LØSNINGSFORSLAG - Seminaroppgaver, uke 5

1.)
Array:
- Har fast størrelse, må holde orden på selv hvor mange elementer man har brukt, lettere å få error
+ Raskere

ArrayList:
- Langsommere (klasse skrevet rundt array)
+ Lettere å bruke, flexibel størrelse, mindre risiko for feil i programmet

2.a-b)

```
// 2.a)
class Node {
   private String verdi;
   private Node neste;

   public Node(String verdi) {
       this.verdi = verdi;
   }

   public void settNeste(Node neste) {
       this.neste = neste;
   }

   public Node hentNeste() {
       return neste;
   }

   public String hentVerdi() {
       return verdi;
   }
}

public class Hovedprogram {
   public static void main(String[] args) {
       // 2.a)
       Node forste = new Node("Hei");
       Node node2 = new Node("paa");
       Node node3 = new Node("deg!");

       forste.settNeste(node2);
       node2.settNeste(node3);

       // 2.b)
       Node peker = forste;
       while (peker != null) {
           System.out.println(peker.hentVerdi());
           peker = peker.hentNeste();
       }
   }
}
```

2.c)

```
class Node<E> {
   // 2.a)
   private E verdi;
   private Node<E> neste;

   public Node(E verdi) {
       this.verdi = verdi;
   }

   public void settNeste(Node<E> neste) {
       this.neste = neste;
   }

   public Node<E> hentNeste() {
       return neste;
   }

   public E hentVerdi() {
       return verdi;
   }
}

public class Hovedprogram {
   public static void main(String[] args) {
       // 2.a)
       Node<String> forste = new Node<>("Hei");
       Node<String> node2 = new Node<>("paa");
       Node<String> node3 = new Node<>("deg!");

       forste.settNeste(node2);
       node2.settNeste(node3);

       // 2.b)
       Node<String> peker = forste;
       while (peker != null) {
           System.out.println(peker.hentVerdi());
           peker = peker.hentNeste();
       }
   }
}
```

3.)

```
class OrdnetPar<K, V> implements Par<K, V> {
   private K key;
   private V value;

   public OrdnetPar(K key, V value) {
       this.key = key;
       this.value = value;
   }

   @Override
   public K hentNokkel() {
       return key;
   }

   @Override
   public V hentVerdi() {
       return value;
   }
```

```
}
```

4.a)
Den tredje noden vil peke til en ny, fjerde node, som har verdien “!”

4.b)
Indeks 3 er den fjerde noden. Altså settes “planter” inn mellom “ikke” og “!”. Vi har nå listen “Jeg”, “liker”, “ikke”, “planter”, “!”.

4.c)
Etter å endre indeks 0 har vi listen “Du”, “liker”, “ikke”, “planter”, “!”.

4.c)
Indeks 2 er det tredje elementet. Altså fjernes “ikke”. Da har vi “Du”, “liker”, “planter”, “!”.

4.e)
```
Du
liker
planter
!
```