Velkommen!

...

Johanna
johannph på mattermost
johannph@uio.no på mail!

Kort: Praktisk informasjon

- Labtime: 10 minutter med en gruppelærer en til en.
  - Her kan du få hjelp med obligen!
  - Mer info om det på emnesiden.
  - Lag drive (G Suite) bruker allerede nå :D
- Oblig 1, håper dere er godt i gang!
  - Tips: tegn datastrukturtegning
  - Da får dere får bedre forståelse for programmet og dere unngår feil

Repetisjon forrige uke

Private/Public

- Hvis man ikke skriver private foran class er den public
- Alt som er public kan aksesseres utenfor klassen
- Konstruktøren må være public for at man skal kunne lage instanser av klassen utenfor klassen
- God skikk at variablene er private, og at man endrer og henter dem ved hjelp av metoder (hent og sett)

```
class Person{
  private int alder;
  private String navn;

  public Person(int alder, String navn){
      this.alder = alder;
      this.navn = navn;
  }
  public String hentNavn(){
    return navn;
  }
  public void settNavn(String navn){
    this.navn = navn;
  }
```

Static

- Lager en klasse Fjellrev med en static variabel rodlistet
- Navn og alder er ikke static.
- Hvis en art er rødlistet gjelder det for alle individer av arten: klassevariabel
  - Derfor er rødlistet static, den gjelder for alle instanser av klassen!
- Men de har ulike navn og ulik alder: instansvariabler
  - Derfor er navn og alder ikke static, de er ikke det samme for alle instanser!
- Oppretter to fjellrever
- Endrer fjellrev1.rodlistet til false, da blir også fjellrev2.rodlistet false!

```java
class Fjellrev{
  private static Boolean rodlistet = true;
  private String navn;
  private int alder;

  public Fjellrev(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public void setRodlistet(Boolean rodlistet){
    this.rodlistet = rodlistet;
  }
  public Boolean hentRodlistet(){
    return rodlistet;
  }
}
```

```java
class TestFjellrev{
  public static void main(String[] args) {
    Fjellrev fjellrev1 = new Fjellrev("Bjarne", 5);
    Fjellrev fjellrev2 = new Fjellrev("Alfred", 10);
    fjellrev1.setRodlistet(false);
    System.out.println(fjellrev2.hentRodlistet());
  }
}
```

Output: >> false

Static

Hvis en metode er static trenger vi ikke et objekt for å kalle på metoden

```java
class Fjellrev{
  private static Boolean rodlistet = true;
  private String navn;
  private int alder;
  private Fjellrev bestevenn;

  public Fjellrev(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public static void setRodlistet(Boolean rodlistetParameter){
    rodlistet = rodlistetParameter;
  }
  public void settnavn(String navn){
    this.navn = navn
  }
```

```java
class TestFjellrev{
  public static void main(String[] args) {
    Fjellrev fjellrev1 = new Fjellrev("Bjarne", 5);
    Fjellrev fjellrev2 = new Fjellrev("Alfred", 10);
    Fjellrev.setRodlistet(false);
    fjellrev1.settnavn("Bjarne")
  }
}
```

Returtype

- I java må hver metode ha en returtype!
- Void betyr at metoden ikke returnerer noe
- Alle datatyper og klasser kan være returtype
- Her er eksempler med Boolean, String, int og klassen Fjellrev
- Returtype matcher typen til det vi skal returnere!
  - f.eks. bestevenn er et objekt av klassen Fjellrev, se linje 5 i koden (der bestevenn er deklarert)!

```
1  class Fjellrev{
2    private static Boolean rodlistet = true;
3    private String navn;
4    private int alder;
5    private Fjellrev bestevenn;
6
7    public Fjellrev(String navn, int alder){
8      this.navn = navn;
9      this.alder = alder;
10   }
11   public void setRodlistet(Boolean rodlistet){
12     this.rodlistet = rodlistet;
13   }
14   public Boolean hentRodlistet(){
15     return rodlistet;
16   }
17   public String hentNavn(){
18     return navn;
19   }
20   public int hentAlder(){
21     return alder;
22   }
23   public Fjellrev hentBestevenn(){
24     return bestevenn;
25   }
26 }
```

This

Ordet “this” brukes i Java dersom det er en lokal variabel med samme navn som en instansvariabel

Alternativt kan man bare gi de lokale variablene andre navn, da trenger man ikke bruke this

```
1  class Fjellrev{
2    private static Boolean rodlistet = true;
3    private String navn;
4    private int alder;
5    private Fjellrev bestevenn;
6
7    public Fjellrev(String navn, int alder){
8      this.navn = navn;
9      this.alder = alder;
10   }
11   public void setRodlistet(Boolean rodlistet){
12     this.rodlistet = rodlistet;
13   }
```

Instansvariabel

Lokal variabel med samme navn

```
1  class Fjellrev{
2    private static Boolean rodlistet = true;
3    private String navn;
4    private int alder;
5    private Fjellrev bestevenn;
6
7    public Fjellrev(String navnParameter, int alderParameter){
8      navn = navnParameter;
9      alder = alderParameter;
10   }
11   public void setRodlistet(Boolean rodlistetParameter){
12     rodlistet = rodlistetParameter;
13   }
```

I/O - Lese fra terminal med Scanner

- Bruker **Scanner**:
  - Dokummentasjon: https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
  - NB: Må importeres med: import java.util.Scanner
- Sender inn **System.in** i konstruktøren
  - Det forteller Scanner at den skal "lytte til terminalen

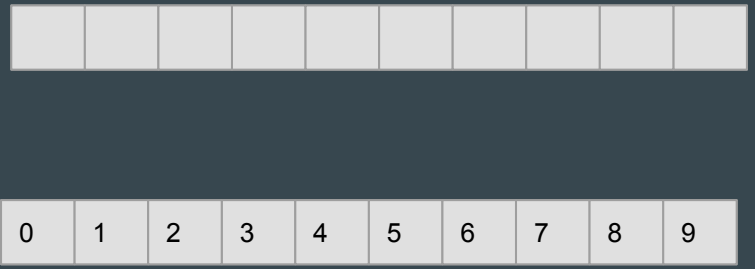
```java
import java.util.Scanner;
class LeseFraTerminal{

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Skriv noe?");
        String linjeFraTerminal = scanner.nextLine();

        System.out.println("Skriv inn en int?");
        String tallFraTerminalString = scanner.nextLine();

        int tallFraTerminal = Integer.parseInt(tallFraTerminalString);
        System.out.println("Linja du skrev: " + linjeFraTerminal);
        System.out.println("Tallet du skrev ganget med 2: " + (tallFraTerminal * 2));


    }
}
```

- Nyttige metoder:
  - nextLine(): henter den neste Stringen (til du trykker enter i terminalen)
  - next() henter neste ord til mellomrom som default, brukes mest når man leser fra fil
  - nextInt() henter neste tall, brukes mest når man leser fra fil og ikke terminal

Array

Viktig å huske at array i java er fixed size, dette er annerledes enn liste i python.
Vi kan endre innhold, men ikke lengden.

```java
class ArrayEksempel{

    public static void main(String[] args){
        Katt[] kattArray = new Katt[10];

        int[] tallArray = {0,1,2,3,4,5,6,7,8,9};

        System.out.println(tallArray[2]); //printer ut 2

        tallArray[2] = 20;
        System.out.println(tallArray[2]); //printer ut 20

        kattArray[0] = new Katt("Pus", 1);
    }
}
```



Array

Viktig å huske at array i java er fixed size, dette er annerledes enn liste i python.
Vi kan endre innhold, men ikke lengden.

```java
class ArrayEksempel{

    public static void main(String[] args){
        Katt[] kattArray = new Katt[10];

        int[] tallArray = {0,1,2,3,4,5,6,7,8,9};

        System.out.println(tallArray[2]); //printer ut 2

        tallArray[2] = 20;
        System.out.println(tallArray[2]); //printer ut 20

        kattArray[0] = new Katt("Pus", 1);
    }
}
```

NullPointer

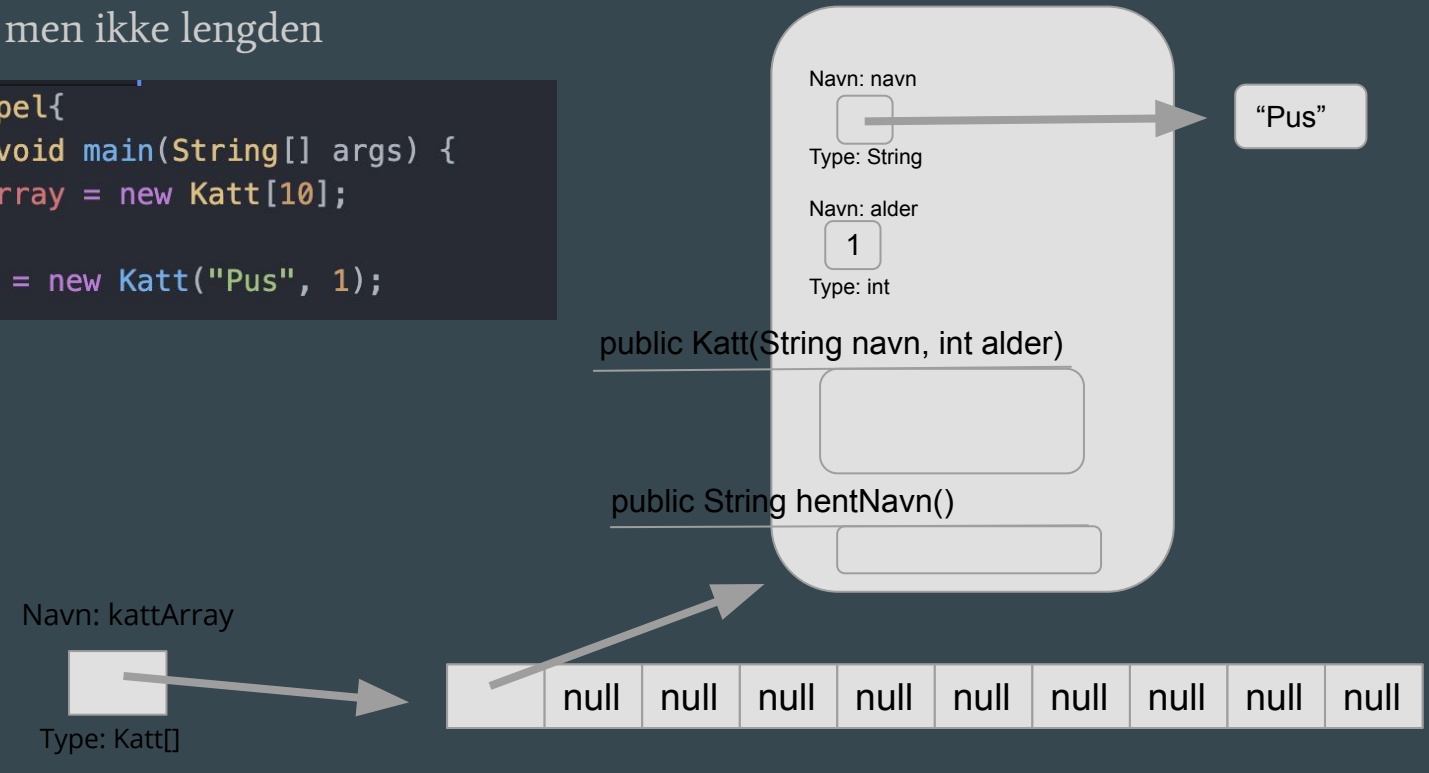
Viktig å huske at array i java er fixed size, dette er annerledes enn liste i python, vi kan endre innhold, men ikke lengden

```java
class ArrayEksempel{
  public static void main(String[] args) {
    Katt[] kattArray = new Katt[10];
```

NullPointer

Viktig å huske at array i java er fixed size, dette er annerledes enn liste i python, vi kan endre innhold, men ikke lengden

```java
class ArrayEksempel{
  public static void main(String[] args) {
    Katt[] kattArray = new Katt[10];

    kattArray[0] = new Katt("Pus", 1);
```



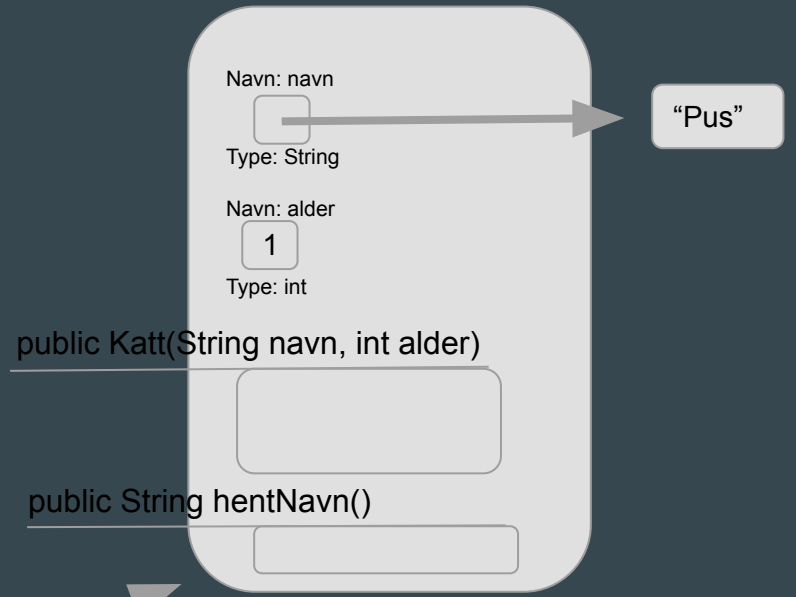
NullPointerException

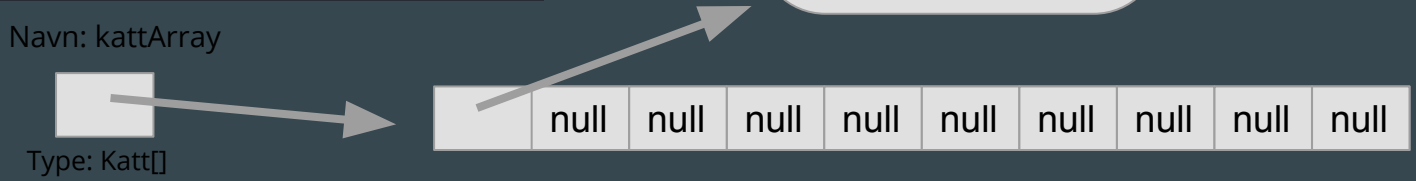
Viktig å huske at array i java er fixed size, dette er annerledes enn liste i python, vi kan endre innhold, men ikke lengden

```
1  class ArrayEksempel{
2    public static void main(String[] args) {
3      Katt[] kattArray = new Katt[10];
4
5      kattArray[0] = new Katt("Pus", 1);
6
7      kattArray[0].hentNavn()
8
9      kattArray[1].hentNavn()
10   }
11 }
```

Evaluerer til "Pus"

null.hentNavn()
Gir error:
java.lang.NullPointerException





Repetisjon denne uken

I dag

Deklarert vs initialisert

Primitive datatyper

Immutable vs mutable

Referanser

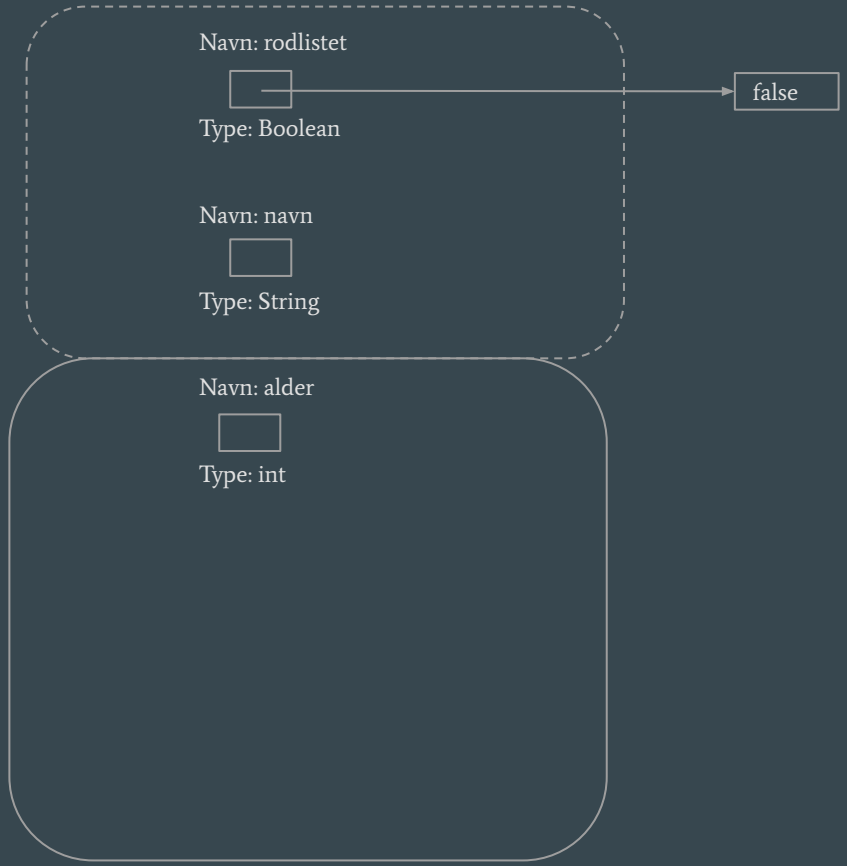
ArrayList

Hashmap

Datastrukturtegning

Error-hpndtering

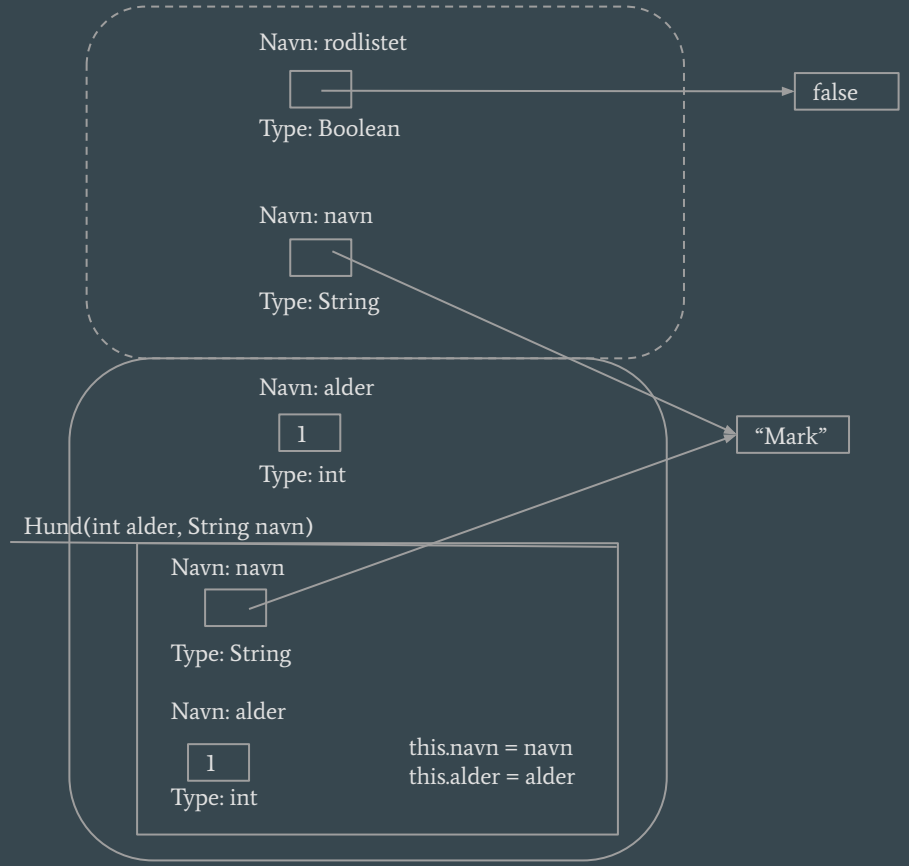
Deklarert vs. initialisert




```java
class Hund{
  public static Boolean rodlistet = false;
  public static String navn;
  private int alder;
```

Her blir alle instansvariablene deklarert, men det er bare instansvariabelen rodlistet som er initialisert!

Deklarert vs. initialisert




```java
class Hund{
  public static Boolean rodlistet = false;
  public static String navn;
  private int alder;

  public Hund(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }

  public String skrivUt(){
    System.out.println(navn);
  }
}
```

Først i konstruktøren (init-metoden) blir også navn og alder initialisert!

Legg merke til at to lokale variabler (som ikke er instansvariabler) også blir opprettet.

Primitive data

Primitive data i Java

- int
- char
- float
- boolean (liten b)

Ikke-primitive data

- String
- Array
- Boolean (stor b)
- Alle klasser

Primitive data

Navn: variabel1

20

Type: int

```
17  variabel1 = 20
18  variabel2 = variabel1
19  // Det over er akkuratt det samme som å si: variabel2 = 20
20  variabel2 = 25
```

Primitive data

Navn: variabel1

20

Type: int

```
17  variabel1 = 20
18  variabel2 = variabel1
19  // Det over er akkuratt det samme som å si: variabel2 = 20
20  variabel2 = 25
```

Primitive data

Navn: variabel1

20

Type: int

Navn: variabel2

20

Type: int

```
17  variabel1 = 20
18  variabel2 = variabel1
19  // Det over er akkuratt det samme som å si: variabel2 = 20
20  variabel2 = 25
```

Primitive data

Navn: variabel1

20

Type: int

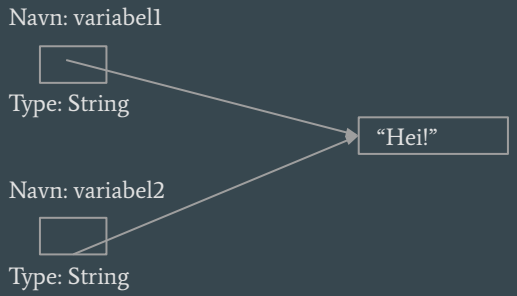
Navn: variabel2

25

Type: int

```
17  variabel1 = 20
18  variabel2 = variabel1
19  // Det over er akkuratt det samme som å si: variabel2 = 20
20  variabel2 = 25
```

Primitive data

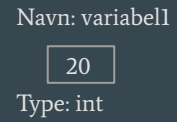



```
variabel1 = "Hei!"
variabel2 = variabel1
```

Primitive-data i datastrukturtegning

Primitive data kan vi bare skrive inni boksen i tegningen.

Ikke-primitive data tegnes med pil og en separat boks.

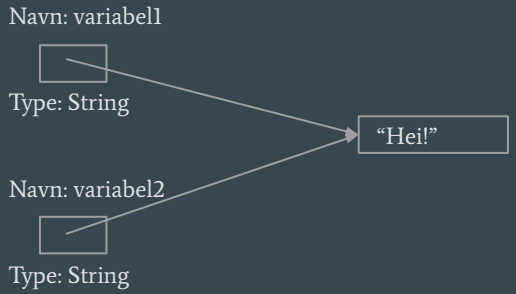


immutble vs. mutable

Strenger og tall er immutable, de kan ikke endres! Kanskje det ser ut som vi endre en string eller et tall, men egentlig lager vi en ny!

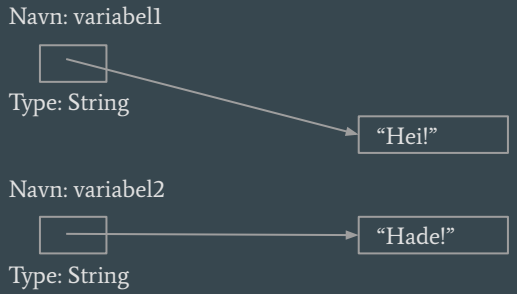
Arrays, dictionaries og mange andre objekter er derimot mutable, altså vi kan endre dem!

Stringer er ikke-primitiv datatype, men de er immutable!



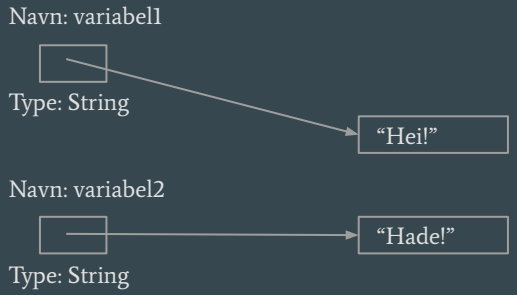

```
variabel1 = "Hei!"
variabel2 = variabel1
```

Stringer er ikke-primitiv datatype, men de er immutable!

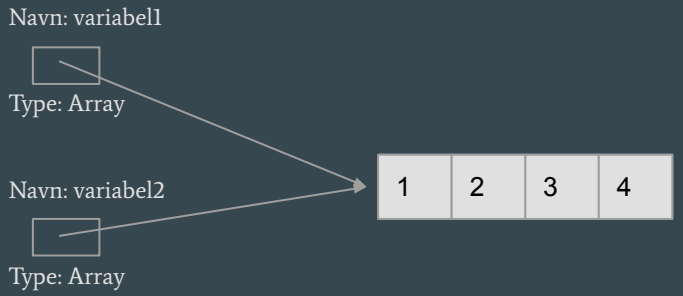



```
variabel1 = "Hei!"
variabel2 = variabel1
variabel2 = "Hade"
```

Arrays derimot er mutable!

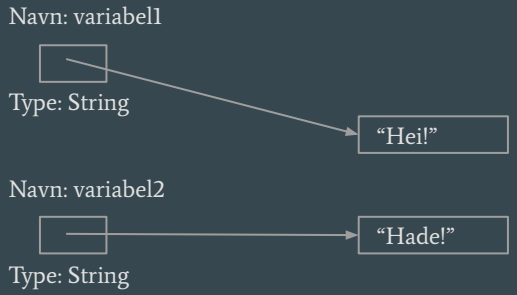



```
variabel1 = "Hei!"
variabel2 = variabel1
variabel2 = "Hade"
```

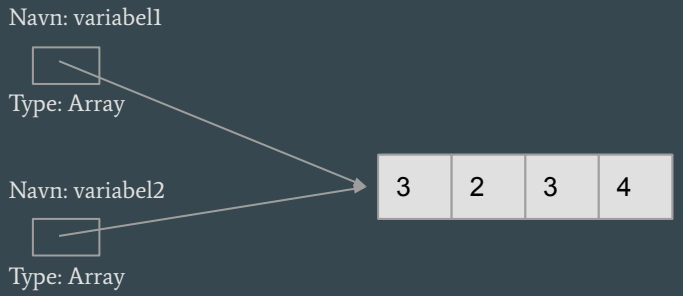



```
int[] variabel1 = {1, 2, 3, 4};
int[] variabel2 = variabel1;
```

Arrays derimot er mutable!



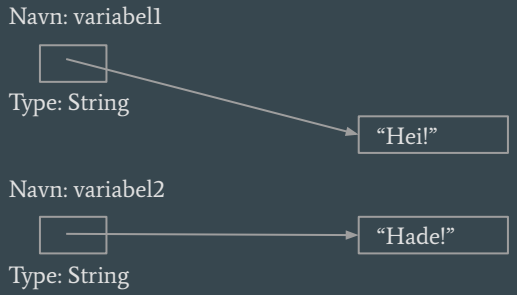

```
variabel1 = "Hei!"
variabel2 = variabel1
variabel2 = "Hade"
```



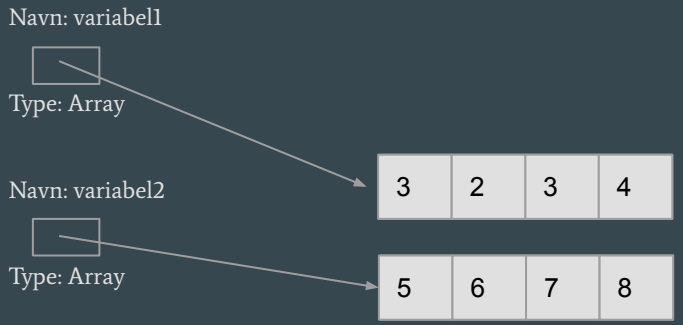

```
int[] variabel1 = {1, 2, 3, 4};
int[] variabel2 = variabel1;
variabel2[0] = 3;
```

Vi endrer det variabel1 peker på selv om det i koden kanskje kan se ut som vi endrer bare endrer det variabel vairabel2 peker på!!

Arrays derimot er mutable!




```
variabel1 = "Hei!"
variabel2 = variabel1
variabel2 = "Hade"
```

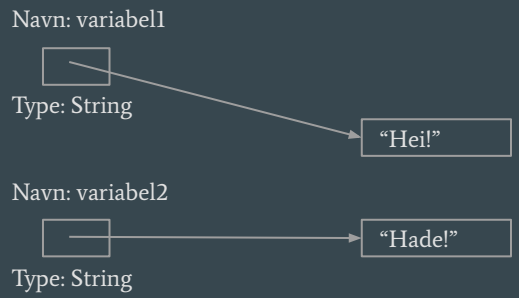



```
int[] variabel1 = {1, 2, 3, 4};
int[] variabel2 = variabel1;
variabel2[0] = 3;
// >> {3, 2, 3, 4}
variabel2 = {5, 6, 7, 8};
```

Men vi kan selvfølgelig også endre det variabel2 peker på uten å endre det variabel1 peker på slik vi gjorde med stringer

Referanser

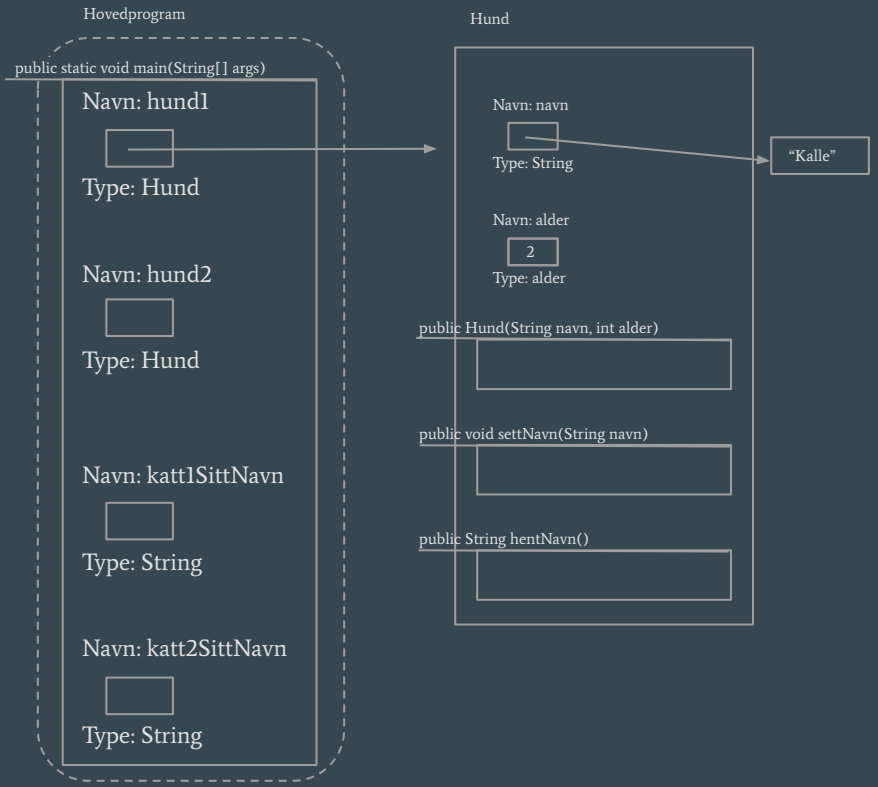
Når vi tegner ikke-primitiv data i datastrukturtegningen så tegner vi altså med en pil. Den pilen symboliserer en referanse! Egentlig er en referansen en adresse i minnet på datamaskinen.



Så i variabel1 ligger det nå en referanse til stringen “Hei!” og i variabel2 ligger det en referanse til stringen “Hade”.

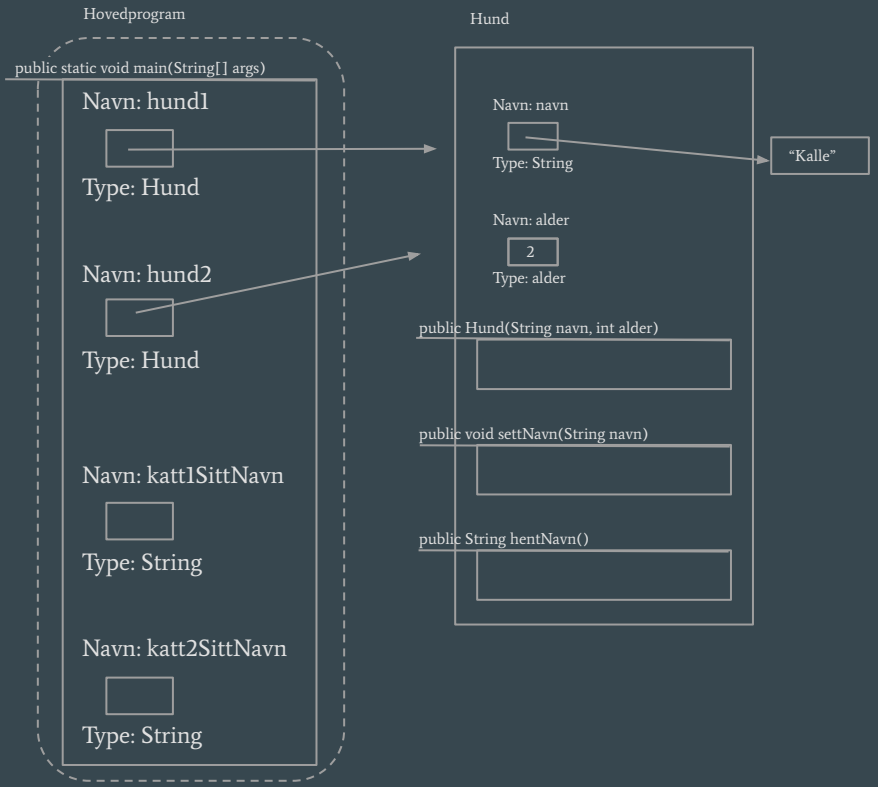
På samme måte kan variabler ha referanser til objekter av klasser som vi selv lager, f.eks. Hund!

Referanser



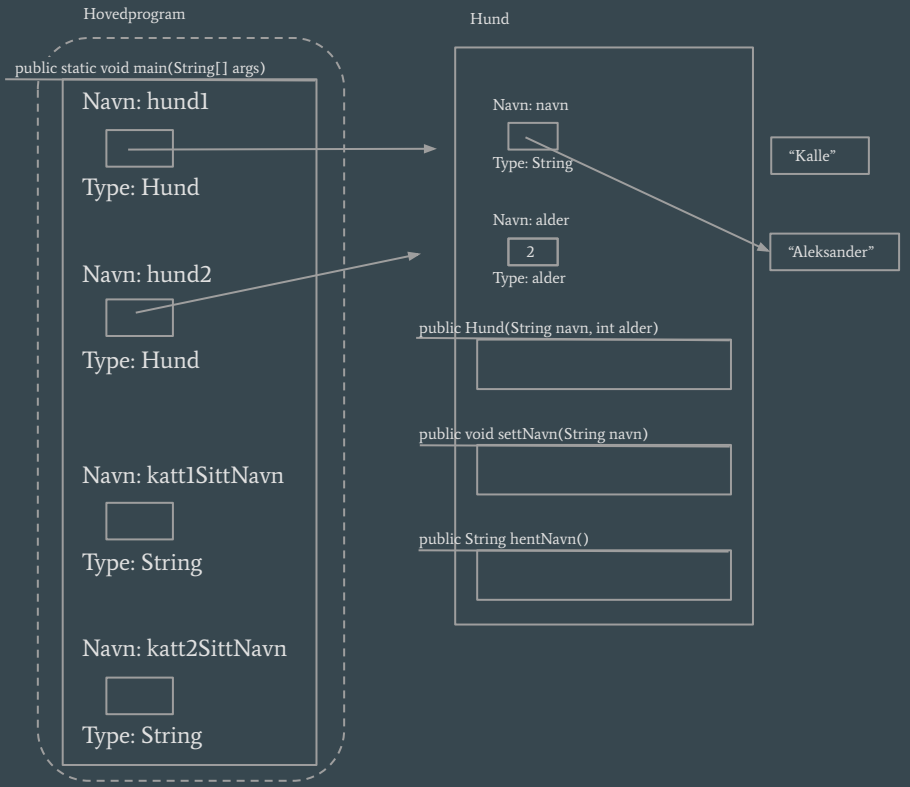

```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Referanser



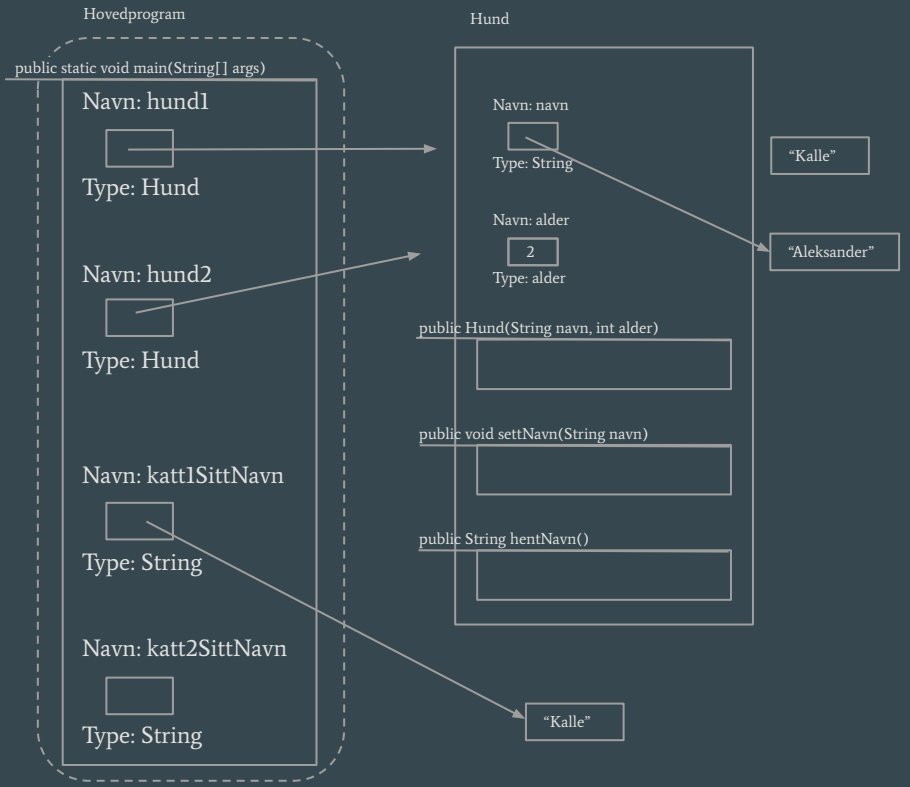

```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Referanser



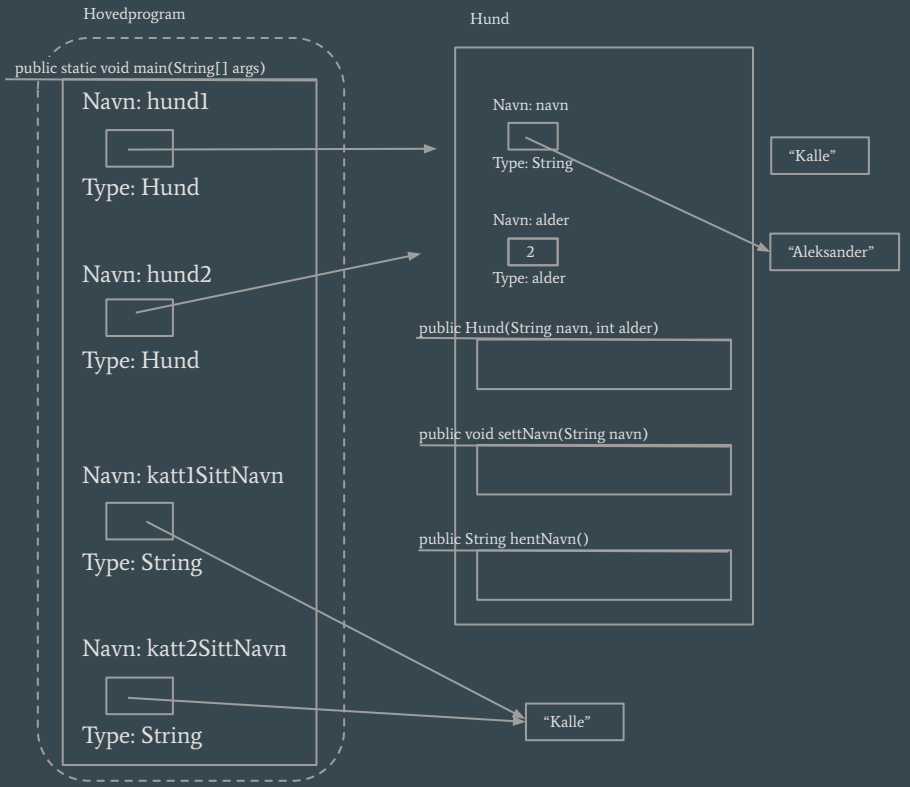

```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Referanser



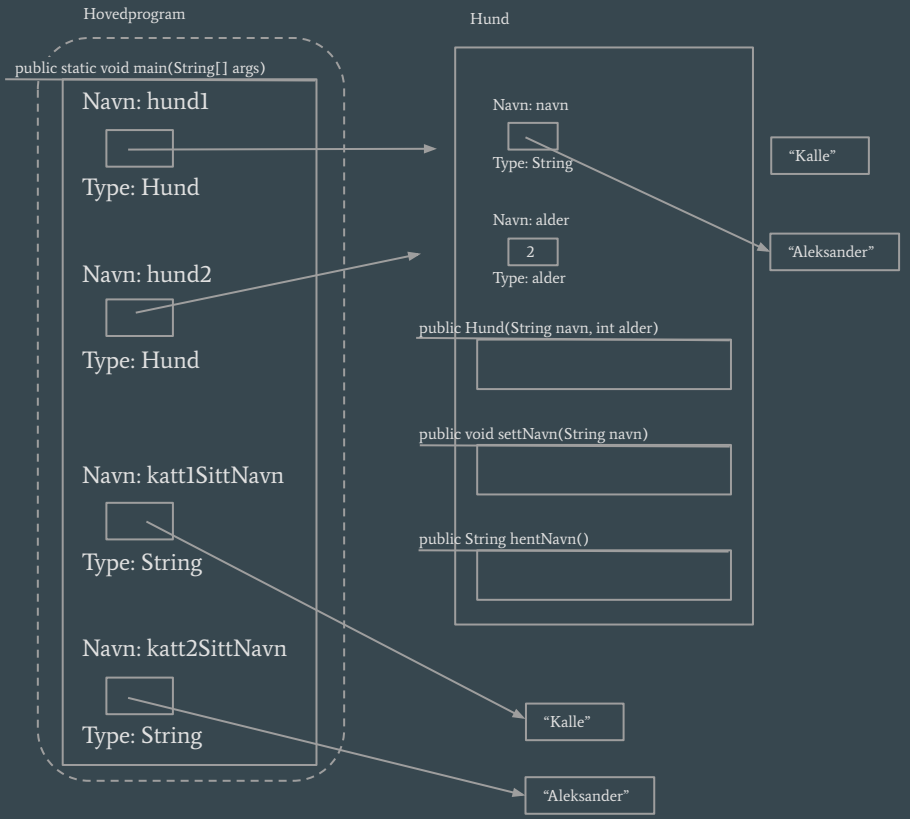

```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Referanser




```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Referanser

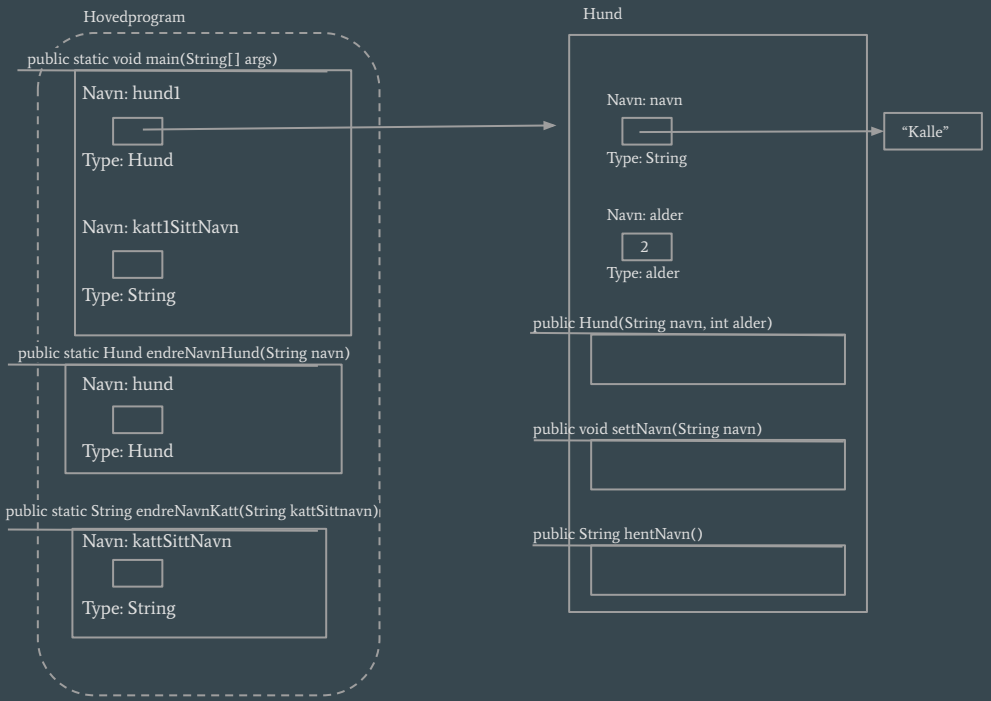



```java
class Hovedprogram1{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    Hund hund2 = hund1;
    hund2.settNavn("Aleksander");
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    String katt2SittNavn = katt1SittNavn;
    katt2SittNavn = "Aleksander";
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }
}
```

Parameteroverføring

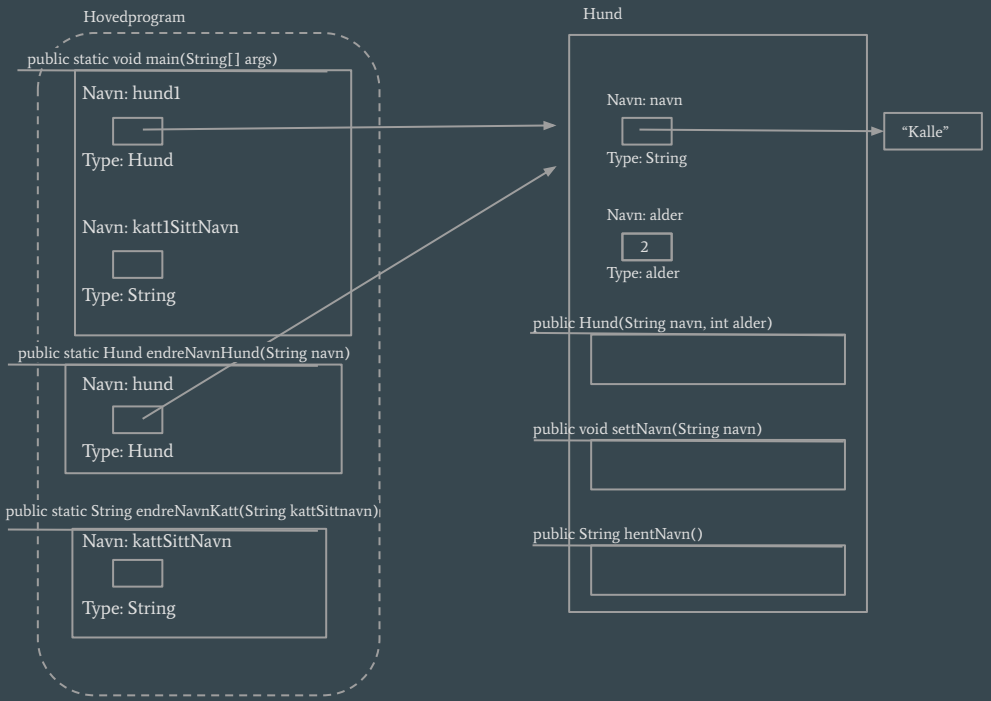
Hvis dere skjønner det med referanser skjønner dere også parameteroverføring!

Paramteroverføring



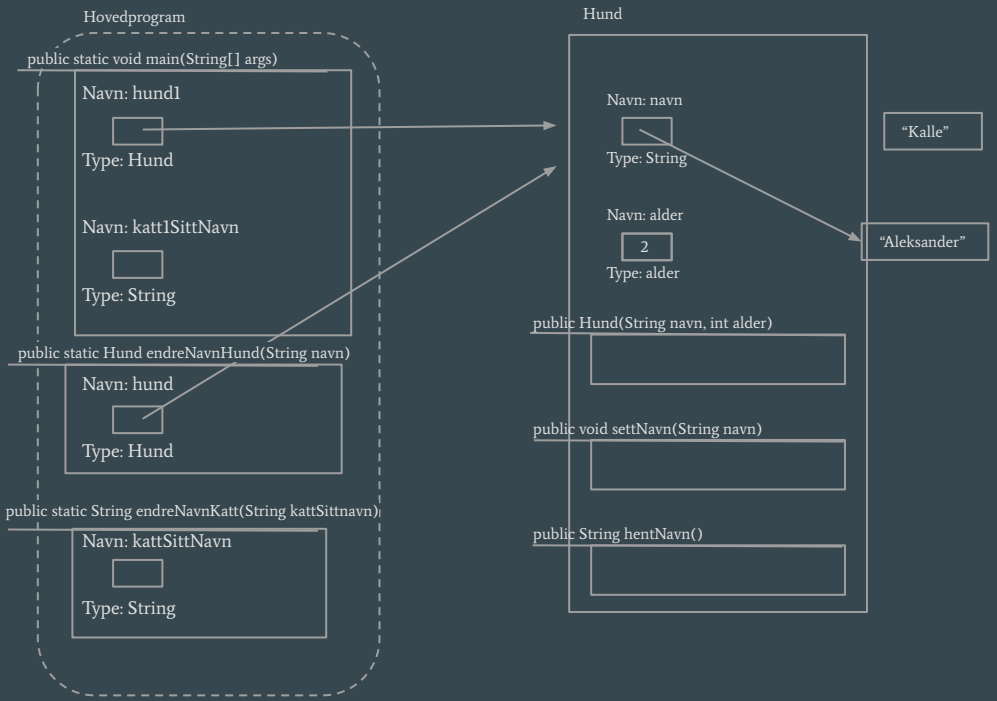

```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Paramteroverføring



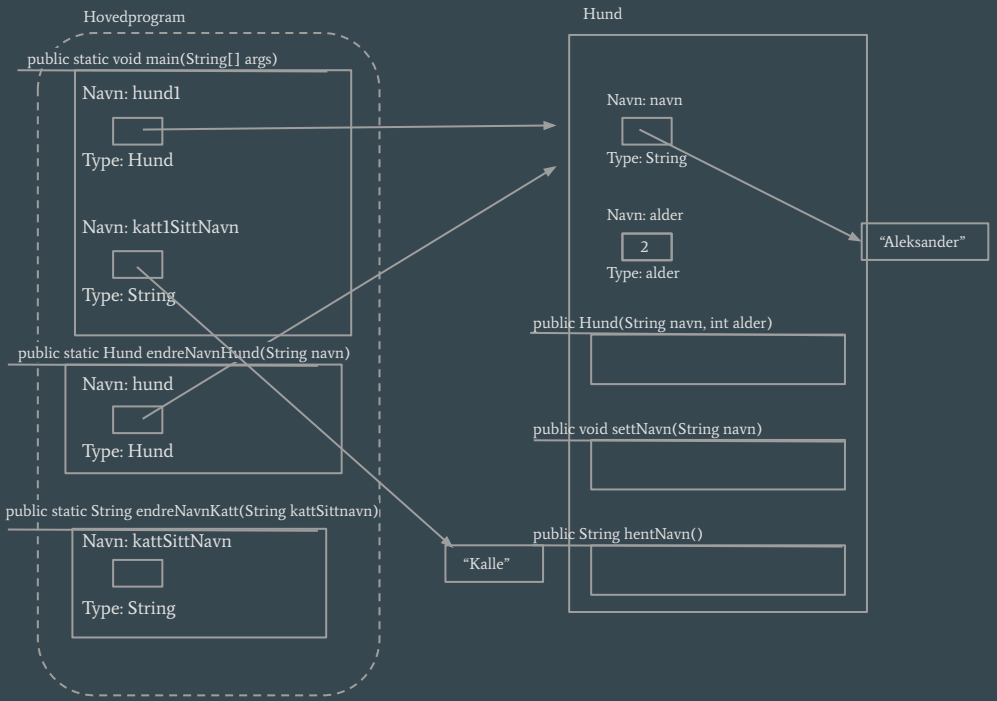

```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Paramteroverføring



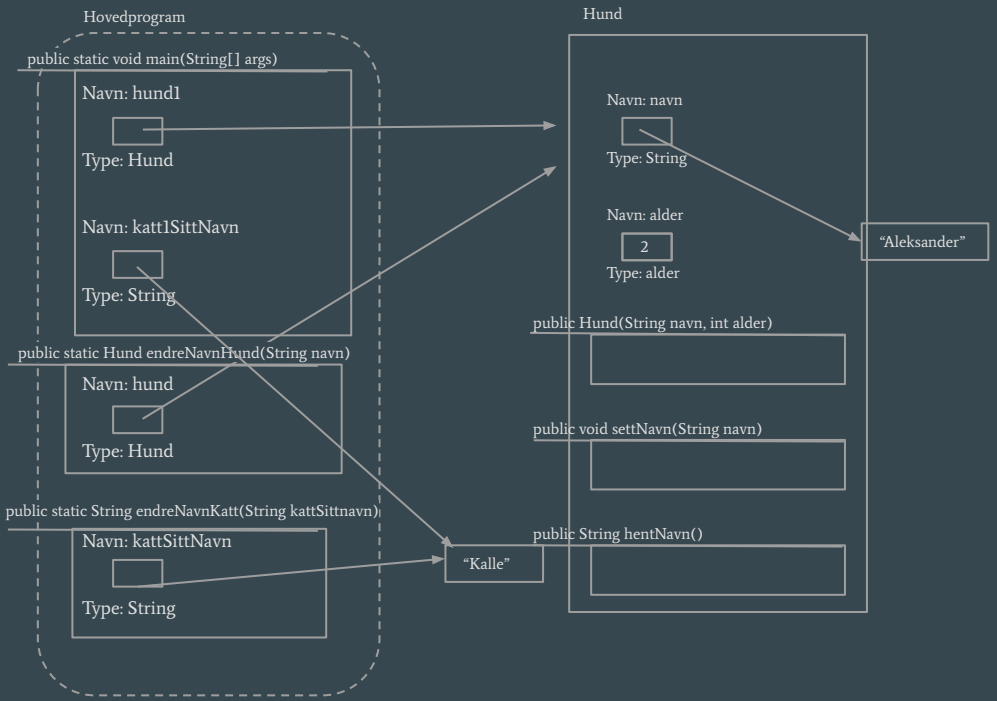

```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Paramteroverføring



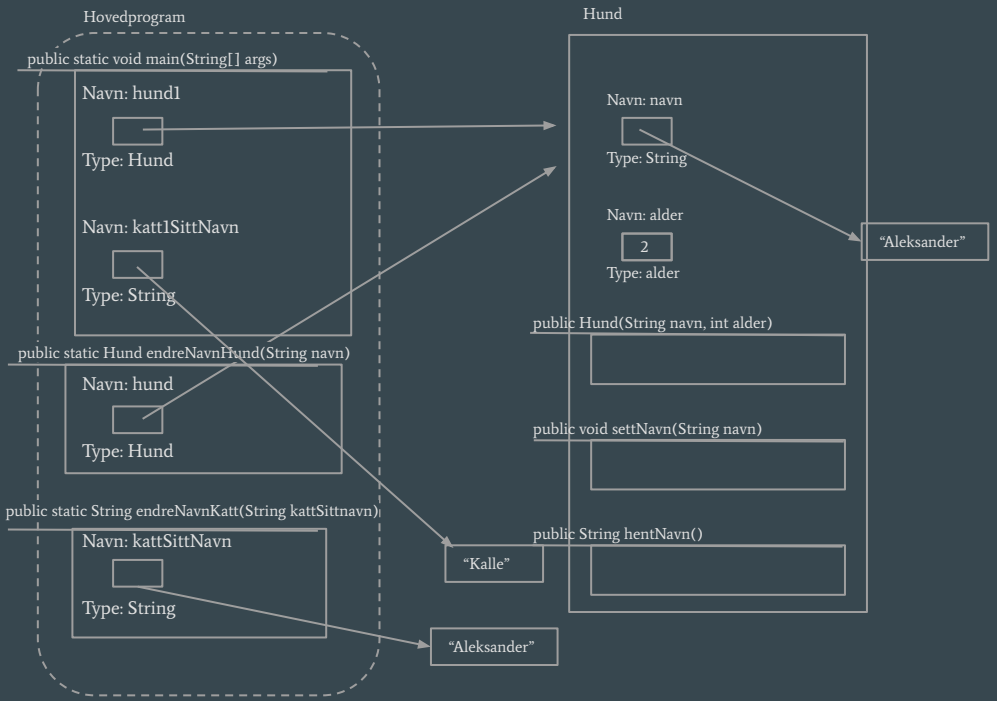

```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Paramteroverføring




```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Paramteroverføring




```java
class Hovedprogram2{

  public static void main(String[] args) {
    Hund hund1 = new Hund("Kalle", 2);
    endreNavnHund(hund1);
    System.out.println("Navnet til hund1 er nå:" + hund1.hentNavn());
    // >> Alexander

    String katt1SittNavn = "Kalle";
    endreNavnKatt(katt1SittNavn);
    System.out.println("Navnet til katt1 er nå:" + katt1SittNavn);
      // >> Kalle
  }

  // Denne metoden er helt unødvendig, vil bare vise et poeng.
  static void endreNavnHund(Hund hund){
    hund.settNavn("Aleksander");
  }

  static void endreNavnKatt(String kattSittNavn){
    kattSittNavn = "Alexander";
  }
}
```

Parameteroverføring

Når dere lager og kaller på metoder som tar argumenter må dere huske på:

- mutable vs immutable
- primitive type vs objekt

Array

Array i java er fixed-size, men man kan endre inholdet

Kan bare ha én type (enten int eller String f.eks., ikke begge)

Man må si hvilken type elementer den skal ha når man deklarerer den

Man kan enten legge elementene direkte inn når man initialiserer, eller legge inn ett og ett element

Man nå si hvilken indeks nye elementer skal ligge på

Man må vite indeksen til elementet man vil hente ut (ev. sjekke alle elementer)

ArrayList

Ikke fixed-size

Har metoder som get() og add()

Kan bare ha én type (enten int eller String f.eks., ikke begge)

Man må si hvilken type elementer den skal ha når man deklarerer den

Man kan enten legge elementene direkte inn når man initialiserer, eller legge elementer senere

ArrayList

Er en klasse, har metoder

- Må importers: java.util.ArrayList
- Link til dokummentasjon: https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Noen metoder: (E er typen du har valgt)

- add(E e) : legge til et element
- size(): returner størrelsen på arraylisten
- get(int index): henter et element på en gitt indeks
- set(E e, int index): setter en verdi på en gitt indeks
- remove(int index): fjerne et element på en gitt index
- isEmpty(): sjekker om lista er tom (boolean)
- clone(): returnerer en kopi av listen
- clear(): fjerner alle elementene fra lista
- toArray(): konverter ArrayListen til ett array og returnerer det

```java
import java.util.ArrayList;

class ArrayListEksempel{

    public static void main(String[] args) {

        ArrayList<String> liste = new ArrayList<>();
        liste.add("Hei ");
        liste.add("paa ");
        liste.add("oss");
        liste.add("!");

        System.out.println(liste.get(0));
        liste.set(2, "deg");
        liste.remove(liste.size() - 1);

        System.out.println(liste);

        if(liste.isEmpty()){
            System.out.println("Listen er tom");
        }else{
            System.out.println("Listen er ikke tom");
        }
    }
}
```

Hashmap

Husk import!

Har keys og values

Det vi kalte ordbok eller dictionary i in1000

Noen metoder

- get(key)
- put(key, value)
- isEmpty(): sjekker om lista er tom (boolean)
- clone(): returnerer en kopi av listen
- clear(): fjerner alle elementene fra lista

```java
// Husk å importere!
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    // Lager et HashMap object som jeg kaller capitalCities
    HashMap<String, String> capitalCities = new HashMap<String, String>();

    // Legger til key og value (Land, By)
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    System.out.println(capitalCities.get("England"));
  }
}
```

Error-håndtering

Med error håndtering (try/catch)

Uten error-håndtering, sendes problemet videre.

Da trenger man try/catch når man kaller på denne metoden ellers vil programmet terminere(avslutte).

(Siden dette er main som "Throws exception" vil programmet terminere ved error.)

```
jonbon@jons-macbook-pro uke2 % java FraFil1
Kan ikke lese filen !
jonbon@jons-macbook-pro uke2 % java FraFil2
Exception in thread "main" java.io.FileNotFoundException: tekst.txt (No such file or directory)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.util.Scanner.<init>(Scanner.java:639)
        at FraFil2.main(Errorhåndtering.java:21)
```

```java
import java.util.Scanner;
import java.io.File;

// Med error-håndtering
class FraFil1{
  public static void main(String[] args){
    // Åpne datafilen:
    Scanner fil = null;
    try {
      fil = new Scanner(new File("tekst.txt"));
    } catch (Exception e) {
        System.out.println("Kan ikke lese filen !");
        System.exit(1);
    }
}
}

// Uten error-håndtering
class FraFil2{
    public static void main(String[] args) throws Exception {
        Scanner fil = new Scanner(new File("tekst.txt"));

        String linje = "";
        while(fil.hasNextLine()) {
            linje = fil.nextLine();
            System.out.println(linje);
        }
    }
}
```

Datastrukturtegning

Anbefaler sterkt å lese dette notatet i sin helhet:

https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/notater/omdatastrukt-2021.pdf

Løsningsfoslag tegning oblig 0:

https://www.uio.no/studier/emner/matnat/ifi/IN1010/v20/Obligatoriske%20innleveringer/oblig-0-losningsforslag/klassediagram_v2.pdf

Error-håndtering!

```java
class Fjellrev{
  private static Boolean rodlistet = true;
  private String navn;
  private int alder;
  private Fjellrev bestevenn;

  public Fjellrev(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public static void setRodlistet(Boolean rodlistetParameter){
    rodlistet = rodlistetParameter;
  }
  public void settnavn(String navn) throws Exception{
    if (navn.equals("Stygg")){
      throw new Exception("Du kan ikke kalle fjellreven din stygg!");
    }
    this.navn = navn;
  }
```

```java
import java.util.Scanner;

class TestFjellrev {
  public static void main(String[] args) throws Exception{
    Fjellrev fjellrev1 = new Fjellrev("Bjarne", 5);
    Fjellrev fjellrev2 = new Fjellrev("Alfred", 10);
    Fjellrev.setRodlistet(false);
    Scanner minScanner = new Scanner(System.in);
    String input = minScanner.nextLine();
    try{
      fjellrev1.settnavn(input);
    } catch(Exception error){
      System.out.println("Du kan ikke kalle Fjellreven din stygg!");
    }
    System.out.println(fjellrev1.hentNavn());

  }
}
```