UiO : Universitetet i Oslo

Klasser og objekter

Tuva Kristine Thoresen (tuvakt@ifi.uio.no)

Institutt for Informatikk

4. november 2017

Innhold

Klasser og objekter

Hva er objektorientering?

- Oppgaver løses av sammarbeidende *objekter*
- Et objekt har *data*
- og *metoder* som kan manipulere dataen

Klasser

En klasse

- er en oppskrift på et objekt
- beskriver objekter med lik oppførsel
- definerer typen til et objekt

Ved å lage egne klasser kan vi lage nye variabeltyper!

Et eksempel

```
class Telefon:
    def __init__(self, nummer):

    def hent_nummer(self):

    def ring(self, annet_nummer):

    def legg_paa(self):
```

Et eksempel

```
class Telefon:
    def __init__(self, nummer):

    def hent_nummer(self):

    def ring(self, annet_nummer):

    def legg_paa(self):
```

- Grensesnitt: Klassens offentlige metoder

Et eksempel

```
class Telefon:
    def __init__(self, nummer):

    def hent_nummer(self):

    def ring(self, annet_nummer):

    def legg_paa(self):
```

- Grensesnitt: Klassens offentlige metoder
- Innkapsling: Prinsippet med å tilby et offentlig grensesnitt, og gjemme implementasjonen

Objekter

Et objekt er et instans av en klasse

```
min_telefon = Telefon(98765432)
```

Vi kan kalle på metoder i objektet

```
mitt_nummer = telefon.hent_nummer()

min_telefon.ring(12345678)
min_telefon.legg_paa()
```

Implementasjon

Konstruktør og instansvariable

Et objekt består av *data* og metoder

- Vi kaller dataene for *instansvariable*
- En konstruktør definerer og initialiserer disse instansvariablene
- Konstruktøren kalles i det et objekt opprettes

```
class Telefon:
    def __init__(self, nummer):
        self._nummer = nummer

min_telefon = Telefon(98765432)
```

Metoder

Et objekt består av instansvariabler og *metoder*

- Grensesnitt: Metodedeklarasjoner
- Implementasjon: Instansvariabler og innholdet i metodene

```
class Telefon:
    def __init__(self, nummer):
        self._nummer = nummer

    def hent_nummer(self):
        return self._nummer

>> min_telefon = Telefon(98765432)
>> mitt_nummer = min_telefon.hent_nummer()
>> print(mitt_nummer)
98765432
```

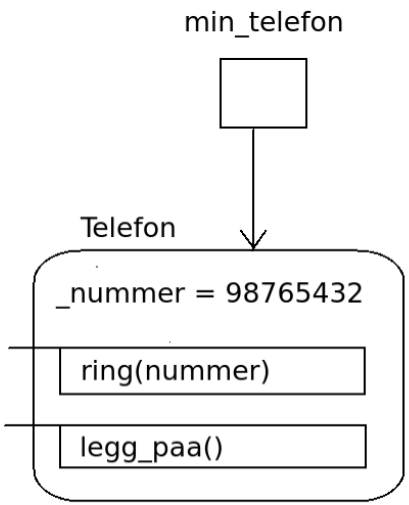
Metoder

```
class Telefon:
    def __init__(self, nummer):
        # ...
        self._ringer = False

    def ring(self, nummer):
        print("Ringer nummer:", nummer)
        self._ringer = True

     def legg_paa(self, nummer):
        self._ringer = False
```

Referanser

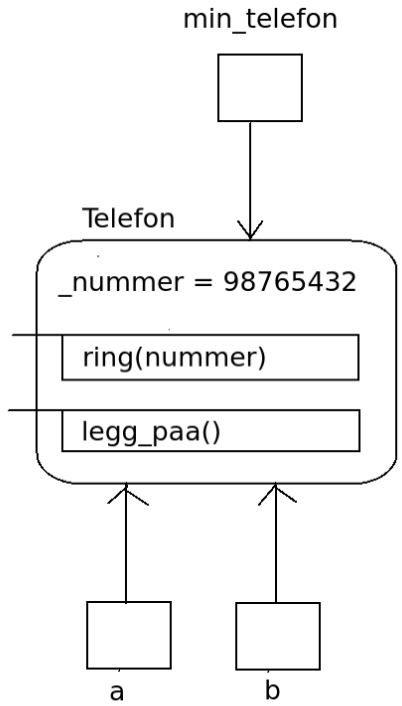
```
min_telefon = Telefon(98765432)

min_telefon.ring(12345678)
min_telefon.legg_paa()
```

- `min_telefon` er en referanse til `Telefon`-objektet
- En referanse holder minnelokasjonen til et objekt
- Den kan brukes til å kalle objektets metoder



Referanser

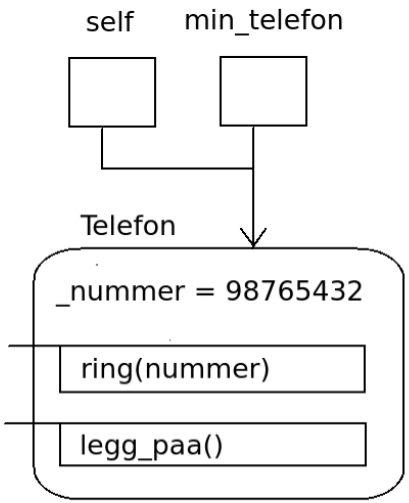
```
min_telefon = Telefon(98765432)

a = min_telefon
b = a
```



- Vi kan også ha flere referanser til et objekt
- Her refererer både `min_telefon`, `a` og `b` til samme objekt
- Alle tre holder minnelokasjonen til objektet

self

```
class Telefon:
    def __init__(self, nummer):
        self._nummer = nummer

min_telefon = Telefon(98765432)
```



- `self` er en referanse som refererer til objektet selv
- Når konstruktøren kalles settes `self` lik objektet som opprettes

self

```
class Telefon:
    def hent_nummer(self):
        return self._nummer

    def skriv_ut(self):
        print("Tlf: ", self.hent_nummer())
```

- `self` må være første parameter i alle metoder
- `self` brukes for å få tilgang til objektets instansvariable
- og for å kalle objektets metoder

Private variabler

- Hvorfor skriver vi `_nummer` og ikke bare `nummer`?

Private variabler

- Hvorfor skriver vi `_nummer` og ikke bare `nummer`?
- Dette er en kodekonvensjon for å deklarere en privat variabel
- Variablen kan ikke (burde ikke) aksesseres utenfor klassen

Private variabler

- Hvorfor skriver vi `_nummer` og ikke bare `nummer`?
- Dette er en kodekonvensjon for å deklarere en privat variabel
- Variablen kan ikke (burde ikke) aksesseres utenfor klassen
- Hvordan kan vi bruke den da?

Private variabler

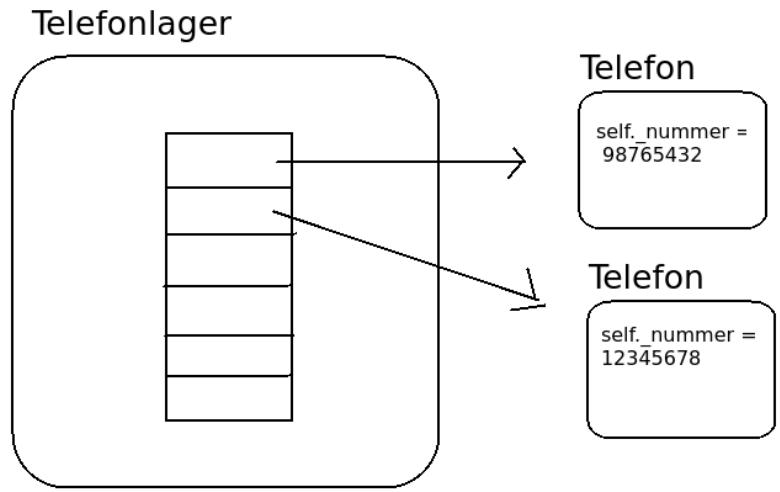
- Hvorfor skriver vi `_nummer` og ikke bare `nummer`?
- Dette er en kodekonvensjon for å deklarere en privat variabel
- Variablen kan ikke (burde ikke) aksesseres utenfor klassen
- Hvordan kan vi bruke den da?
- Vi kan skrive metoder for å hente/sette variablen: `hent_nummer`, `sett_nummer`

Et eksempel

Oppgave: Telefonlager

Vi skal lage et system for å lagre telefoner, et `Telefonlager`. Telefonlageret skal innehold en liste av `Telefon`-objekter, og ulike metoder for å manipulere telefonene i lista.

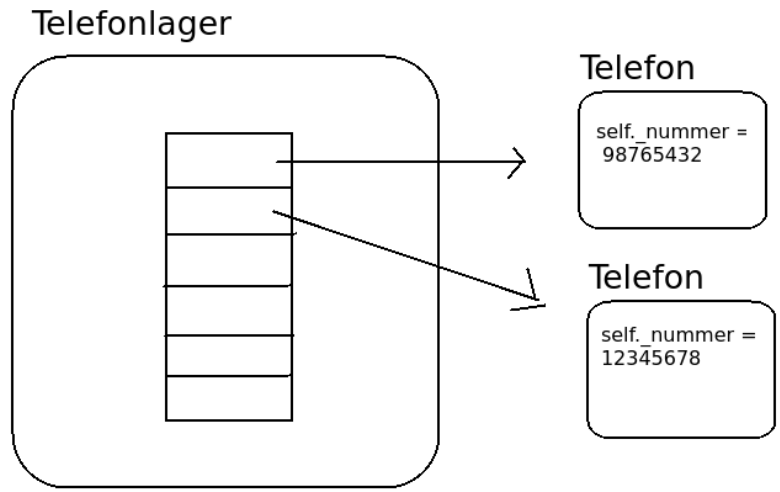
```
class Telefonlager:
    def __init__(self):
        self._lager = []

    def sett_inn(...):
        # ...
```



Oppgave: Telefonlager

Vi skal lage et system for å lagre telefoner, et `Telefonlager`. Telefonlageret skal innehold en liste av `Telefon`-objekter, og ulike metoder for å manipulere telefonene i lista.

```
class Telefonlager:
    def __init__(self):
        self._lager = []

    def sett_inn(...):
        # ...
```



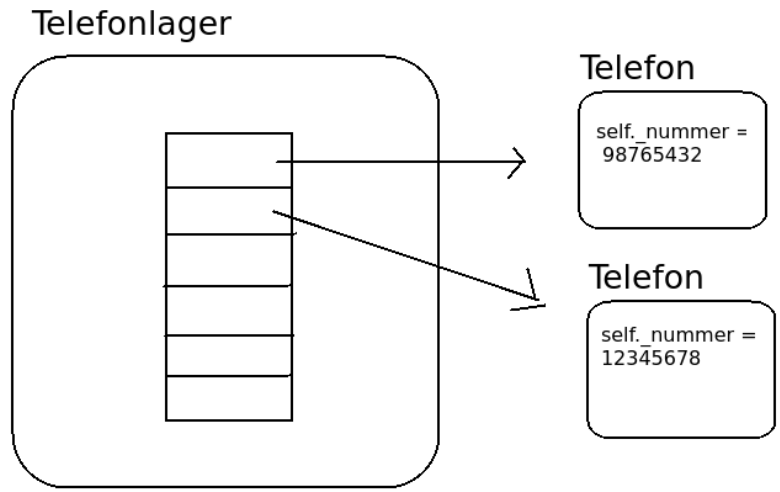
- Første oppgave: Sett inn telefoner i lageret

Sett inn telefoner

```
class Telefonlager:

    def __init__(self):
        self._lager = []

    def sett_inn(self, telefon):
        self._lager.append(telefon)
```

Oppgave: Telefonlager

```
class Telefonlager:
    def __init__(self):
        self._lager = []

    def skriv_ut(...):
        # ...
```



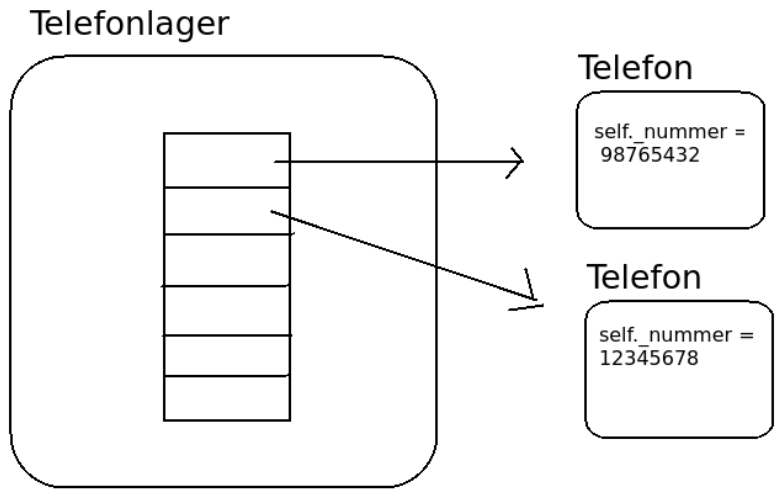
- Andre oppgave: Skriv ut telefonene

Skrive ut telefoner

```
def skriv_ut(self):
    for telefon in self._lager:
        print(telefon.hent_nummer())
```

Oppgave: Telefonlager

```
class Telefonlager:
    def __init__(self):
        self._lager = []

    def finn(...):
        # ...
```

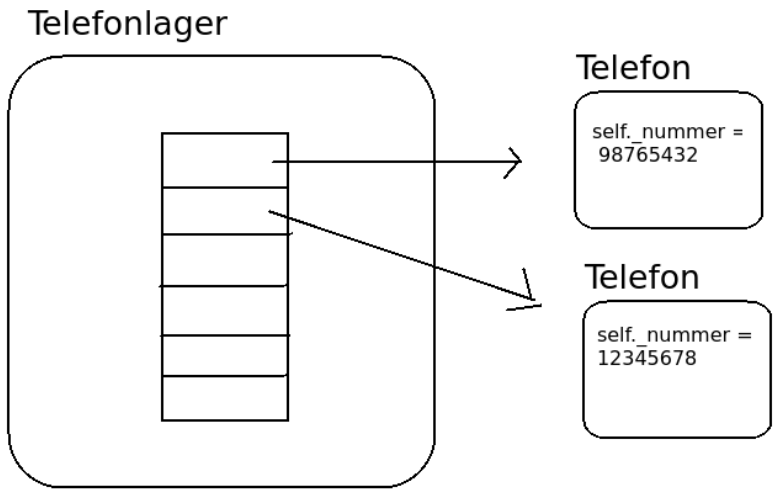


- Tredje oppgave: Finne og returnere en gitt telefon

Finne en gitt telefon

Det er tre spørsmål vi må få svar på:

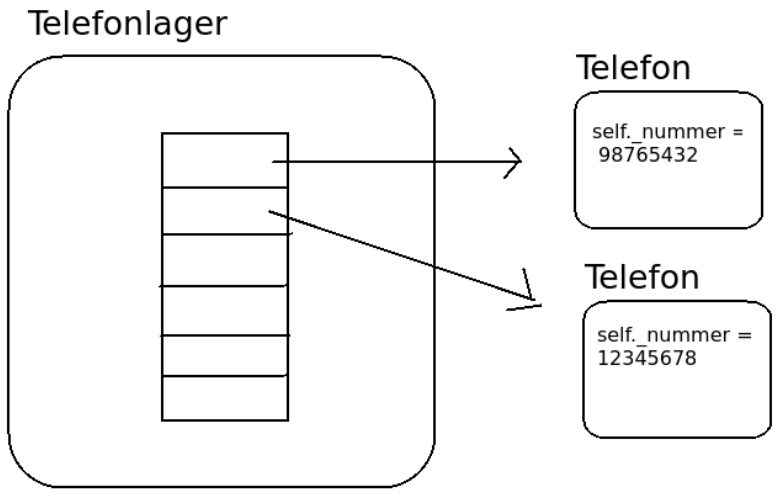
- Hva kan unikt identifisere en telefon?



Finne en gitt telefon

Det er tre spørsmål vi må få svar på:

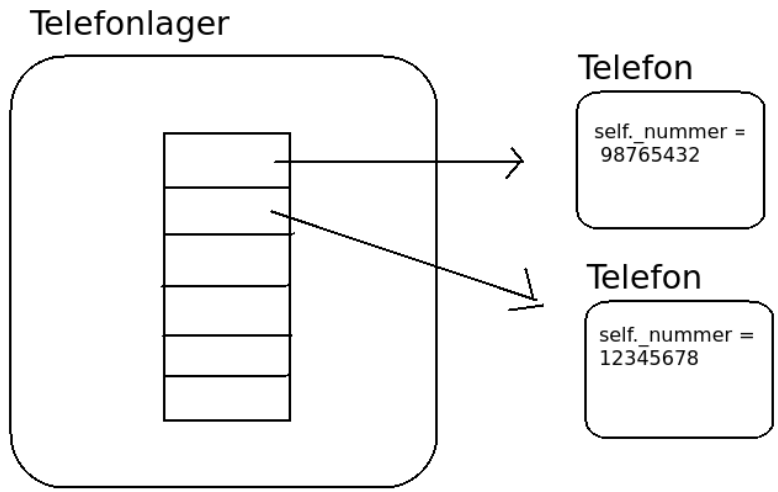
- Hva kan unikt identifisere en telefon?
- Hvordan vet vi om to telefoner er like?



Finne en gitt telefon

Det er tre spørsmål vi må få svar på:

- Hva kan unikt identifisere en telefon?
- Hvordan vet vi om to telefoner er like?
- Hvordan kan vi besøke hvert element i lista?



Finne en gitt telefon

Finne telefonen

```
def finn(self, nummer):
    for telefon in self._lager:
        if (telefon.hent_nummer() == nummer):
            return telefon
```

Finne en gitt telefon

Hva gjør vi hvis vi ikke finner telefonen?

Finne en gitt telefon

Hva gjør vi hvis vi ikke finner telefonen?

- `return None`

En referanse som ikke refererer til noe objekt, har verdien `None`

```
def finn(self, nummer):
    for telefon in self._lager:
        if (telefon.hent_nummer() == nummer):
            return telefon
    return None
```

Flere oppgaver: Telefonlager

Hvis dere synes dette var vanskelig:

- Finn telefonen med det minste telefonnummeret
- Finn telefonen med det største telefonnummeret

Hvis dere synes dette var lett:

- Sorter telefonene i lageret
- Finn det nest minste og nest største telefonnummeret
- Finn det miderste telefonnummeret

Et større eksempel

Kontakter

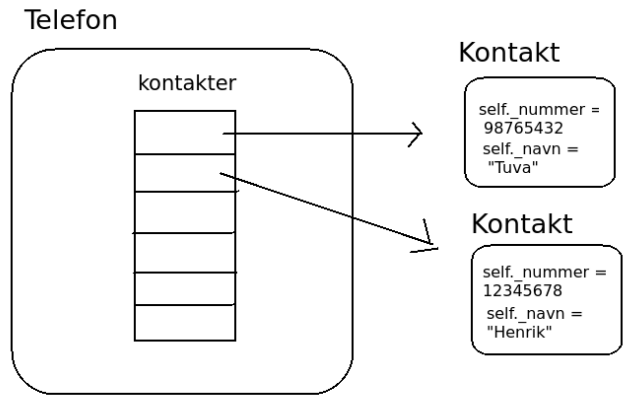
Nå skal vi utvide klassen `Telefon` slik at den kan lagre kontakter. Vi starter med å introdusere klassen Kontakt:

```
class Kontakt:
    def __init__(self, navn, nummer):
        self._navn = navn
        self._nummer = nummer
        # ...
```

Kontakter

En Telefon skal inneholde en liste av `Kontakt`-objekter, og ulike metoder for å manipulere kontaktene i lista.

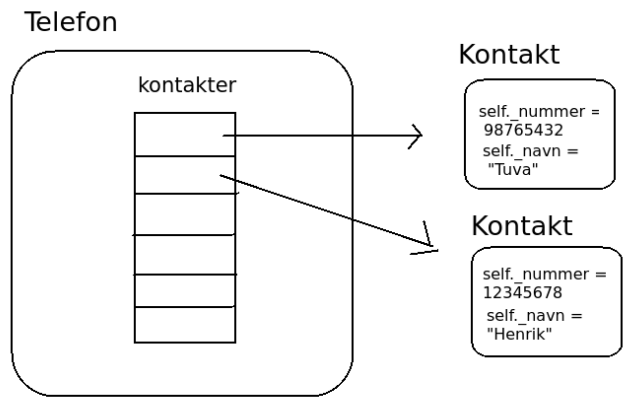
```
class Telefon:
    def __init__(self):
        # ...
        self._kontakter = []

    def sett_inn(...):
        # ...
```



Kontakter

En Telefon skal inneholde en liste av `Kontakt`-objekter, og ulike metoder for å manipulere kontaktene i lista.

```
class Telefon:
    def __init__(self):
        # ...
        self._kontakter = []

    def sett_inn(...):
        # ...
```

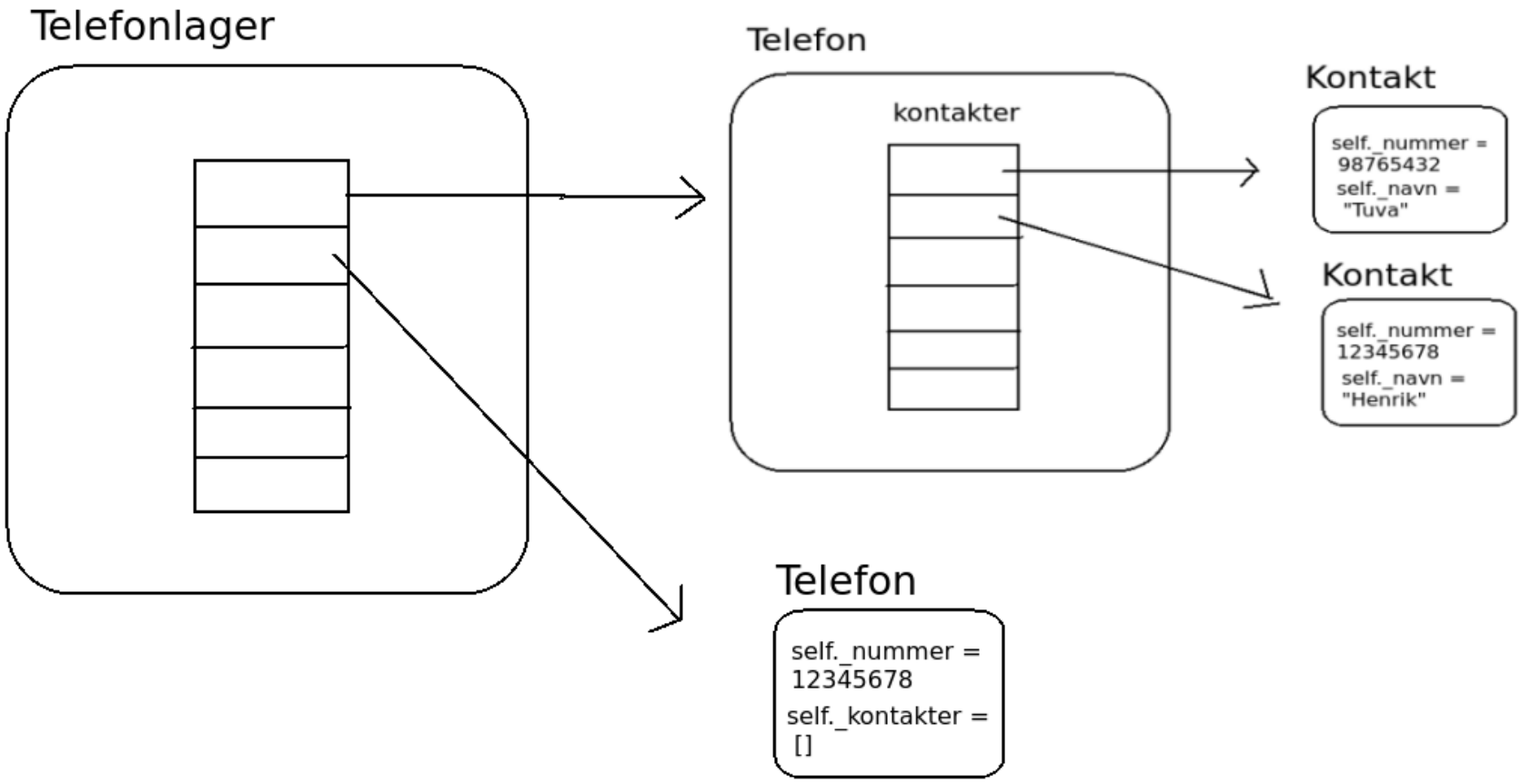


- Oppgave: Sett inn kontakter i telefonen

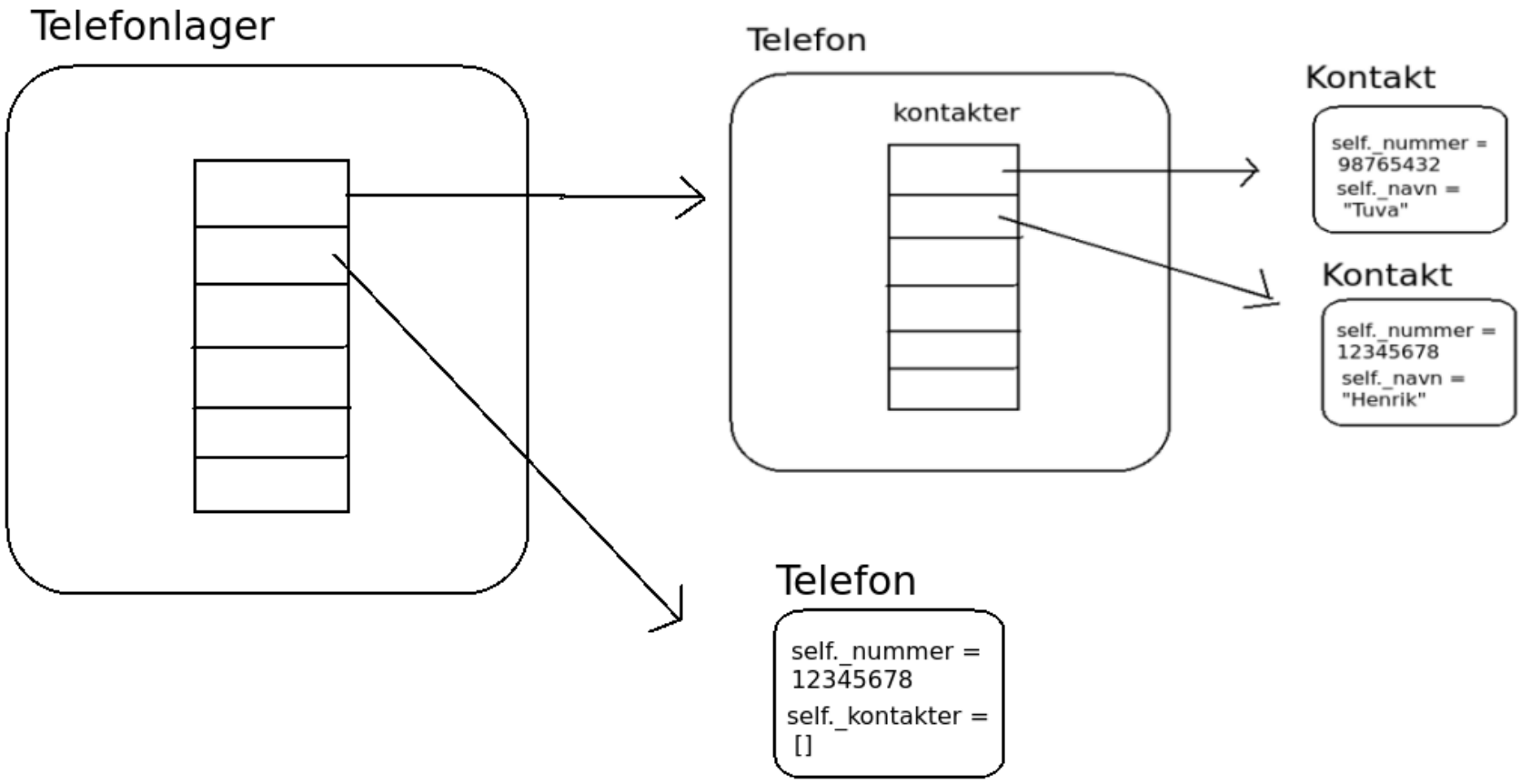
Sett inn kontakter

```
class Telefon:
    def __init__(self):
        # ...
        self._kontakter = []

    def sett_inn(self, kontakt):
        self._kontakter.append(kontakt)
```

Systemet vårt



Systemet vårt



- Oppgave: Hvordan kan vi utvide systemet slik at vi skriver ut kontaktene?

Skrive ut kontakter

Gitt følgende kodeskjellett, hvordan kan vi, for hver telefon, skrive ut alle kontaktene?

```
class Telefonlager:
    def skriv_ut(...):
        # ...

class Telefon:
    def skriv_ut(...):
        # ...
```

```
class Kontakt:
    def skriv_ut(...):
        # ...
```

Skrive ut kontakter

```
class Telefonlager:

    # old
    def skriv_ut(self):
        for telefon in self._lager:
            print(telefon.hent_nummer())

    # new
    def skriv_ut(self):
        for telefon in self._lager:
            telefon.skriv_ut()
```

Skrive ut kontakter

```
class Telefon:
    def skriv_ut(self):
        for kontakt in self._kontakter:
            kontakt.skriv_ut()

class Kontakt:
    def skriv_ut(self):
        print(self._navn, self._nummer)
```

Takk for meg! ☺

Noen spørsmål?