Interface Comparable

- Java-biblioteket har et interface kalt **Comparable** som angir at noe er sammenlignbart. Det inneholder kun én metode

```
public interface Comparable<T> {
    public int compareTo(T otherObj);
}
```

- "Vårt" objekt kan sammenlignes med et annet objekt og returnere et heltall
  - < 0 hvis objektet vi kaller compareTo på er "mindre" enn det andre
  - = 0 hvis objektet vi kaller compareTo på er "likt" det andre
  - > 0 hvis objektet vi kaller compareTo på er "større" enn det andre

Klasse som implementerer prioritetskø ved hjelp av Arrayliste

- Klassen ArrayPrioKoe er en subklasse av Arrayliste.

- Klasseparameteren T må implementere Comparable så vi får en sammenligning å sortere etter (og denne sammenligningen må sammenfalle med prioritet)

- Vi redefinerer kun metoden add siden innsettingen skal gjøres sortert.
  De øvrige metodene i Arrayliste kan være som de er.

```
public class ArrayPrioKoe<T extends Comparable<T>> extends Arrayliste<T> {
    @Override
    public void add(T x) {
```

Java interface **Iterable**

- En spesialisert for-løkke itererer gjennom elementene i en beholder som implementerer interface **Iterable**

- interface **Iterable** krever bare at beholderen tilbyr én ekstra metode

`Iterator<T>`	`iterator()` Returns an iterator over elements of type T.

(klippet fra Java API, java.lang)

- Men hva er en **Iterator<T>** som denne metoden skal returnere?

Javas interface **Iterator**

- En iterator kan gi oss elementene i en beholder ett for ett, holde orden på hvor langt vi er kommet i gjennomgangen og sjekke om det er flere elementer igjen

```
public interface Iterator<T> {
    boolean hasNext();
    T next();
}
```

interface **Iterator** ligger i Java API (java.util)

- Vi må med andre ord lage en egen Iterator-klasse for Arrayliste som tilbyr disse metodene for Arrayliste beholdere

- **iterator**-metoden i **Arrayliste** kan så returnere et objekt av denne **Iterator**-klasen