Egne Exceptions (kan også bruke innebygde)

- Feilmeldinger bør være en subklasse av passende Exception
- Her: RuntimeException eller en subklasse(se Exception klasse-hierarki med forklaringer i Big Java eller Java API).
- Konstruktøren tar parametere med nyttig informasjon om feilen (her: hvilken indeks ble brukt, og hvilke er lovlige)

```
class UlovligListeIndeks extends RuntimeException {
    public UlovligListeIndeks(int pos, int max) {
        super("Listeindeks " + pos + " ikke i intervallet 0-" + max);
    }
}
```

Oppdage at noe er feil

- Vi tar vare på relevant informasjon der feil kan oppstå (for eksempel i metoden remove)
- ... og sender den med til Exception-objektet vi oppretter
- ... som vi så "kaster" tilbake til kallstedet med **throw**

```
@Override
public T remove(int pos) {
    if (pos<0 || pos>=iBruk) {
        throw new UlovligListeIndeks(pos, iBruk-1); }
    T res = data[pos];
    for (int i = pos + 1; i < iBruk; i++) {
        data[i - 1] = data[i]; }
    iBruk--;
    return res; }
```

Håndtere feil som oppstår i en metode

- Når vi bruker metoder som kan kaste unntak (som remove) skriver vi en try - catch blokk for å håndtere dem

```
// Lag en feil - og håndter den
try {
    lx.remove(999);
} catch (UlovligListeIndeks u) {
    System.out.println("Feil: " + u.getMessage()); }
```

- Hvordan vet vi om andres metoder kaster unntak?
  - Unchecked: Metode-signatur, dokumentasjon, eller "for sikkerhets skyld"
  - Checked: Alltid i metode-signaturen