Arv og subklasser

IN1010 - Uke 3

Tobias Paulsen

Vår 2021

Repetisjon - array

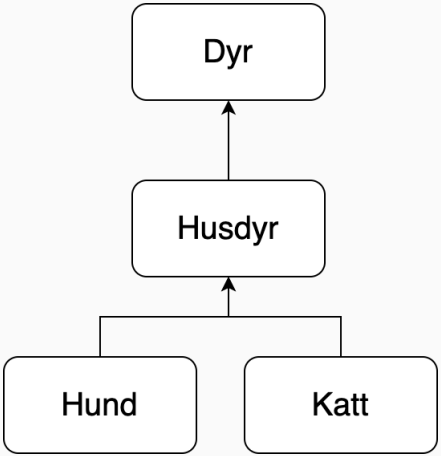
```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        int[] heltall = new int[5];
        double[] flyttall = new double[5];
        char[] tegn = new char[5];
        String[] ord = new String[5];

        System.out.println(Arrays.toString(heltall));
        // [0, 0, 0, 0, 0]
        System.out.println(Arrays.toString(flyttall));
        // [0.0, 0.0, 0.0, 0.0, 0.0]
        System.out.println(Arrays.toString(tegn));
        // [, , , , ]
        System.out.println(Arrays.toString(ord));
        // [null, null, null, null, null]
    }
}
```

Arv

- Objektorientering bruker vi til å simulere den virkelige verdien
- Ofte så ønsker man at helt ulike typer objekter skal ha en del felles i tillegg til en del annerledes
- Ønsker en måte å håndtere ulike klasser sine fellestrekk, dette for å slippe å skrive det samme om og om igjen
- Kan da lage en spesialiserte klasser for spesifikke oppgaver, mens superklassen kan være mer generell

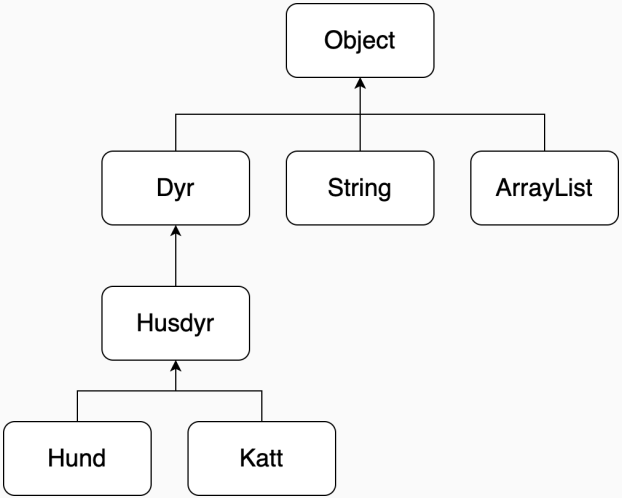
Klassehierarki

- Kan ha flere nivåer med arv
- Alle hunder er både dyr og husdyr
- Alle katter er både dyr og husdyr
- Ingen hunder er katter
- Ingen katter er hunder



Object

- Alle klasser arver fra Object
- Metoder som clone(), equals() og toString()



Nøkkelord

```
class Hund extends Husdyr{...}
```

- extends: for å lage en subklasse

```
protected int alder;
```

- protected: alle av klassens subklasser kan se egenskapen

```
abstract class Dyr {...}
```

- abstract: hvis en klasse er abstrakt kan man ikke opprette en instans av denne klassen, men subklassene arver egenskaper

```
public abstract void spis();
```

- i en abstrakt klasse så kan man også ha abstrakte metoder som man ikke trenger å implementere
- alle subklassene må da implementere metoden

Referanser

```
class A {}

class B extends A {}
```

Variabel av type A kan referere til objekt av B, men ikke omvendt

Kan derfor skrive:

- `A var1 = new A();`
- `A var2 = new B();`
- `B var4 = (B) var2;`

men ikke:

- `B var5 = new A();`
- `B var6 = (B) new A();`