UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	INF1010 — Objektorientert programmering
Dato:	9. juni 2016
Tid for eksamen:	09.00 – 15.00 (6 timer)
Oppgavesettet er på 7 sider.	
Vedlegg:	Kapittel 20 fra BIG JAVA
Tillatte hjelpemidler:	Læreboka (med notater i)

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

Les gjennom hele oppgaveteksten før du begynner å svare. Noter ned uklarheter/spørsmål under gjennomlesningen slik at du har spørsmålene klare når faglærer kommer. Husk å skrive i besvarelsen der du gjør egne forutsetninger. Forklar med ord hva en kode det ikke eksplisitt er spurt etter, gjør.

Oppgave 1 — Løsningsforslag

a) 4 **b)** 1 **c)** 9

Oppgave 2 — Løsningsforslag

```
void bytt (Node n) {
    Node m = n.forrige;
    Node o = n.neste;
    Node p = o.neste;
    // neste-rekkefølgen er nå m, n, o, p. Ny rekkefølge: m, o, n, p
    n.neste = p;
    o.neste = n;
    n.forrige = o;
    o.forrige = m;
    m.neste = o;
    p.forrige = n;
}
```

Oppgave 3 — Løsningsforslag

```
public class LenkeListe<T extends Comparable<T>> {
    private ListeHode lhode;   // listehodet
    private ListeHale lhale;   // listehalen
    private int antall;

    LenkeListe() {
        lhode = new ListeHode(null);
        lhale = new ListeHale(lhode, null);
```

(Fortsettes på side 2.)

```
        lhode.neste = lhale ;
        antall = 0;
    }

    private class ListeHode extends AbstrNode {
        ListeHode( T t ) { super(t); }
    }

    private class ListeHale extends AbstrNode {
        AbstrNode forrige = null; // peker til siste Node (forrige)
        ListeHale (AbstrNode lhode, T t) {
            super(t);
            forrige = lhode;
        }
    }

    private class Node extends AbstrNode {
        Node(T t) { // t peker til objekt av type Comparable<T>
            super(t);
        }
    }
}
```

Oppgave 4 — Løsningsforslag

```
        int compareTo(AbstrNode k) {
            return -9; // eller et annet negativt tall
        }
```

Oppgave 5 — Løsningsforslag

```
        int compareTo(AbstrNode k) {
            return 9; // eller et annet positivt tall
        }
```

Oppgave 6 — Løsningsforslag:

```
    public void settInnBak (T nyComparable){
        Node nyNode = new Node(nyComparable);
        lhale.forrige.neste = nyNode;
        lhale.forrige = nyNode;
        nyNode.neste = lhale;
        antall++;
    }
```

Oppgave 7 — Løsningsforslag

```
// ListeHode
        void settInnOrdnet(AbstrNode ny) {
            if (neste.compareTo(ny) >= 0) {
                // objektet skal inn etter listehodet
                ny.neste = neste;
                neste = ny;
                // hvis ny havner bakerst, må vi opprettholde invarianten:
                if ( ny.neste == lhale ) lhale.forrige = ny;
                antall++;
```

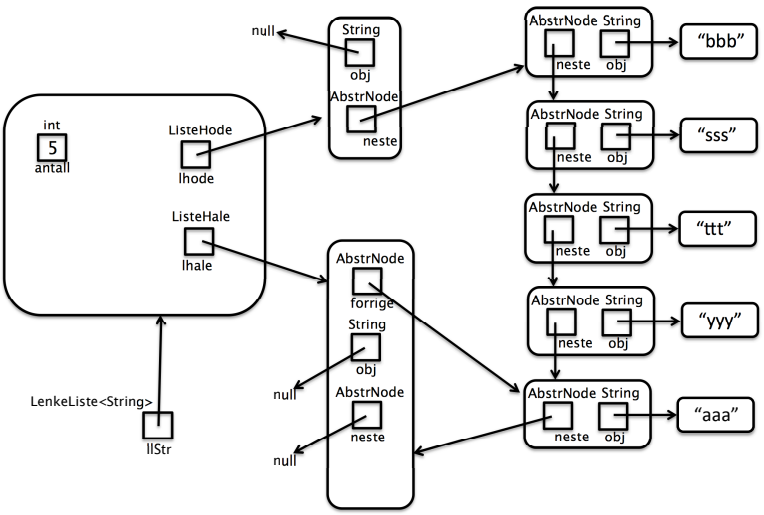
(Fortsettes på side 3.)

```
            }
            else neste.settInnOrdnet(ny);
        }


// ListeHale
        void settInnOrdnet(AbstrNode ny) {
            System.out.println("FEIL: Denne metoden skal aldri kalles!");
        }


// Node
        void settInnOrdnet(AbstrNode ny) {
            if ( neste.compareTo(ny) < 0)
                neste.settInnOrdnet(ny);
            else if (neste.compareTo(ny) >= 0) {
                ny.neste = neste;
                neste = ny;
                // hvis ny havner bakerst, må vi opprettholde invarianten:
                if ( ny.neste == lhale ) lhale.forrige = ny;
                antall++;
            }
        }
```

Oppgave 8 — Løsningsforslag:

```
    public T taUtForan () throws Exception {
        if ( antall() > 0 ) {
            AbstrNode ut = lhode.neste;
            lhode.neste = ut.neste;
            antall--;
            return ut.obj;
        }
        else throw new Exception("Kall på taUtForan i tom liste");
    }
```

Oppgave 9 — Løsningsforslag:



Oppgave 10 — Løsningsforslag:

(Fortsettes på side 4.)

```
private interface Elem {
    int compareTo(Elem k);
    void settInnOrdnet(Elem k);

    // void test(T t); // Kan ikke referere til T her:
    // non-static type variable T cannot be referenced from a static context

    // void settInnOrdnet(Node k); // Kan ikke referere til Node her:
    // non-static class Lenkeliste10.Node cannot be referenced from a static context
}


private class ListeHode implements Elem {
    Elem neste;

    public int compareTo(Elem k) {
        return -9;
    }

    public void settInnOrdnet(Elem n) {
        settInnNy( (Node) n );
    }

    public void settInnNy(Node ny) { // uforandret

}

private class Node implements Elem {
    T obj;
    Elem neste;

    Node(T t) {
        obj = t;
    }

    public int compareTo(Elem k) {
        return obj.compareTo(((Node) k).obj);
    }

    public void settInnOrdnet(Elem n) {
        if ( neste.compareTo(n) < 0)
            neste.settInnOrdnet(n);
        else if (neste.compareTo(n) > 0) {
            Node ny = (Node) n;
            ny.neste = neste;
            neste = ny;
            if ( ny.neste == lhale ) lhale.forrige = ny;
            antall++;
        }
    }

}

// ListeHale uforandret bortsett fra at typen AbstrNode byttes med Elem
private class ListeHale  implements Elem { }
```

(Fortsettes på side 5.)

Oppgave 11 — Løsningsforslag:

a) Vi vet at en av dem må være tom, men ikke begge. Eller negasjonen av while-betingelsen.

b)

```
        while ( fraA != null ) {
            c.settInnBak(fraA);
            fraA = a.taUtForan();
        }
        while ( fraB != null ) {
            c.settInnBak(fraB);
            fraB = b.taUtForan();
        }

        return c;
```

Oppgave 12 — Løsningsforslag

```
    public static void main (String[] a) {
        Monitor monitor = new Monitor();
        CountDownLatch barriere = new CountDownLatch(39);
        String[] ordene = lesInnStringArray(a[0]);

        for (int i = 0; i < 39; i++) {
        new SorterOgFlett(i, ordene, monitor, barriere,
                          i*10000, 10000).start();
            // Starter tråd nr. i.
        }
        try {
            barriere.await();
        }
        catch (InterruptedException ex){ }

        LenkeListe<String> sluttresultat = monitor.hentResultat();
        // sluttresultat er sortert.
    }

class Monitor {       // monitor med plass til 1 lenkeliste
    LenkeListe<String> ll = null;

    synchronized LenkeListe<String> leggInnEllerTaUt(LenkeListe<String> lliste) {
        LenkeListe<String> ret = null;
        if (ll == null)
            ll = lliste;

        else {
            ret = ll;
            ll = null;
        }
        return ret;
    }

    LenkeListe<String> hentResultat() { return ll; }
}

class SorterOgFlett extends Thread {
```

(Fortsettes på side 6.)

```
    Monitor monitor;
    CountDownLatch barriere;
    String[] ordene;
    int start, antall;
    int trNr;

    SorterOgFlett(int tn, String[] ord, Monitor m,
                  CountDownLatch b, int st, int ant) {
        monitor = m;
        barriere = b;
        ordene = ord;
        start = st;
        antall = ant;
        trNr = tn;
    }

    LenkeListe<String> flett(LenkeListe<String> a, LenkeListe<String> b) { ... }

    public void run() {
        LenkeListe<String> ll2 = null, ll3 = null;
        LenkeListe<String> ll = new LenkeListe<String>();
        // Innstikksortering:
        for (int j = start; j < start+antall; j++) {
            ll.settInnOrdnet(ordene[j]);
        }
        // Tråd nr. trNr er ferdig med innsettingen. Fletting:
        ll2 = ll;
        ll3 = monitor.leggInnEllerTaUt(ll);
        while (ll3 != null) {
            ll = flett(ll2, ll3);
            ll2 = ll;
            ll3 = monitor.leggInnEllerTaUt(ll);
        }
        // Tråd nr trNr avslutter ...
        barriere.countDown();
    }
}
```

Oppgave 13 — Løsningsforslag:

```
interface I {}
interface IA {}
interface IBD extends I {}
interface IC {}
interface IG {}
abstract class C extends A implements IC {}
abstract class D extends C implements IBD {}
class A implements I, IA {}
class B extends A implements IBD {}
class E extends C {}
class G extends D implements IG {}
```

Oppgave 14 — Løsningsforslag:

a) 5

(Fortsettes på side 7.)

b) 17

c) 21

d) 13

Oppgave 15 — Løsningsforslag:

Her er løsningsforslaget vist i programmet nedenfor. De utkommenterte setningene feiler. K ved kompilering, R ved kjøring. Riktig svar her er samme program med de 6 setningene som blir igjen.

```
1
2  public static void main(String [] args) {
3
4      C c = new E();
5      IC ic = c;
6      // IBD ibd = c; // - K
7      Object o = new B();
8      IA ia = (IA) o;
9      // ic = (IC) o; // - R ClassCastException: B cannot be cast to IC
10     A a = (C) ic;
11     // E e = new A(); // - K
12     // IBD ibd = (IBD) a; // - R ClassCastException: E cannot be cast to IBD
13     a = (B) o;
14     // c = new C(); // -K C is abstract; cannot be instantiated
15
16 }
```

Oppgave 16 — Løsningsforslag: