GRUPPE 1

Velkommen til
IN1010 – Uke 12

Plan for dagen

- Rekursjon

Hva er rekursjon?

- En rekursiv metode er en metode som kaller på seg selv

- Hva trenger vi for å lage en rekursiv metode?
  - Basistilfelle – Hva skal til for at metoden min skal slutte å kalle på seg selv?
  - Rekursjonssteg – Gjør noe og kall på metoden igjen med endringene gjort

- Fungerer som en while løkke
- Alt som kan gjøres iterativt, kan også gjøres rekursivt

Eksempel

```java
public static void tellRekursivt(int t) {
    if (t < 0) {
        return;
    }
    System.out.println(t);
    tellRekursivt(t - 1);
}

Run | Debug
public static void main(String[] args) {
    tellRekursivt(5);
}
```

```java
public static void tellIterativt(int t) {
    while (t >= 0) {
        System.out.println(t);
        t--;
    }
}

Run | Debug
public static void main(String[] args) {
    tellIterativt(5);
}
```

Pre-order og post-order

- Utfør handling før eller etter det rekursive kallet

```java
public static void tellPreOrder(int t) {
    if (t < 0) {
        return;
    }
    System.out.println(t);
    tellPreOrder(t - 1);
}

Run | Debug
public static void main(String[] args) {
    tellPreOrder(5);
}
```

```
5
4
3
2
1
0
```

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```

```
0
1
2
3
4
5
```

Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```

Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```



tellPostOrder(5)

Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```

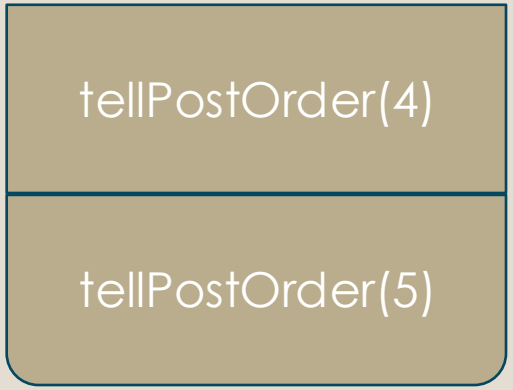
tellPostOrder(5)

Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```



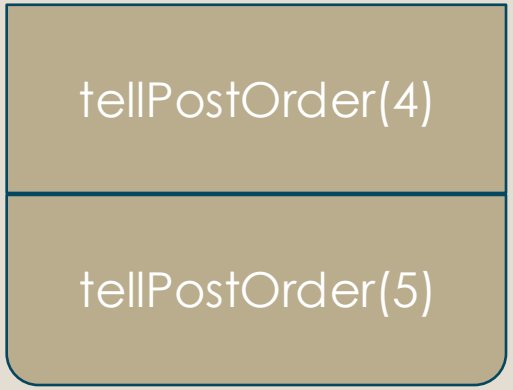


Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```





0

Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```







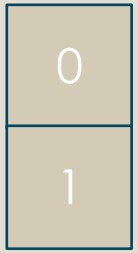
Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```







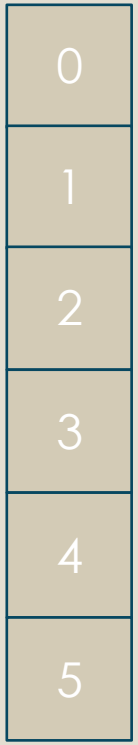
Call stack

```java
public static void tellPostOrder(int t) {
    if (t < 0) {
        return;
    }
    tellPostOrder(t - 1);
    System.out.println(t);
}

Run | Debug
public static void main(String[] args) {
    tellPostOrder(5);
}
```







```java
static int finnStørste(Node node) {

    if (node.neste == null) {
        return node.innhold;
    }

    int maxResten = finnStørste(node.neste);

    if (node.innhold > maxResten) {
        return node.innhold;
    } else {
        return maxResten;
    }

}
```

Eksempel

- Steg 1 – Finn basis steget
- Steg 2 – Hva skal metoden gjøre? Hva skal skje for akkurat denne noden?
- Steg 3 – Anta at rekursjonen funker
- Steg 4 – Hva må vi gjøre med svaret fra rekursjonen og denne noden?

Oppgaver, trix eller oblig :D