

Oppgave 1

Bruk en lenkeliste til å løse de følgende oppgavene.

1a)

Skriv ut elementene i lenkelisten først i riktig, så i motsatt rekkefølge ved hjelp av rekursjon. Hvor mye forskjell trenger det være i disse to metodene?

Løsning:

```
//Forklar hvorfor det er brukt overloading, mao. Fordi vi vet at vi alltid vil begynne med hode-noden, men da trenger man ikke fra hovedprogrammet å vite dette.
public void skriv_rek_forst(){
    skriv_rek_forst(hode);
}
public void skriv_rek_forst(Node denne){
    if (denne == null){
        return;
    }
    System.out.println(denne.data);
    skriv_rek_forst(denne.neste);
}

public void skriv_rek_sist(){
    skriv_rek_sist(hode);
}
public void skriv_rek_sist(Node denne){
    if (denne == null){
        return;
    }
    skriv_rek_sist(denne.neste);
    System.out.println(denne.data);
}
}
```

1b)

Finne størrelse på lenkeliste med rekursjon.

Løsning:

```
public int antNoder(){
    return antNoder(hode, 0);
}

public int antNoder(Node denne, int ant){
    if (denne == null){
        return ant;
    }
    return antNoder(denne.neste, ant+1);
}
}
```

Testprogrammet fra uke 7, med litt ekstra for nye oppgavene:

```
class TestFifoListe {
    //Teste Fifo-listen:
    public static void main(String[] args) {
        FIFOListe<String> liste = new FIFOListe<>();
        liste.leggTil("Hei");
    }
}
```

```

liste.leggTil("paa");
liste.leggTil("deg");
liste.leggTil("!");
liste.skrivUtListe();

System.out.println("Iterator-utskrift:");
for(String t: liste){
    System.out.println(t);
}

liste.skriv_rek_sist();
System.out.println();
liste.skriv_rek_forst();

System.out.println("ANT NODER: " + liste.antNoder());
}
}

```

Oppgave 2

2a)

Vi har en rekke med kaniner og alle kaniner har to ører som står rett til værs. Vi ønsker nå å skrive en rekursiv metode for å regne ut hvor mange ører kaninene har totalt. Metoden skal ta inn antall kaniner.

```

class KaninOreTeller {
    //Teste Fifo-listen:
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 2
        System.out.println(tellKaninOrer(2)); //Forventet resultat 4
        System.out.println(tellKaninOrer(12)); //Forventet resultat 24
        System.out.println(tellKaninOrer(234)); //Forventet resultat 486
    }

    public static int tellKaninOrer(int antallKaniner){
        //kode inn her
    }
}

```

Løsningsforslag

```

class KaninOreTeller {
    //Teste Fifo-listen:
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 2
        System.out.println(tellKaninOrer(2)); //Forventet resultat 4
        System.out.println(tellKaninOrer(12)); //Forventet resultat 24
        System.out.println(tellKaninOrer(234)); //Forventet resultat 486
    }
}

```

```

public static int tellKaninOrer(int antallKaniner){
    if(antallKaniner == 0){
        return 0;
    }
    return tellKaninOrer(antallKaniner - 1) + 2;
}
}

```

2b)

Noen ganger velger kaniner kun å ha ett øre opp. I dette tilfellet vil kaniner på partalls plasser (2,4,6, ...) ha to ører oppe, mens kaniner på oddetalls plasser (1,3,5, ...) vil kun ha ett øre oppe. Endre metoden i **2a** slik at den tar hensyn til dette

```

class KaninOreTeller {
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 1
        System.out.println(tellKaninOrer(2)); //Forventet resultat 3
        System.out.println(tellKaninOrer(12)); //Forventet resultat 18
        System.out.println(tellKaninOrer(234)); //Forventet resultat 351
    }

    public static int tellKaninOrer(int antallKaniner){
        //kode inn her
    }
}

```

Løsningsforslag

```

class KaninOreTeller {
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 2
        System.out.println(tellKaninOrer(2)); //Forventet resultat 4
        System.out.println(tellKaninOrer(12)); //Forventet resultat 24
        System.out.println(tellKaninOrer(234)); //Forventet resultat 486
    }

    public static int tellKaninOrer(int antallKaniner){
        if(antallKaniner == 0){
            return 0;
        }

        if(antallKaniner % 2 == 0){
            return tellKaninOrer(antallKaniner - 1) + 2;
        }
        return tellKaninOrer(antallKaniner - 1) + 1;
    }
}

```

Oppgave 3

3a)

Skriv en metode som rekursivt finner antall forekomster av en string i en annen string. Du kan anta at ingen av stringen er tomme (""). (Et tips her kan være å bruke substring- og length-metodene til String)

```
class StringOprasjoner {
    public static void main(String[] args) {
        String str = "kattkukatt"
        System.out.println(antallSubStringer(str, "katt")); //Forventet resultat 2
        System.out.println(antallSubStringer(str, "ku")); //Forventet resultat 1
        System.out.println(antallSubStringer(str, "hund")); //Forventet resultat 0

    }

    public static int antallSubStringer(String str, String sub){
        //kode inn her
    }
}
```

Løsningsforslag

```
class StringOprasjoner {
    public static void main(String[] args) {
        String str = "kattkukatt"
        System.out.println(antallSubStringer(str, "katt")); //Forventet resultat 2
        System.out.println(antallSubStringer(str, "ku")); //Forventet resultat 1
        System.out.println(antallSubStringer(str, "hund")); //Forventet resultat 0

    }

    public static int antallSubStringer(String str, String sub){
        if(str.length() < sub.length()){
            return 0;
        }
        if(str.substring(0, sub.length()).equals(sub)){
            return 1 + antallSubStringer(str.substring(1, str.length()), sub);
        }
        return antallSubStringer(str.substring(1, str.length()), sub);
    }
}
```

3b)

Lag en rekursiv metode som tar inn en string hvis en bokstav forekommer 2 ganger etter hverandre. F.eks. ved l i hello skal det plasseres en * i mellom de doble bokstavene (hello blir da hel*lo). (Et tips her kan være å bruke substring-, length- og charAt-metodene til String)

```
class StringOprasjoner {
    public static void main(String[] args) {
```

```

        System.out.println(settInnStjerner("hello")); //Forventet resultat hel*lo
        System.out.println(settInnStjerner("helllo")); //Forventet resultat hel*1*lo
        System.out.println(settInnStjerner("helloo")); //Forventet resultat hel*lo*o
        System.out.println(settInnStjerner("")); //Forventet resultat
        System.out.println(settInnStjerner("trollmannen")); //Forventet resultat
    trollmannen
}

    public static String settInnStjerner(String str){
        //kode inn her
    }
}

```

Løsningsforslag

```

class StringOperasjoner {
    public static void main(String[] args) {
        System.out.println(settInnStjerner("hello")); //Forventet resultat hel*lo
        System.out.println(settInnStjerner("helllo")); //Forventet resultat hel*1*lo
        System.out.println(settInnStjerner("helloo")); //Forventet resultat hel*lo*o
        System.out.println(settInnStjerner("")); //Forventet resultat
        System.out.println(settInnStjerner("trollmannen")); //Forventet resultat
    trollmannen
    }

    public static String settInnStjerner(String str){
        if(str.length() < 2){
            return str;
        }
        if(str.charAt(0) == str.charAt(1)){
            return str.charAt(0) + "*" + settInnStjerner(str.substring(1, str.length()));
        }
        return str.charAt(0) + settInnStjerner(str.substring(1, str.length()));
    }
}

```