Mer Java til Python

IN1010 - Uke 2

Tobias Paulsen

Vår 2022

Praktisk

- Oblig 1 er lagt ut, frist 07.02 kl 23.59
- Bruk lab, dårligere kapasitet nærme obligfrist
- Fortsett å sjekke emnesiden
- Oppgaver fra sist
- Datastruktur-notat

Repetisjon

- Tilgang, public/private, hva skjer om man ikke skriver noe?
- Static, uavhengig av objekter
- Konstruktør, metoder, returtyper
- Hvorfor trenger man ikke main-metode i alle klasser?

```java
public class Motorsykkel {
    private int kmStand, prodNummer;
    private String regNr;
    private static int teller = 1;

    public Motorsykkel(String regNr) {
        this.regNr = regNr;
        this.kmStand = 0;
        this.prodNummer = teller++;
    }
    public int hentProdNr() {
        return this.prodNummer;
    }
    public int hentKmStand() {
        return this.kmStand;
    }
    public void kjoer(int antallKm) {
        this.kmStand += antallKm;
    }
}
```

Primitive typer

I java er alt låst til klasser utenom de primitive typene:

- Heltall: `byte` (8 bit), `short` (16 bit), `int` (32 bit), `long` (64 bit)
- Flyttall: `float` (32 bit), `double` (64 bit)
- Boolske verdier: `boolean`
- Tegn: `char` (16 bit Unicode)

Av og til trenger man klasser av disse også, f.eks. ved konverteringer eller bruk av generiske typer (e.g. ArrayList<T>)
Har da wrapper klasser:
`Byte, Short, Integer, Long, Float, Double, Boolean, Character`

Primitive typer

Arrays er ikke en primitive, men objekter av noe som kalles en dynamisk generert klasse. Har derfor ikke metoder.

Repetisjon

- Main, kalles automatisk, ikke i alle klasser
- Array, kun elementer av samme type, ikke utvidbar
  - Hvordan printer man en array?

```
public static void main(String[] args) {
    String[] regNr = {"VF12345", "AG87987", "JO23423", "KJ29344"};
    Motorsykkel[] sykler = new Motorsykkel[regNr.length];

    for (int i=0; i<regNr.length; i++) {
      sykler[i] = new Motorsykkel(regNr[i]);
      sykler[i].kjoer(i*100);
    }

    for (Motorsykkel m : sykler) {
      System.out.println(m.hentKmStand());
    }
}
```

Beholdere

- ArrayList, dynamisk liste
- HashMap, ligner på ordbøker i Python
- Må importeres: `import java.util.*;`

```
ArrayList<String> liste = new ArrayList<String>();
liste.add("Hei");
HashMap<String, String> hovedsteder = new HashMap<String,String>();
hovedsteder.put("Norge", "Oslo");
System.out.println(liste);
System.out.println(hovedsteder);
```

Nyttige metoder ArrayList og HashMap

```
liste.add(E e);                 // Legger til bakerst
liste.add(int index, E e);      // Legger til på index
liste.size();                   // Returnerer størrelsen
liste.get(int index);           // Henter element på index
liste.set(int index, E e );     // Erstatter en verdi på index
liste.remove(int index);        // Fjerner på index
liste.contains(E e);            // Sjekker om e er i lista
liste.isEmpty();                // Gir true om lista er tom, false hvis ikke
liste.clone();                  // Gir en kopi av lista
liste.clear();                  // Tømmer lista
liste.toArray();                // Konverterer til array
// HashMap har size, clear, clone, isEmpty som fungerer likt som for ArrayList
hm.put(K key, V value);         // Legger til i ordboken
hm.containsKey(K key);          // Sjekker om den nøkkelen er i ordboken
hm.containsValue(V value);      // Sjekker om den verdien er i ordboken
hm.get(K key);                  // Henter verdien som hører til den nøkkelen
hm.remove(K key);               // Fjerner mappinga til key
hm.replace(K key, V value);     // Erstatter den gamle verdien
hm.values()                     // Henter alle verdiene i ordboken
```

I/O

- I java brukes **Scanner** til input
- Brukes til input fra bruker og til input som fil
- Må importeres: `import java.util.Scanner;`
- Skriving og lesing i java: link
- Les dokumentasjonen når dere jobber:

`https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html`

- Må håndtere unntak
- For output til terminal brukes: `System.out.println("...");`
- For output til fil bruker man gjerne **PrintWriter**
- Importeres: `import java.io.PrintWriter;`

`https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html`

- Skal se på eksempeler etterpå

Exceptions

- Flere metoder kan kaste feilmeldinger
- F.eks hvis programmet ikke finner filen.(FileNotFoundException)
- I Python er det try, except, finally
- I Java er det try, catch, finally
- `try{...}`: Kodeblokka hvor en feil kan oppstå
- `catch(FileNotFoundException e){...}`: Spesifiserer hvilken exception vi ønsker å fange, sier hva som skjer hvis denne oppstår
- `catch(Exception e){...}`: Fanger alle exceptions, ofte lurt å unngå
- `finally{...}`: Hva som kjøres til slutt uansett
- Skal se mer på feilhåndtering senere i vår