IN1010 - Gruppe 11

Uke 2

Plan for dagen

- Tilgangsnivåer
- Static
- Lese og skrive til fil
- Håndtere unntak (try/catch)
- Oppgaver

Tilgangsnivåer

```
Run | Debug
public static void main(String[] args) {

}
}
```

```
private class eksempel {

}
```

	Default	Private	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	Yes	No	Yes	Yes
Same Package Non-Subclass	Yes	No	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package Non-Subclass	No	No	No	Yes

Men hvorfor har vi de?

- Innkapsling: Skjule implementasjonsdetaljer
- Sikkerhet: Forhindre uautorisert tilgang eller endring av data
- Kodeorganisering: Definere klare grensesnitt
- Vedlikehold: Begrense hvor endringer kan påvirke systemet

- `private` - kun i egen klasse
- `default` - i samme pakke
- `protected` - samme pakke eller subklasser
- `public` - alle har tilgang

Static- metoden

```java
public class Sau {
    // Instansvariabel (tilhører hver enkelt sau)
    String navn;

    // Konstruktør
    public Sau(String navn) {
        this.navn = navn;
    }

    // VANLIG metode (instansmetode) – krever et konkret sau-objekt
    public void bæ() {
        System.out.println(navn + " sier: Bææ!");
    }

    // STATISK metode – tilhører klassen Sau generelt
    public static void forklaring() {
        System.out.println("En sau er et pattedyr i fårefamilien.");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        // VANLIG metode – må ha et sau-objekt:
        Sau lars = new Sau("Lars");
        lars.bæ();  // "Lars sier: Bææ!"

        // STATISK metode – kan kalles direkte via klassen:
        Sau.forklaring();  // "En sau er et pattedyr..."

        // STATISK metode kan IKKE bruke instansvariabler
        // (vet ikke hvilken sau den snakker om)
    }
}
```

Forskjellene

Vanlig metode	Statisk metode
Tilhører objekter	Tilhører klassen
Kan bruke instansvariabler (`navn`)	Kan ikke bruke instansvariabler (uten objekt)
Må kalles på et objekt (`lars.bææ()`)	Kalles via klasse (`Sau.forklaring()`)
Brukes for objektspesifikk logikk	Brukes for generell, delt logikk

Eksempel på typisk statisk metode:

- `Math.sqrt(25)` – trenger ikke et "Matte-objekt", det er bare en generell mattefunksjon.

Lese fra fil

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class LeseFil {

    Run | Debug
    public static void main(String[] args) throws FileNotFoundException {

        String tekst = "";
        File fil = new File("tekst.txt");
        Scanner scanner = new Scanner(fil);

        while (scanner.hasNext()) {
            tekst = scanner.nextLine();
            System.out.println(tekst);
        }
        scanner.close();
    }
}
```

I java så bruker vi ikke with open!!!

1. Lager først en fil objekt med File klassen
2. Deretter lager vi en scanner (akkurat som med inputs!)
3. Deretter en løkke som leser fra scanneren!

Skrive til fil

```java
import java.io.PrintWriter;
import java.io.FileNotFoundException;

public class SkriveFil {
    Run | Debug
    public static void main(String[] args) throws
    FileNotFoundException {
        PrintWriter utfil = new PrintWriter("utfil.txt");

        utfil.println("Tekst her");
        utfil.close();
    }
}
```

1. Lager et objekt av typen PrintWrinter
2. println skriver linjer til filen
3. lukker objektet!

Lab!