IN1010

Repetisjon Uke 8

14. mars, 2025

Gjennomgang konteeksamen fra i fjor

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    …
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
int taVare = tall[pos];
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    int taVare = tall[pos];
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    int taVare = tall[pos];
    …
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
tall[pos] = ny;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    int taVare = tall[pos];
    tall[pos] = ny;
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    int taVare = tall[pos];
    tall[pos] = ny;
    …
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
return taVare;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
taVare = pos;
```

```
int taVare = pos;
```

1.

Velg tre linjer med kode slik at denne metoden setter inn en *ny* verdi (gitt som tredje parameter) på posisjon *pos* (gitt som andre parameter) i arrayen *tall* (første parameter).

Metoden skal returnere det som sto i posisjon *pos* opprinnelig.

```
public static int settInnOgReturner (int[] tall, int pos, int ny) {
    int taVare = tall[pos];
    tall[pos] = ny;
    return taVare;
}
```

```
taVare = ny;
```

```
int taVare = tall;
```

```
int taVare;
```

```
return ny;
```

```
return tall[pos];
```

```
tall[pos] = taVare;
```

```
return pos;
```

```
tall[ny] = taVare;
```

```
taVare = pos;
```

```
int taVare = pos;
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = ▭▭▭;
    ▭▭▭;
    System.out.println("Brevets vekt: " + ▭▭▭;
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt
b2 = b1
b1 = b2
vekt = v
new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt
new Brev(19)
vekt
finnVekt(b2)
b2.finnVekt()
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = ▭▭▭;
    ▭▭▭;
    System.out.println("Brevets vekt: " + ▭▭▭;
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt
b2 = b1
b1 = b2
vekt = v
~~new Brev()~~
~~b2 = new Brev(18);~~
vekt = this.v
b2.vekt
b1.vekt
new Brev(19)
vekt
finnVekt(b2)
b2.finnVekt()
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    ▭▭▭;
    System.out.println("Brevets vekt: " + ▭▭▭;
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt
b2 = b1
b1 = b2
vekt = v
new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt

vekt
finnVekt(b2)
b2.finnVekt()
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    ▭▭▭;
    System.out.println("Brevets vekt: " + ▭▭▭;
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
~~neste.finnVekt()~~
b2
v = vekt
b2 = b1
b1 = b2
vekt = v
new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt

vekt
~~finnVekt(b2)~~
b2.finnVekt()
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    b2 = b1;
    System.out.println("Brevets vekt: " + b2.finnVekt();
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt

b1 = b2
vekt = v
new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt

vekt
finnVekt(b2)
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    b2 = b1;
    System.out.println("Brevets vekt: " + b2.finnVekt();
  }
}

class Brev {
  private int vekt;
  Brev(int v) { ▭▭▭; }
  int finnVekt () { return ▭▭▭; }
}
```

```
~~this.v = vekt~~
neste.finnVekt()
b2
~~v = vekt~~

b1 = b2
vekt = v
new Brev()
b2 = new Brev(18);
~~vekt = this.v~~
b2.vekt
b1.vekt

vekt
finnVekt(b2)
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    b2 = b1;
    System.out.println("Brevets vekt: " + b2.finnVekt();
  }
}

class Brev {
  private int vekt;
  Brev(int v) { vekt = v; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt

b1 = b2

new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt

vekt
finnVekt(b2)
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    b2 = b1;
    System.out.println("Brevets vekt: " + b2.finnVekt();
  }
}

class Brev {
  private int vekt;
  Brev(int v) { vekt = v; }
  int finnVekt () { return ▭▭▭; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt

b1 = b2

new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt

vekt
finnVekt(b2)
```

2.

Dette lille programmet skal opprette et objekt av klassen **Brev** og skrive ut dets vekt; svaret skal være **Brevets vekt: 19**.

Sett inn de kodebitene som mangler.

```
class EnkelProgrammering {
  public static void main (String[] args) {
    Brev b1, b2;
    b1 = new Brev(19);
    b2 = b1;
    System.out.println("Brevets vekt: " + b2.finnVekt();
  }
}

class Brev {
  private int vekt;
  Brev(int v) { vekt = v; }
  int finnVekt () { return vekt; }
}
```

```
this.v = vekt
neste.finnVekt()
b2
v = vekt

b1 = b2

new Brev()
b2 = new Brev(18);
vekt = this.v
b2.vekt
b1.vekt


finnVekt(b2)
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = ▭▭▭;
    int antallTall = ▭▭▭;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v    0    -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
new int[10]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = ▭▭▭;
    int antallTall = ▭▭▭;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v    0    -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
new int[10]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = ▭▭▭;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v    0    -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = ▭▭▭;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null    10    v    0    -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( ▭▭▭ ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println
antallTall < 10
tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = ▭▭▭;
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0
tastatur.nextInt()
System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( ▭▭▭ ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
v < 0
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      ▭▭▭ = ▭▭▭;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10    v          -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println

tallene[antallTall]
antallTall = antallTall - 1
v >= 0

System.out.println
~~tallene[0]~~
int i = 0; i < tallene.length;i++
~~tallene[i]~~
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println


antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      ▭▭▭;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1

antallTall = antallTall + 1
int i = antallTall-1; i >= 0; i--
tastatur.println


antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      antallTall = antallTall + 1;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1


int i = antallTall-1; i >= 0; i--
tastatur.println


antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      antallTall = antallTall + 1;
    } // end while-loop
    for ( ▭▭▭ ) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1


int i = antallTall-1; i >= 0; i--
tastatur.println


antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
~~int i = 0; i < tallene.length;i++~~
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      antallTall = antallTall + 1;
    } // end while-loop
    for ( int i = antallTall-1; i >= 0; i--) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1


tastatur.println


antallTall = antallTall - 1
v >= 0

System.out.println
tallene[0]
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      antallTall = antallTall + 1;
    } // end while-loop
    for ( int i = antallTall-1; i >= 0; i--) {
      ▭▭▭ ( ▭▭▭ );
    } // end for-loop
  }
}
```

```
null   10                -1


~~tastatur.println~~


antallTall = antallTall - 1
v >= 0

System.out.println
~~tallene[0]~~
int i = 0; i < tallene.length;i++
tallene[i]
```

3.

Dette lille programmet skal lese inntil 10 positive heltall. Et negativt tall signaliserer slutten på innlesingen hvis det er færre enn ti tall. Etterpå skal tallene skrives ut i omvendt rekkefølge.

```
import java.util.Scanner;
class SkrivTall {
  public static void main (String[] args) {
    int[] tallene = new int[10];
    int antallTall = 0;
    Scanner tastatur = new Scanner(System.in);
    while ( antallTall < 10 ) {
      int v = tastatur.nextInt();
      if ( v < 0 ) break;
      tallene[antallTall] = v;
      antallTall = antallTall + 1;
    } // end while-loop
    for ( int i = antallTall-1; i >= 0; i--) {
      System.out.println ( tallene[i] );
    } // end for-loop
  }
}
```

```
null   10                -1


tastatur.println


antallTall = antallTall - 1
v >= 0


tallene[0]
int i = 0; i < tallene.length;i++
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);
    linda.byttH(per);
    trine.byttH(linda);
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);
    linda.byttH(per);
    trine.byttH(linda);
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);
    linda.byttH(per);
    trine.byttH(linda);
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);
    linda.byttH(per);
    trine.byttH(linda);
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);
    linda.byttH(per);
    trine.byttH(linda);
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);   //ole-sl,anne-bl
    linda.byttH(per);  //linda-tr,per-sv
    trine.byttH(linda);//trine-tr,linda-sv
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4. Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);   //ole-sl,anne-bl
    linda.byttH(per);  //linda-tr,per-sv
    trine.byttH(linda);//trine-tr,linda-sv
    ole.skriv("Ole");
    per.skriv("Per");
    anne.skriv("Anne");
    linda.skriv("Linda");
    trine.skriv("Trine");
  }
}
```

4.

Hva blir skrevet ut?

```
class Hest {
  String navn;
  Hest (String n) { navn = n; }
}

class Hesteeier {
  Hest minH;
  public void leggTilH (Hest h) {
    minH = h;
  }
  public void byttH (Hesteeier andre) {
    Hest taVare = andre.minH;
    andre.minH = minH;
    minH = taVare;
  }
  public void skriv (String navn) {
    System.out.println(navn + ": " + minH.navn);
  }
}
```

```
class FinnHesteeiere {
  public static void main (String[] args) {
    Hest blakken = new Hest("Blakken");
    Hest trofast = new Hest("Trofast");
    Hest slitern = new Hest("Slitern");
    Hest svarten = new Hest("Svarten");
    Hesteeier ole = new Hesteeier();
    Hesteeier per = new Hesteeier();
    Hesteeier anne = new Hesteeier();
    Hesteeier linda = new Hesteeier();
    Hesteeier trine = new Hesteeier();
    ole.leggTilH(blakken);
    per.leggTilH(trofast);
    anne.leggTilH(slitern);
    linda.leggTilH(svarten);
    trine.leggTilH(svarten);
    ole.byttH(anne);   //ole-sl,anne-bl
    linda.byttH(per);  //linda-tr,per-sv
    trine.byttH(linda);//trine-tr,linda-sv
    ole.skriv("Ole");    // Slitern
    per.skriv("Per");    // Svarten
    anne.skriv("Anne");  // Blakken
    linda.skriv("Linda");// Svarten
    trine.skriv("Trine");// Trofast
  }
}
```

5. Ikke aktuell

Rekursjon

6.

Vi har en klasse Dobbeltliste som implementerer en dobbeltlinket liste. Den har i hvert fall disse tre metodene:

- **fjernFørste** fjerner første element i listen og returnerer det. (Vi antar at listen ikke blir tom etterpå.)
- **fjernSiste** fjerner siste element i listen og returnerer det. (Vi antar at listen ikke blir tom etterpå.)
- **hentNr** henter element nr *n* (nummerert fra 0) og returnerer det, men fjerner det ikke. Vi kan anta at elementet finnes.

Flytt de riktige kodebitene på plass. Bitene kan brukes mer enn én gang.

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = ▭▭▭;
    ▭▭▭ ;
    return ▭▭▭;
  }
  T fjernSiste () {
    Node resP = ▭▭▭;
    ▭▭▭;
    return ▭▭▭;
  }
  T hentNr (int n) {
    Node nodeP = ▭▭▭;
    for (int i = 0; i < n; ++i) {
      ▭▭▭;
    }
    return ▭▭▭;
  }
}
```

```
første.data
siste
nodeP.data
null
nodeP
første = første.neste
siste.data
resP
siste.forrige
første
resP = resP.neste;
siste = siste.forrige
nodeP = nodeP.neste
resP.data
første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = ▭▭▭;
    ▭▭▭ ;
    return ▭▭▭;
  }
  T fjernSiste () {
    Node resP = ▭▭▭;
    ▭▭▭;
    return ▭▭▭;
  }
  T hentNr (int n) {
    Node nodeP = ▭▭▭;
    for (int i = 0; i < n; ++i) {
      ▭▭▭;
    }
    return ▭▭▭;
  }
}
```

```
første.data
siste
nodeP.data
null
nodeP
første = første.neste
siste.data
resP
siste.forrige
første
resP = resP.neste;
siste = siste.forrige
nodeP = nodeP.neste
resP.data
første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = første;
    første = første.neste;
    return resP.data;
  }
  T fjernSiste () {
    Node resP = ▭▭▭;
    ▭▭▭;
    return ▭▭▭;
  }
  T hentNr (int n) {
    Node nodeP = ▭▭▭;
    for (int i = 0; i < n; ++i) {
      ▭▭▭;
    }
    return ▭▭▭;
  }
}
```

```
første.data
siste
nodeP.data
null
nodeP

siste.data
resP
siste.forrige

resP = resP.neste;
siste = siste.forrige
nodeP = nodeP.neste

første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = første;
    første = første.neste;
    return resP.data;
  }
  T fjernSiste () {
    Node resP = ▭▭▭;
    ▭▭▭;
    return ▭▭▭;
  }
  T hentNr (int n) {
    Node nodeP = ▭▭▭;
    for (int i = 0; i < n; ++i) {
      ▭▭▭;
    }
    return ▭▭▭;
  }
}
```

```
første.data
siste
nodeP.data
null
nodeP
første = første.neste
siste.data
resP
siste.forrige
første
resP = resP.neste;
siste = siste.forrige
nodeP = nodeP.neste
resP.data
første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = første;
    første = første.neste;
    return resP.data;
  }
  T fjernSiste () {
    Node resP = siste;
    siste = siste.forrige;
    return resP.data;
  }
  T hentNr (int n) {
    Node nodeP = □□□;
    for (int i = 0; i < n; ++i) {
      □□□;
    }
    return □□□;
  }
}
```

```
første.data

nodeP.data
null
nodeP
første = første.neste
siste.data
resP
siste.forrige
første
resP = resP.neste;

nodeP = nodeP.neste

første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = første;
    første = første.neste;
    return resP.data;
  }
  T fjernSiste () {
    Node resP = siste;
    siste = siste.forrige;
    return resP.data;
  }
  T hentNr (int n) {
    Node nodeP = □□□;
    for (int i = 0; i < n; ++i) {
      □□□;
    }
    return □□□;
  }
}
```

```
første.data
siste
nodeP.data
null
nodeP
første = første.neste
siste.data
resP
siste.forrige
første
resP = resP.neste;
siste = siste.forrige
nodeP = nodeP.neste
resP.data
første.neste
```

6.

```
class Dobbeltliste<T> {

  class Node {
    Node forrige, neste;
    T data;
  }
  Node første, siste;

  T fjernFørste () {
    Node resP = første;
    første = første.neste;
    return resP.data;
  }
  T fjernSiste () {
    Node resP = siste;
    siste = siste.forrige;
    return resP.data;
  }
  T hentNr (int n) {
    Node nodeP = første;
    for (int i = 0; i < n; ++i) {
      nodeP = nodeP.neste;
    }
    return nodeP.data;
  }
}
```

```
første.data
siste

null
nodeP
første = første.neste
siste.data
resP
siste.forrige

resP = resP.neste;
siste = siste.forrige

resP.data
første.neste
```

7.

Ikke aktuell

Rekursjon

8.

Hvilke fem tall skriver dette programmet ut?

```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```

8.

Hvilke fem tall skriver dette programmet ut?

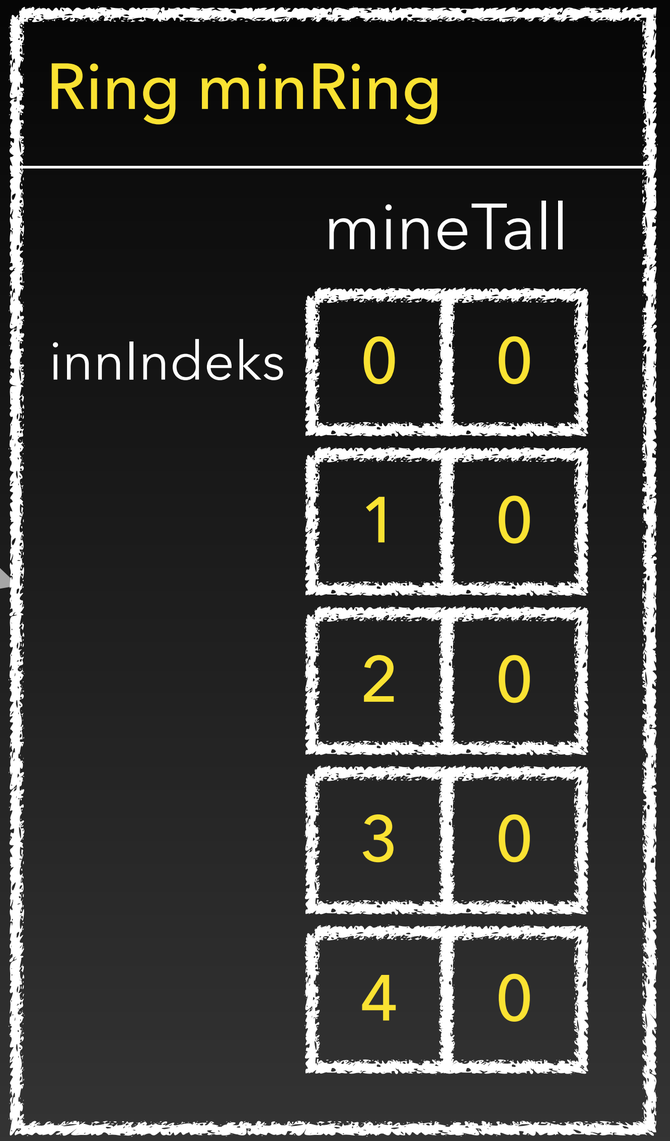
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

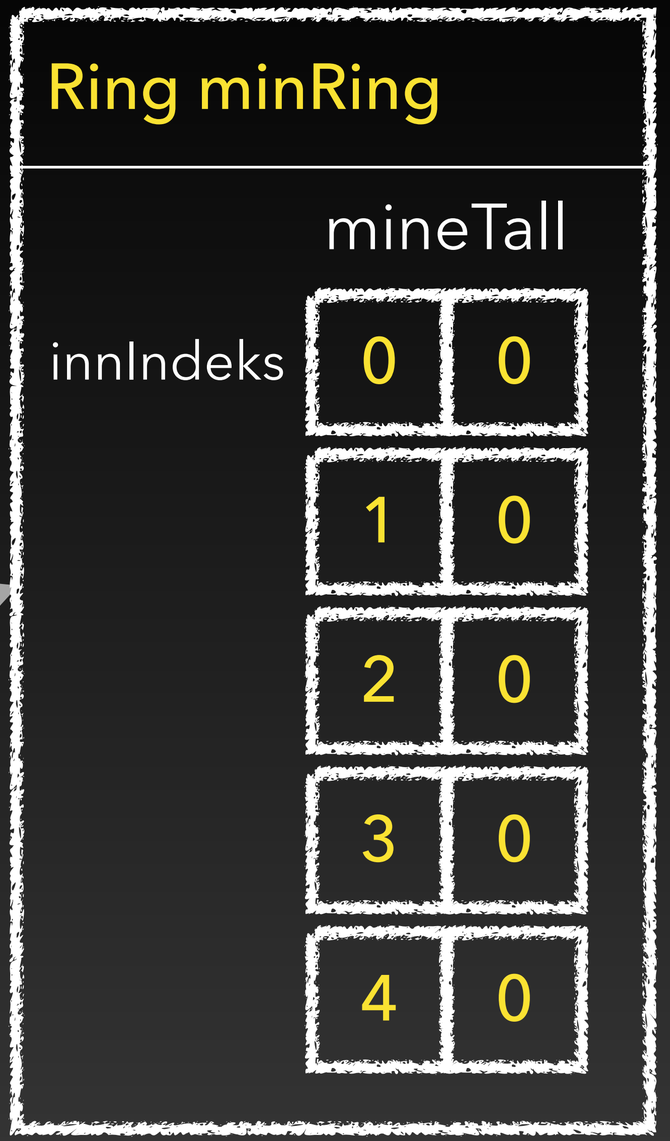
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

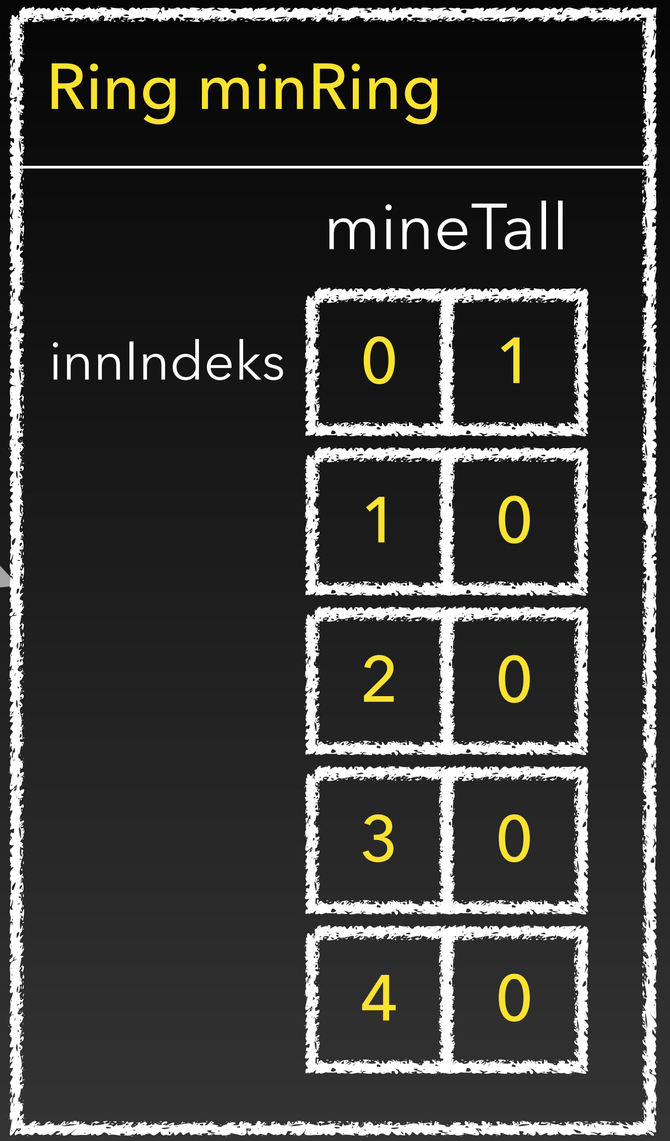
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

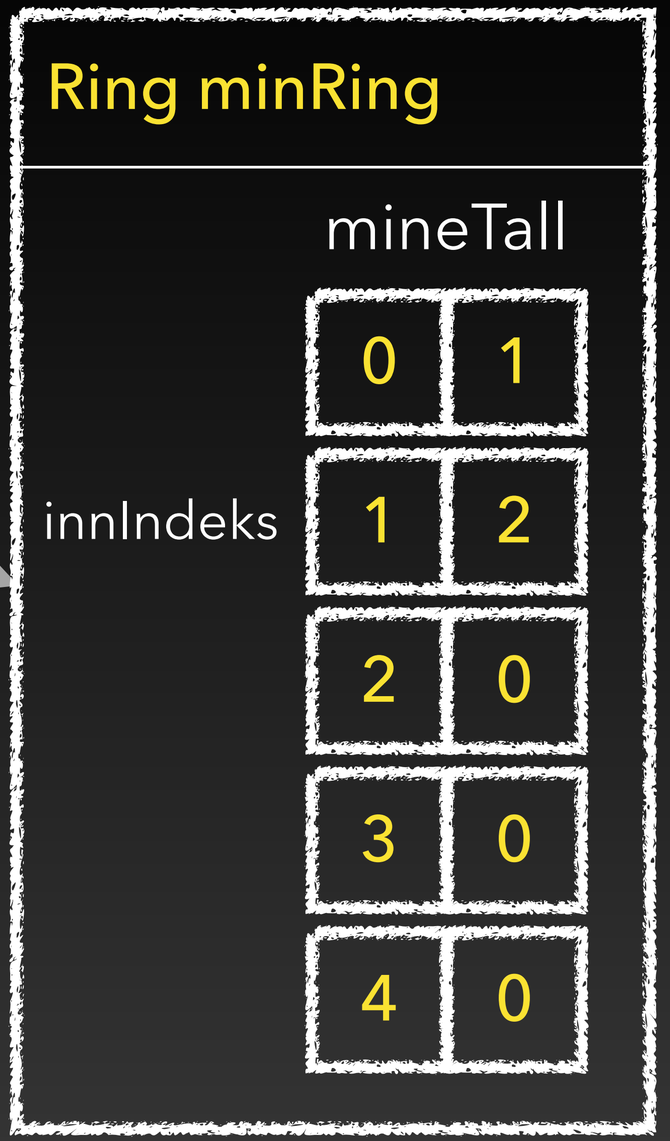
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

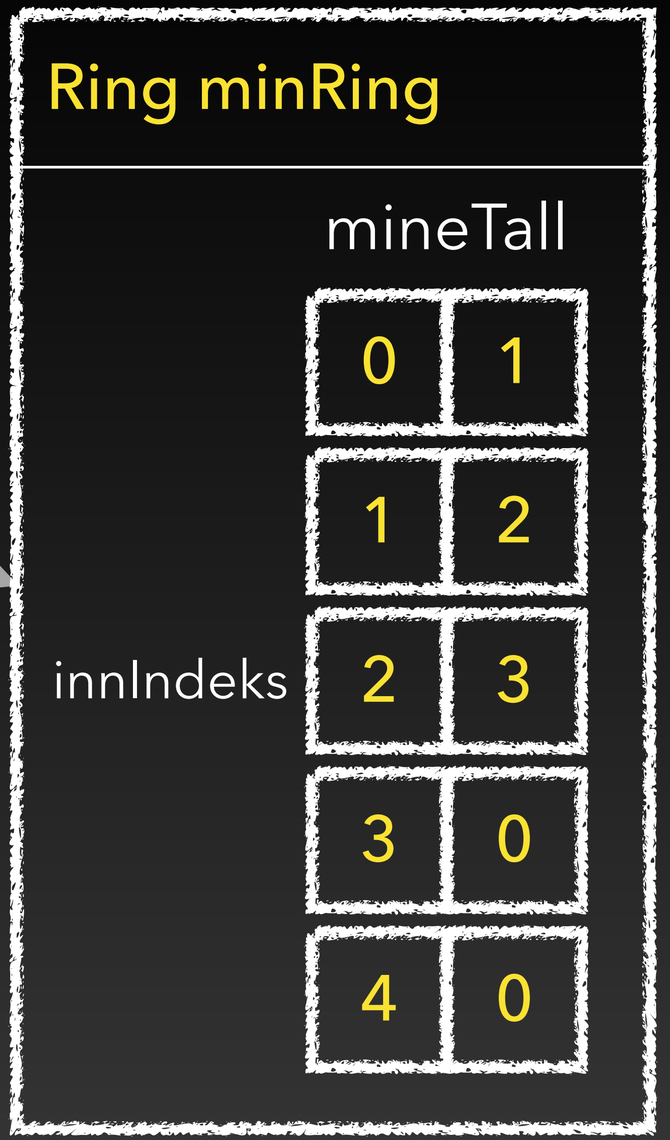
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

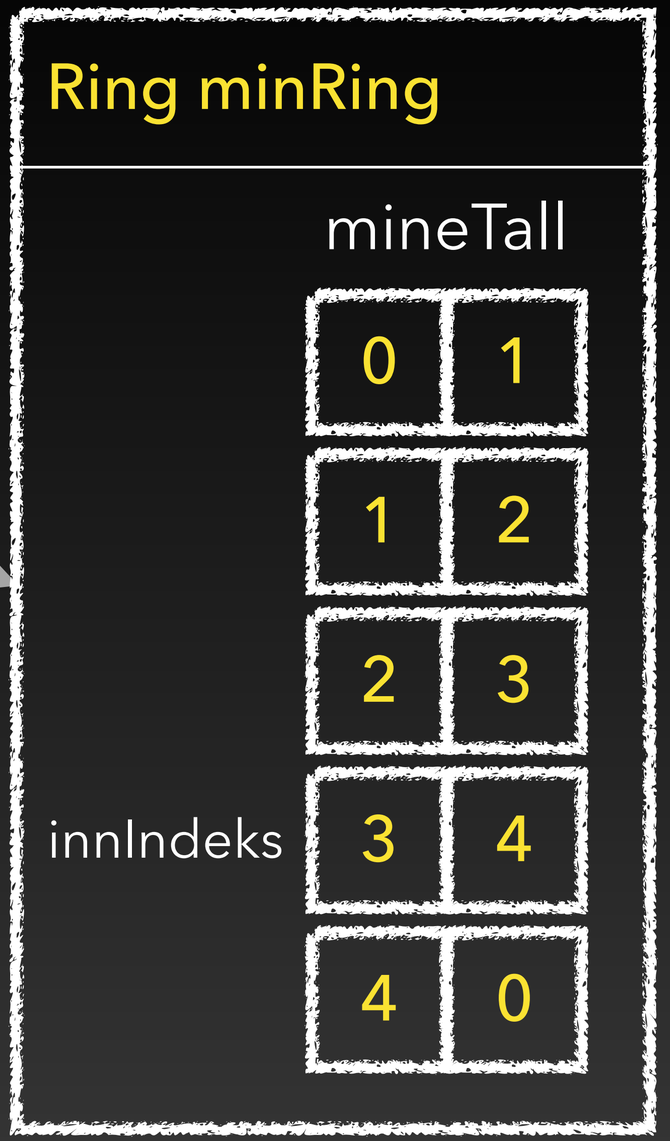
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

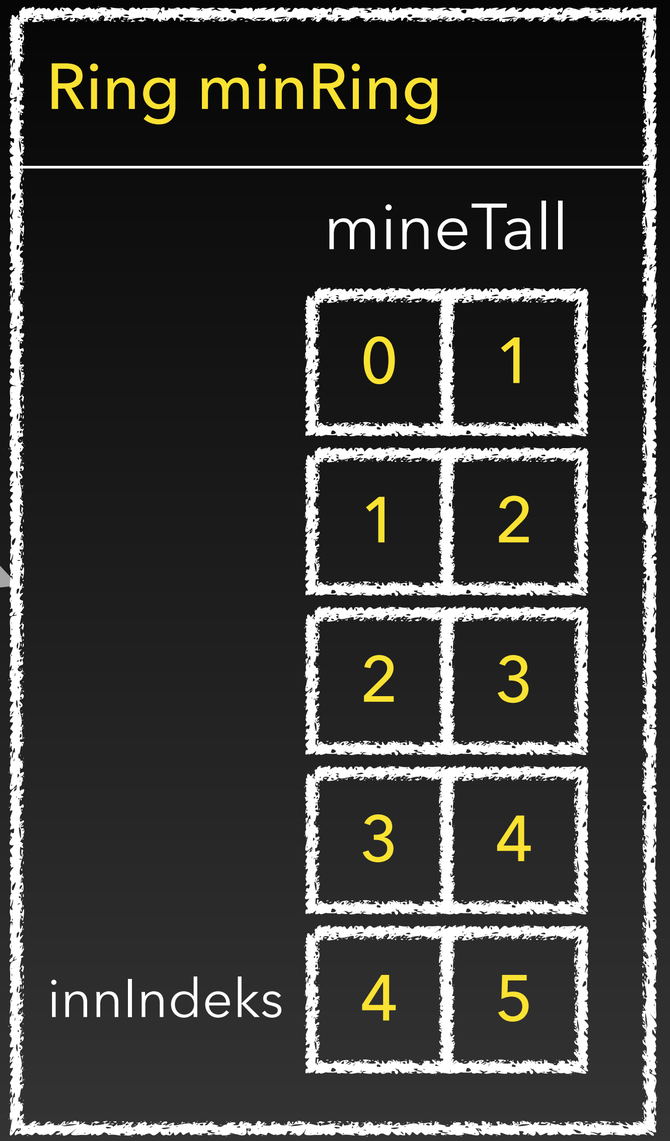
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

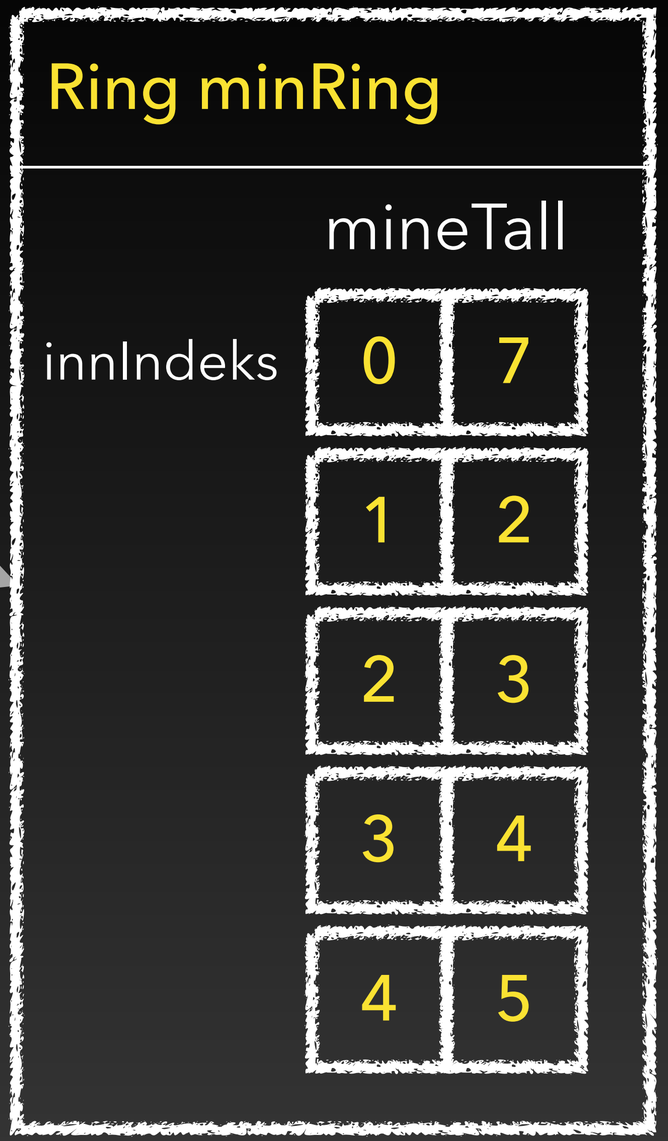
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

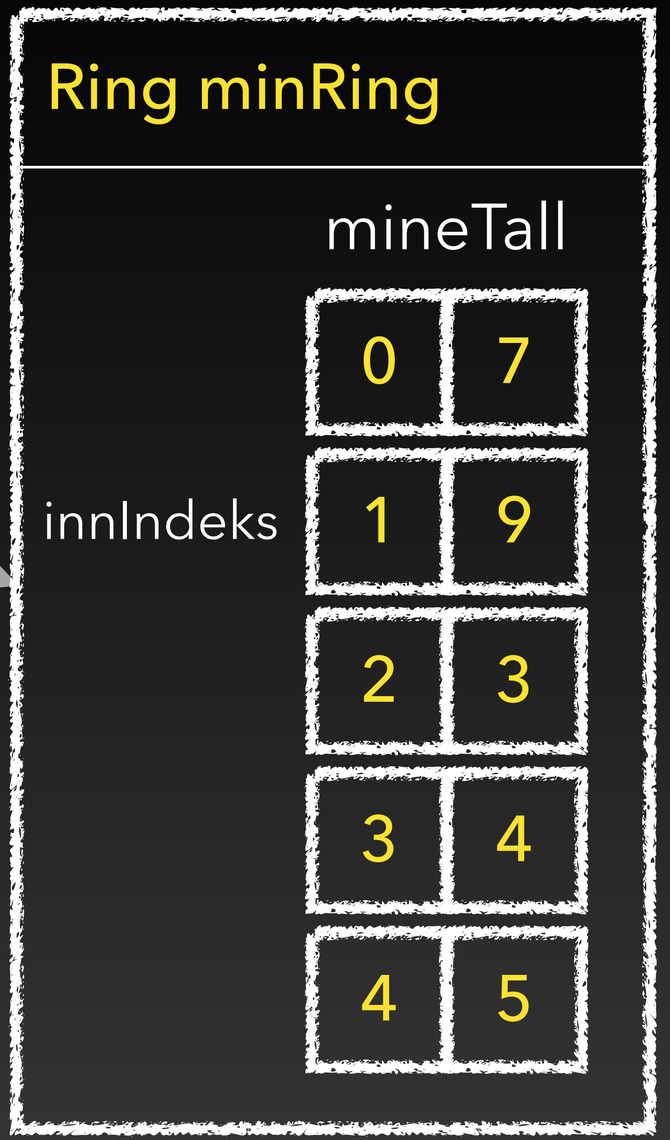
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

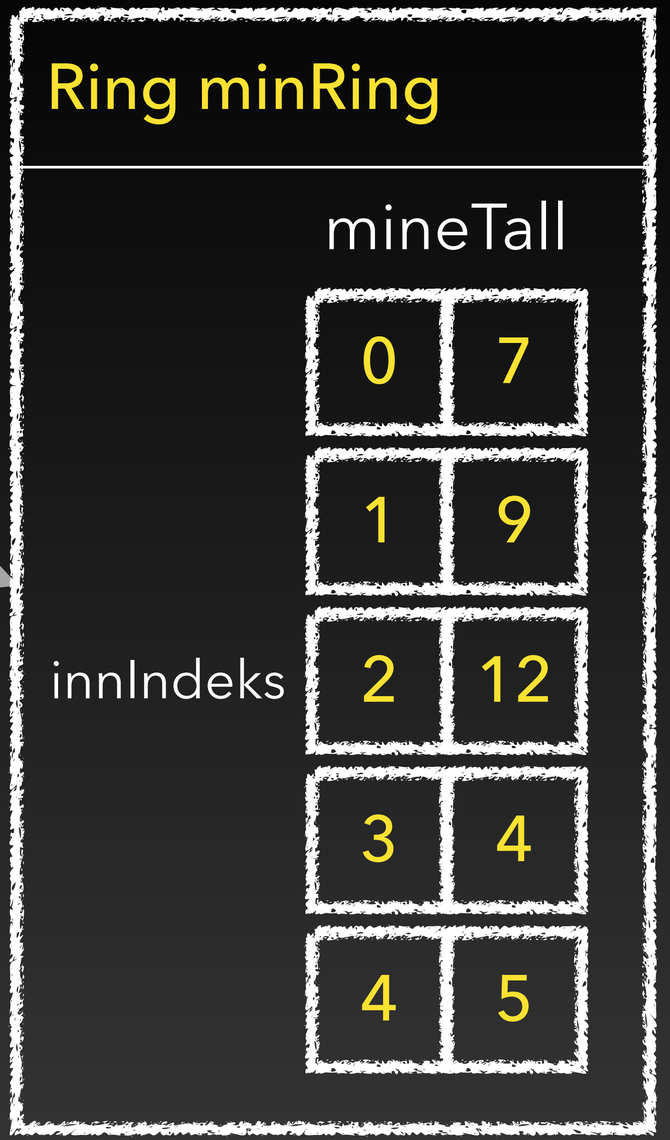
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

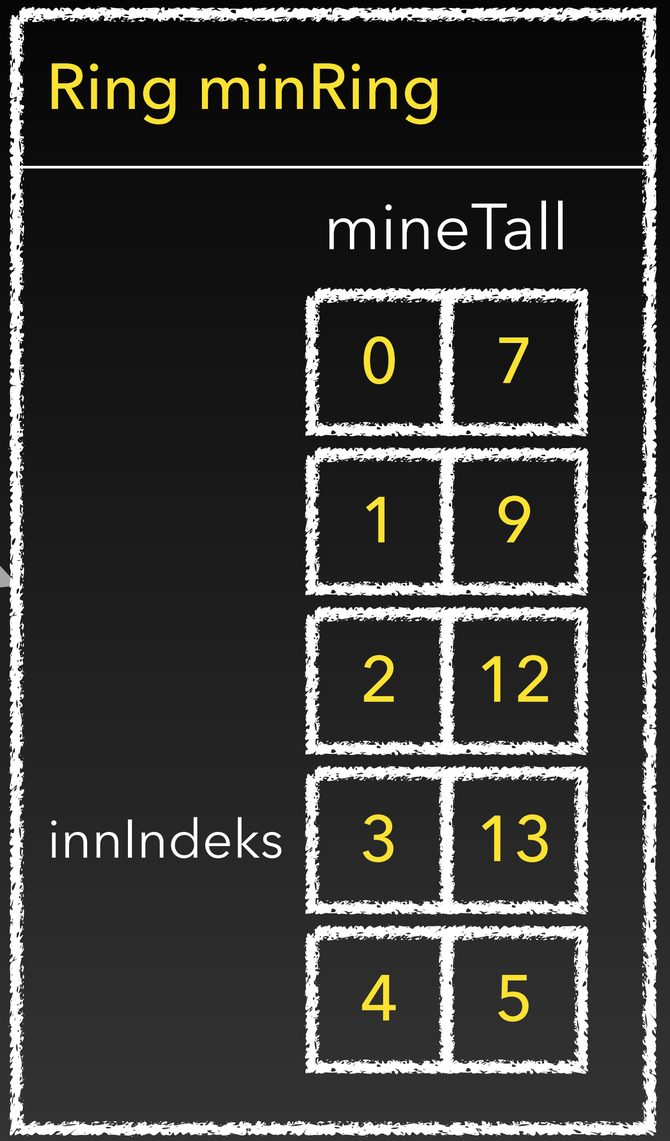
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

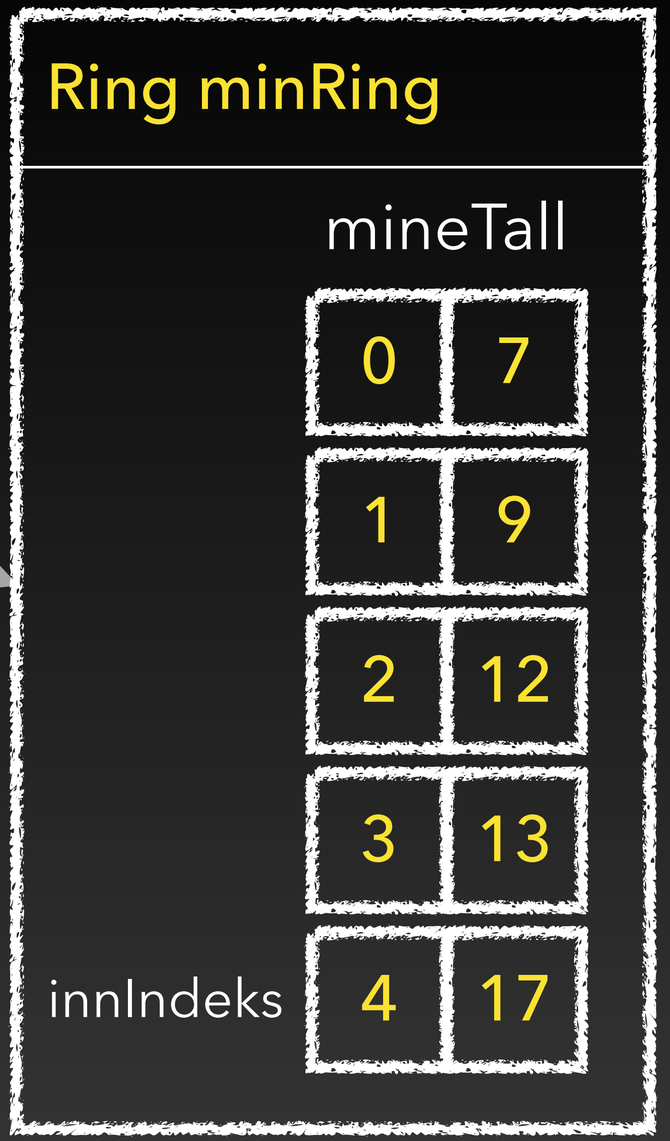
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

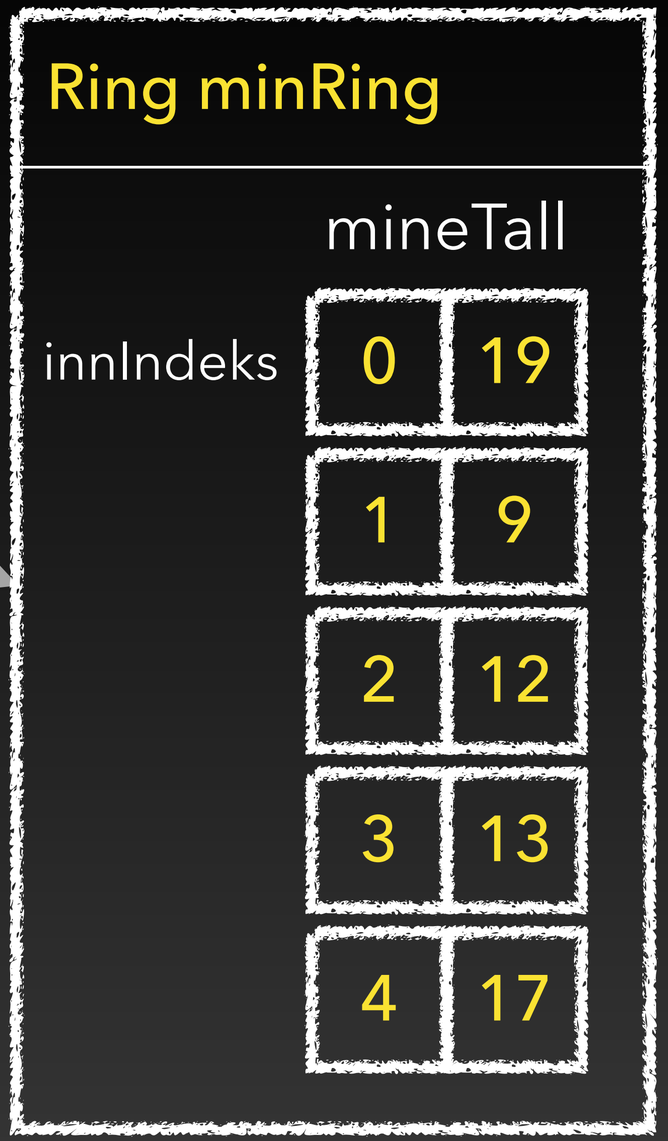
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

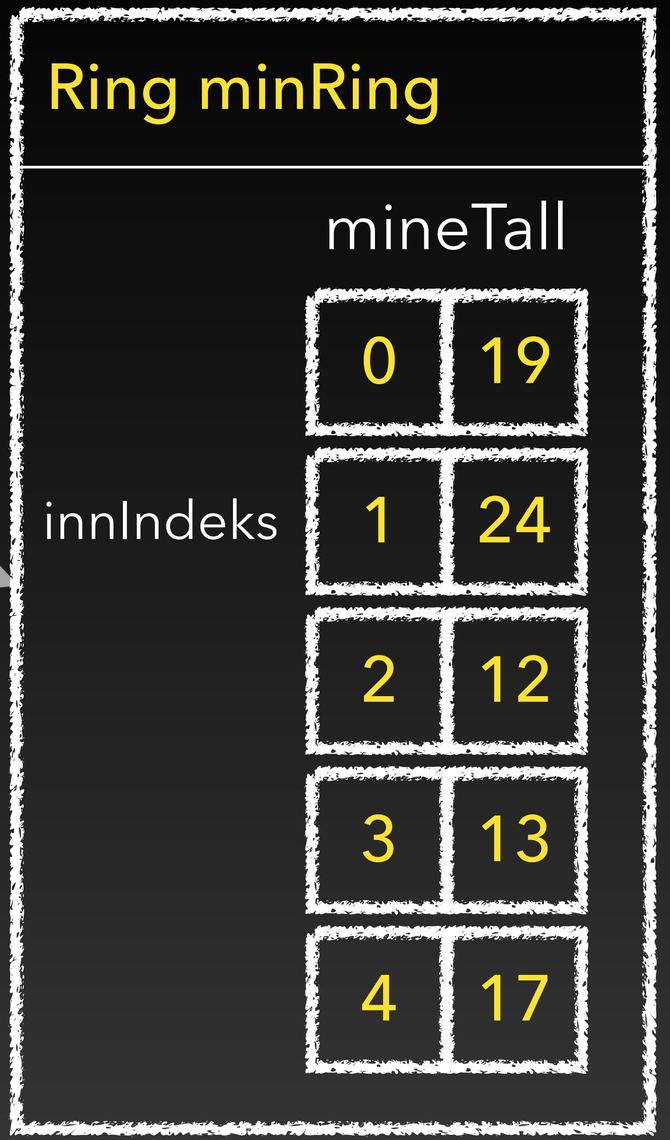
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

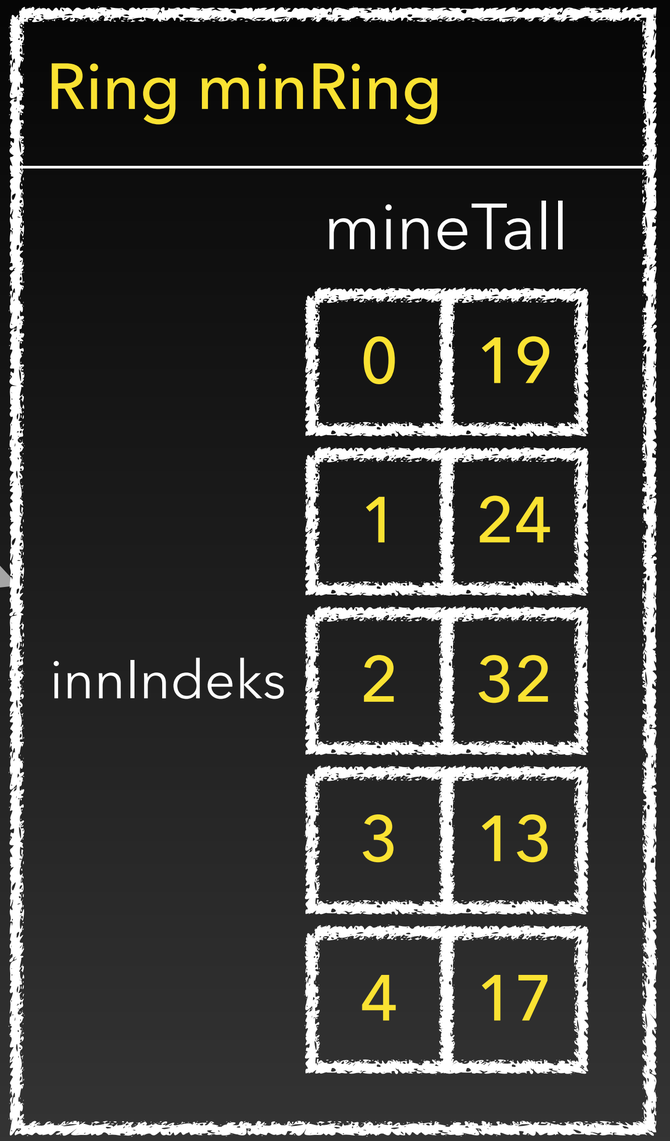
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



8.

Hvilke fem tall skriver dette programmet ut?

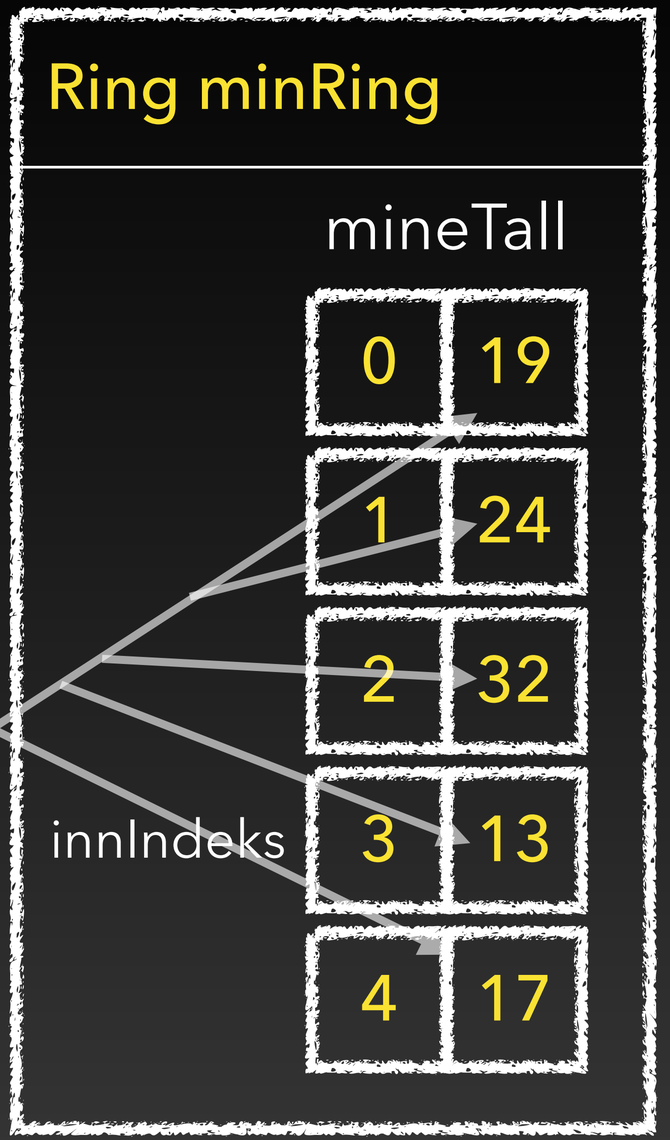
```
class RingBruk {
  public static void main(String[] a) {
    int[] tallene = { 1, 2, 3, 4, 5, 7, 9, 12, 13, 17, 19, 24, 32 };
    Ring minRing = new Ring(5);
    for (int tall: tallene) minRing.settInn(tall);
    minRing.dump();
  }
}

class Ring {
  int antall;
  int[] mineTall;

  Ring (int ant) {
    antall = ant; mineTall = new int[antall];
  }

  int innIndeks = 0;
  // Invariant: innIndeks peker på neste plass å sette inn
  public void settInn (int inn) {
    mineTall[innIndeks] = inn;
    innIndeks++;
    if (innIndeks == antall) innIndeks = 0;
  }

  void dump () {
    for (int ut: mineTall) System.out.println(ut);
  }
}
```



9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    ▭▭▭
    ▭▭▭
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
b.neste = null; b.forrige = null;

første == siste

siste = null;

b.forrige = siste; siste.neste = b;

b.forrige.neste = b.neste; b.neste.forrige = b.forrige;

b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    ▭▭▭
    ▭▭▭
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
b.neste = null; b.forrige = null;

første == siste

siste = null;

b.forrige = siste; siste.neste = b;

b.forrige.neste = b.neste; b.neste.forrige = b.forrige;

b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    ▭▭▭
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
b.neste = null; b.forrige = null;

første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    ▭▭▭
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
b.neste = null; b.forrige = null;

første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( ▭▭▭ ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;

første == null

første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;


første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      ▭▭▭
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;

første = b;

siste = b;


første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      første = b;
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;


siste = b;


første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      første = b;
    } else {
      ▭▭▭
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;

b.forrige = siste; siste.neste = b;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;


siste = b;


første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      første = b;
    } else {
      b.forrige = siste; siste.neste = b;
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;


b.neste.neste = b.neste; b.forrige.forrige = b.forrige;


siste = b;


første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      første = b;
    } else {
      b.forrige = siste; siste.neste = b;
    }
    ▭▭▭
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;
```

```
b.neste.neste = b.neste; b.forrige.forrige = b.forrige;
```

```
siste = b;
```

```
første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

9.

Vi har en klasse Bilkø som implementerer en dobbeltlenket liste av Biler. Flytt de riktige kodebitene på plass slik at de to metodene gjør det de sier de skal gjøre.

```
class Bilkø {
  Bil første, siste;
  // Invariant: Hvis både første og siste er null: køen er tom.
  // Hvis ikke, peker første på første bil og siste på siste bil i køen.

  void taUtInni (Bil b) {
    // Ta ut bilen b som verken er først eller sist i køen.
    b.forrige.neste = b.neste; b.neste.forrige = b.forrige;
    b.neste = null; b.forrige = null;
  }

  void settInn (Bil b) {
    // Setter bilen b inn sist i køen.
    if ( første == null ) {
      første = b;
    } else {
      b.forrige = siste; siste.neste = b;
    }
    siste = b;
  }
}
class Bil {
  Bil forrige, neste;
}
```

```
første == siste

siste = null;
```

```
b.neste.neste = b.neste; b.forrige.forrige = b.forrige;
```

```
første = siste

b.forrige = første; siste.neste = null;

b.neste = null; første.neste = null;
```

10.

Dette programmet skriver ut 5 verdier. Hvilke verdier skrives ut?

```
class TestBoeker {
    public static void main (String[] args) {
        Bok bok1 = new Bok("Snorres kongesagaer", 1000);
        System.out.println("Snorre internpris " + bok1.internpris());
        Bok bok2 = new Laerebok("Linux for dummies", 400);
        System.out.println("Linux internpris " + bok2.internpris());
        Bok bok3 = new IN1010bok("Big Java", 1200);
        System.out.println("Big Java " + bok3.internpris());
        Bok bok4 = new Roman("Krig og fred", 2000);
        System.out.println("Krig og fred " + bok4.internpris());
        Bok bok5 = new Krimbok("Sherlock Holmes", 240);
        System.out.println("Sherlock Holmes " + bok5.internpris());
    }
}

class Bok {
    String tittel;  double pris;
    Bok (String t, double p) { tittel = t;  pris = p; }
    double internpris () { return pris; }
}

class Laerebok extends Bok {
    Laerebok (String t, double p) { super(t, p); }
    @Override double internpris () { return pris * 0.8; }
}

class IN1010bok extends Laerebok {
    IN1010bok (String t, double p) { super(t,p); }
    @Override double internpris () { return super.internpris() * 0.5; }
}

class Roman extends Bok {
    Roman (String t, double p) { super(t, p); }
}

class Krimbok extends Roman {
    Krimbok (String t, double p) { super(t,p); }
    @Override double internpris () { return pris * 0.75; }
}
```

10.

Dette programmet skriver ut 5 verdier. Hvilke verdier skrives ut?

```
class TestBoeker {
    public static void main (String[] args) {
        Bok bok1 = new Bok("Snorres kongesagaer", 1000);
        System.out.println("Snorre internpris " + bok1.internpris());
        Bok bok2 = new Laerebok("Linux for dummies", 400);
        System.out.println("Linux internpris " + bok2.internpris());
        Bok bok3 = new IN1010bok("Big Java", 1200);
        System.out.println("Big Java " + bok3.internpris());
        Bok bok4 = new Roman("Krig og fred", 2000);
        System.out.println("Krig og fred " + bok4.internpris());
        Bok bok5 = new Krimbok("Sherlock Holmes", 240);
        System.out.println("Sherlock Holmes " + bok5.internpris());
    }
}

class Bok {
    String tittel;  double pris;
    Bok (String t, double p) { tittel = t;  pris = p; }
    double internpris () { return pris; }
}

class Laerebok extends Bok {
    Laerebok (String t, double p) { super(t, p); }
    @Override double internpris () { return pris * 0.8; }
}

class IN1010bok extends Laerebok {
    IN1010bok (String t, double p) { super(t,p); }
    @Override double internpris () { return super.internpris() * 0.5; }
}

class Roman extends Bok {
    Roman (String t, double p) { super(t, p); }
}

class Krimbok extends Roman {
    Krimbok (String t, double p) { super(t,p); }
    @Override double internpris () { return pris * 0.75; }
}
```

bok1: 1000

pris = p → 1000

bok2: 400

pris * 0.8 → 320

bok3: 1200

pris * 0.8 * 0.5 → 480

bok4: 2000

pris = p → 2000

bok5: 240

pris * 0.75 → 180

11.

Hvilke av disse setningene kan kompileres og kjøres uten feil og hvilke vil gi kompileringsfeil eller feile under kjøringen?

```java
class FlaskeBruk {
    public static void main (String[] a) {
        Drikkeflaske drikke = new Drikkeflaske();
        Plastflaske plast = new Plastflaske();
        Miljøplastflaske miljø = new Miljøplastflaske();
        Pappflaske papp = new Pappflaske();
        Glassflaske glass = new Glassflaske();
        PappflaskeMedPant pappPant = new PappflaskeMedPant();

        Flaske f1 = pappPant;
        Plastflaske pl1 = miljø;

        // Hvilke av disse setningene er gyldige som neste setning?
        Pant p1 = plast;
        Pant p2 = pappPant;
        Pant p3 = miljø;
        Pant p4 = (Pant)f1;
        Pant p5 = (Pant)pl1;
        Pant p6 = f1;
        Pant p7 = pl1;
        Papirsøppel ps1 = papp;
        Papirsøppel ps2 = pappPant;
        Papirsøppel ps3 = (Papirsøppel)f1;
        Papirsøppel ps4 = f1;
        Flaske f2 = drikke;
        Flaske f3 = pl1;
        Plastflaske pl2 = miljø;
        Pappflaske pf1 = f1;
    }
}

interface Pant {}
interface Papirsøppel {}

abstract class Flaske {}
class Drikkeflaske extends Flaske {}
class Plastflaske extends Flaske implements Pant {}
class Miljøplastflaske extends Plastflaske {}
class Pappflaske extends Flaske implements Papirsøppel {}
class Glassflaske extends Flaske {}
class PappflaskeMedPant extends Pappflaske implements Pant {}
```

11.

Hvilke av disse setningene kan kompileres og kjøres uten feil og hvilke vil gi kompileringsfeil eller feile under kjøringen?

```
class FlaskeBruk {
    public static void main (String[] a) {
        Drikkeflaske drikke = new Drikkeflaske();
        Plastflaske plast = new Plastflaske();
        Miljøplastflaske miljø = new Miljøplastflaske();
        Pappflaske papp = new Pappflaske();
        Glassflaske glass = new Glassflaske();
        PappflaskeMedPant pappPant = new PappflaskeMedPant();

        Flaske f1 = pappPant;
        Plastflaske pl1 = miljø;

        // Hvilke av disse setningene er gyldige som neste setning?
        Pant p1 = plast;
        Pant p2 = pappPant;
        Pant p3 = miljø;
        Pant p4 = (Pant)f1;
        Pant p5 = (Pant)pl1;
        Pant p6 = f1;
        Pant p7 = pl1;
        Papirsøppel ps1 = papp;
        Papirsøppel ps2 = pappPant;
        Papirsøppel ps3 = (Papirsøppel)f1;
        Papirsøppel ps4 = f1;
        Flaske f2 = drikke;
        Flaske f3 = pl1;
        Plastflaske pl2 = miljø;
        Pappflaske pf1 = f1;
    }
}

interface Pant {}
interface Papirsøppel {}

abstract class Flaske {}
class Drikkeflaske extends Flaske {}
class Plastflaske extends Flaske implements Pant {}
class Miljøplastflaske extends Plastflaske {}
class Pappflaske extends Flaske implements Papirsøppel {}
class Glassflaske extends Flaske {}
class PappflaskeMedPant extends Pappflaske implements Pant {}
```