

Parametriserte klasser

Java <Generics>

Agenda

- ⦿ Når bruker man generics
- ⦿ Hvorfor generics
- ⦿ Hvordan bruker man generiske klasser
- ⦿ Hvordan lager man en generisk klasse

Når bruker man generics?

Trenger beholder for ulike ting

Foreksempel

- ⦿ Generell beholder
- ⦿ Generelle lister
- ⦿ Generelle binærtre

ElektronikkVare



Pc

Kamera

Mobil

Hvorfor generics, vi har
jo Object :)

Let's see..

1. Vær smart programmerer og lag en klasse **Beholder**, som tar vare på objekter (**Object!**) av en hvilken som helst type. Klassen Beholder skal ha et Array, og metoder for å legge til og ta ut av beholderen.
2. Lag en main klasse som oppretter Pc-objekter og putter disse inn i en **Beholder** klassen.

hint: <http://heim.ifi.uio.no/inf1010/blog/?p=3720>

Lett! Hva var problemet med det?

3. Vær uskikkelig og putt et kamera-objekt inn i beholderen. Funker det?
4. Vær naiv og hent ut alle objektene du putta inn. Cast dem til Pc objekter.

Veldig lett å få feil under kjøring!

- Kompileringsfeil er mye bedre, for da kan vi fikse feilen før programmet blir lansert

Noen ganger ønsker man
å ha en beholder av en
hvilken som helst type.

Andre ganger vil man at
en beholder bare kan ta
EN bestemt type.

Vi har generics fordi vi
ønsker å låse en peker
til å kun tillate bruk av
EN bestemt type
objekter!

Hvordan?

- `LinkedList <Pc> minePcer = new LinkedList<Pc>();`
- `ArrayList <Kamera> mineKamera = new ArrayList<Kamera>();`
- `Stack <MobilTelefon> mineMobiler = new Stack <MobilTelefon>();`
- `HashMap <String, Pc> = new HashMap<String,Pc>();`

Bak kulissene

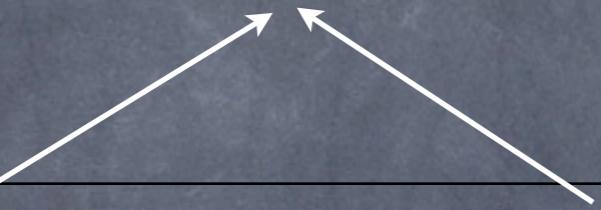
```
class GeneriskBeholder <E> {  
  
    private E generiskPeker;  
  
    public void settPeker (E nyPeker) {  
        generiskPeker = nyPeker;  
    }  
  
}
```

```
class Main {  
  
    public static void main(String []args){  
  
        GeneriskBeholder <Pc> pcer =  
        new GeneriskBeholder<Pc>();  
  
        pcer.settPeker( new Pc() ); // Nice  
        pcer.settPeker( new Bil() ); // Kompilerer ikke  
  
        GeneriskBeholder <Kamera> pcer =  
        new GeneriskBeholder<Kamera>();  
  
        pcer.settPeker( new Kamera() ); // Nice  
        pcer.settPeker( new Kamera() ); // Nice  
    }  
}
```

Hva med arv?

```
class GeneriskBeholder <E> implements BeholderInterface<E> {  
  
    private E generiskPeker;  
  
    public void settPeker (E nyPeker) {  
        generiskPeker = nyPeker;  
    }  
  
    public E hent () {  
        return generiskPeker;  
    }  
}
```

Samme "E"



```
interface BeholderInterface <E> {  
  
    public void settPeker (E nyPeker);  
    public E hent ();  
}
```

```
public static void main(String [] args) {  
    GeneriskBeholder <String> genBh =  
        new GeneriskBeholder <String>();  
  
    genBh.settPeker ("Super string");  
    String minStreng = genBh.hent();  
}
```

Hva hvis man ønsker at beholderen KUN skal godta objekter som er instance of ElektronikkVare??

Brain-teaser!

Men ikke så vanskelig som det ser ut.

Vi ønsker å kunne gjøre dette:

```
ElBeholder<ElektronikkVare> superGenerell = new ElBeholder<ElektronikkVare>();  
superGenerell.leggTil( new Pc() );  
superGenerell.leggTil( new Kamera() );  
superGenerell.leggTil ( new Mobil() );
```

```
ElBeholder <String> superGenerell2 = new ElBeholder<String>(); // Her ønsker vi å få kompileringsfeil!
```

Slik implementeres dette

```
class ElBeholder <E extends ElektronikkVare> {  
  
    private E generiskPeker;  
  
    public void settPeker (E nyPeker) {  
        generiskPeker = nyPeker;  
    }  
  
    public E hent () {  
        return generiskPeker;  
    }  
}
```

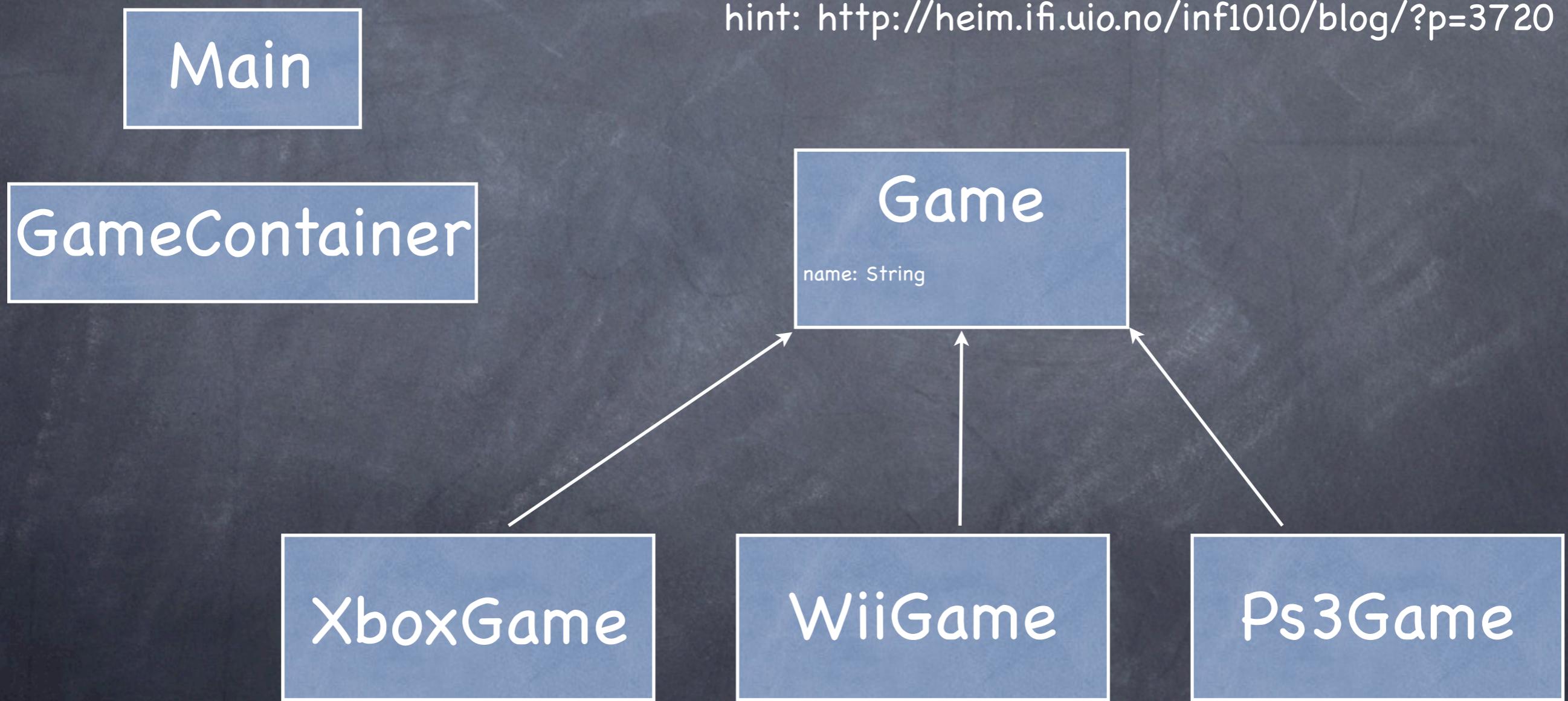
E kan nå være alle typer som extender ElektronikkVare, men ikke noe annet!

Show off session.

Oppgave

1. Lag dette klasse hierarkiet

hint: <http://heim.ifi.uio.no/inf1010/blog/?p=3720>



- ⦿ 2. GameContainer skal ha en simpel peker av en hvilken som helst type. Ikke bruk Object. Object er Evil. Bruk generiske "E".
- ⦿ 3. Lag en metode for å sette dette objektet, og en metode for å returnere objektet. Hint, objektet skal være generisk

• 4. Lag interfacet FifoCollection som har en metode for å putte et generisk objekt inn i en liste, og en metode for å hente et objekt FIFO-wise.

- ➅ 5. Endre GameContainer til å være en listebeholder istedenfor å bare kunne ta vare på ett objekt, ved å implementere grensesnittet FifoCollection. Husk riktig bruk av E!

6. Hvis du har gjort alt riktig er dette lov:

```
GameContainer <XboxGame> xBoxList = new GameContainer <XboxGame>();  
GameContainer <WiiGame> xBoxList = new GameContainer <WiiGame>();  
GameContainer <Bil> xBoxList = new GameContainer <Bil>();  
GameContainer <Hus> xBoxList = new GameContainer <Hus>();
```

Med andre ord, ikke helt det vi vil. Vi ønsker at GameContainer kun kan ta objekter som extender Game. Fix dette.