeng om de ikke bruker metoden feil til å skrive ut en feilmelding.
- Feil og mangler ved dekodingen av data kan gi inntil 4 poeng trekk.
- Innsettingen av data i arrayen på riktig måte teller 5 poeng. Sjekk spesielt om pytonslanger håndteres riktig.

Oppgave 10 — Rekursjon (10 poeng)

```
class Bur implements Iterable<Dyr> {
    private DyrNode første = null;
        :
    private class DyrNode {
            :
        void skrivDyrRekursivt () {
            if (neste != null)
                neste.skrivDyrRekursivt();
            System.out.print(dyr + " ");
        }
    }

    void skrivDyr () {
        if (første != null)
            første.skrivDyrRekursivt();
    }
}
```

Dette er en oppgave som viser hvor godt studentene behersker rekursjon.

- En ikke-rekursiv løsning kan maksimalt få 2 poeng.
- Trekk 3 poeng om en tom liste vil gi kjørefeil.

Oppgave 11 — Utskrift (5 poeng)

```
class Dyrepensjonat {
        :
    static void skrivAlleDyrene () {
        for (int ix = 0;  ix < alleBurene.length;  ix++) {
            if (alleBurene[ix] != null) {
                System.out.print("Bur nr " + ix + ": ");
                alleBurene[ix].skrivDyr();
                System.out.println();
            }
        }
    }
}
```

Dette er en ganske enkel oppgave som tester gjennomløp av arrayer samt bruk av metoder studentene har skrevet tidligere under eksamen.

Oppgave 12 — Monitor (13 poeng)

```
class Dyresluse {
    Dyr buffer = null;

    Lock lås = new ReentrantLock();
    Condition venterPåDyr = lås.newCondition();
    Condition venterPåLedig = lås.newCondition();
```

```
    void settInnISlusen (Dyr d) {
        lås.lock();
        try {
            while (buffer != null)
                venterPåLedig.await();
            buffer = d;
            venterPåDyr.signalAll();
        } catch (InterruptedException e) {
            return;
        } finally {
            lås.unlock();
        }
    }

    Dyr hentUtAvSlusen () {
        lås.lock();
        try {
            while (buffer == null)
                venterPåDyr.await();
            Dyr res = buffer;
            buffer = null;
            venterPåLedig.signalAll();
            return res;
        } catch (InterruptedException e) {
            return null;
        } finally {
            lås.unlock();
        }
    }
}
```

Denne oppgaven skal vise om studentene forstår hvordan en monitor fungerer (spesielt ved venting) og hvorledes den programmeres i Java.

- Deklarasjon og opprettelse av lås og ventebetingelser teller 2 poeng.
- Trekk 4 poeng om de to metodene ikke låses skikkelig.
- Ved manglende eller gal venting (dvs await og signal/signalAll) trekkes inntil 5 poeng.
- Trekk 1 poeng for manglende håndtering av InterruptedException.
- Deklarasjon og bruk av bufferen teller 1 poeng.