Generics

Generics gjør at feks en funksjon kan ta inn hvilkensomhelst type, eller implementere ting på en smart måte

Kan ikke brukes på primitive typer

Eksempel: Box som kan ha hvilken som helst type i seg

```
public class MyBox<T> {
  // T vil være samme type gjennom hele funksjonen
  private T element = null;

  public void set(T element) {
    this.element = element;
  }

  public T get() {
    return element;
  }
}
```

Eller flere ting:

```
public class Pair<A, B> {
  // A og B vil kunne være samme, eller forskjellige typer gjennom hele
funksjonen
  private A first;
  private B second;

  public Pair(A first, B second) {
    this.first = first;
    this.second = second;
  }

  public A getFirst() {
    return this.first;
  }

  public B getSecond() {
    return this.second;
  }
}
```

eller i en klassemetode:

```
public static <T> T mostCommonElement(List<T> a) {
  HashMap<T, Integer> counts = new HashMap<>();
  int maxCount = 0;
  T maxElement = null;
  for (T x : a) {
    if (counts.containsKey(x)) {
      counts.put(x, counts.get(x) + 1);
    } else {
      counts.put(x, 1);
    }
    if (counts.get(x) > maxCount) {
      maxCount = counts.get(x);
      maxElement = x;
    }
  }
  return maxElement;
}
```

Hvis det er viktig at vi kan kalle på en metode som typen vi får inn skal ha:

feks en `Box` sin `getArea()`

```
public <T extends Box> T largestArea(List<T> a) { // <T extends Box> viktig
  T largest = a.get(0);
  for (T x : a) {
    if (x.getArea() > largest.getArea()) {  // Vi kaller area()-metoden på
typen T!
      largest = x;
    }
  }
  return largest;
}
```

Eller wildcard generics:

```
public static void addTo(Cat cat, List<? super T> list){
  // Her kan ? være en hvilken som helst type over Cat, altså: Mammal,
Animal eller Object
  list.add(cat);
}
```

Kan også bruke

```
<T extends Comparable<T>>
```

slik at man kan putte inn hva som helst (så lenge klassen som går inn implementer `Comparable` og dermed har `compareTo`)

Hvis både `Integer` og `Double` er subtyper av `Number` kan man skrive:`

`List<Number> a = new ArrayList<>();` og da kan man legge inn både double og int objekter

Men OBS! *`List<Integer> og List<Double> er IKKE subtyper av List<Number>`*