Overall objectives

Be able to

- Read and understand a recursive solution
- Use a Binary Search Tree

A recursive solution:
Is built on the idea that ..

- Either the solution is straight forward (obvious)
- Or the solution can be expressed in terms of the solution to a ”**smaller**” problem of ”**the same type”**

A recursive Java method

1. The method calls itself (”recursive call”)
2. The **recursive call** solves a smaller problem
3. The method determines whether a ”**base-case**” has been reached. If this is the case, the recursive call is not made
4. Sooner or later, the smaller problem will turn into the base case

Solve exercise A-1 from day2excercises

Solve the exercises

- factorial,
- bunnyEars,
- bunnyEars2,
- triangle

from

http://codingbat.com/java/Recursion-1

Demo af bunnyEars on http://pythontutor.com/java.html

```
public int bunnyEars(int bunnies) {
        int result = -1; // initial value diff from zero
        if ( bunnies == 0)
                result = 0;
        else {
                result = bunnyEars( bunnies - 1 );
                result += 2;
        }
        return result;
}
```

Binary Search Tree

- Data Structure and Collection
- Binary tree => Binary Search Tree
- Operations
- Good qualities
  - insert and search
  - Traverse in sorted order

Exercises

- Exercises B.1 – B-5

Traversals

- Exercise B6 & B7

Recursive Search algorithm - for binary search tree

If the tree is empty
    element is not in the tree
else
    if root match the key
        element found
    else
        if key < root-value
            search in the left sub tree (repeat)
        else
            search in the right sub tree (repeat)

Exercise B9

Efficiency
- Binary Search Tree

- The height of the tree influences the efficiency of the operations
- Traverse
  - O(n)

- Search/ Insert/ Delete
  - Average : O(log n)
  - Worst : O(n) - Which kind of tree?

Choice of data structure (array/reference/hash tabel/**tree**?
Criterion: Frequency of operations

	array	Linked list	Hash map	Array & linked	**BST (Binary search tree**
Indsæt	Rarely	often	often	often	**often**
Slet	Rarely	often			
søg	often		often		**often**
Gennemløb usorteret		often			
Gennemløb sorteret	often			often	**often**

”Treesort”

- Sort an array:
  - insert all elements into a Binary Search Tree
  - Traverse the tree

- Efficiency
  - Average O(n*log n)
  - Worst $O(n^2)$
  - Principle ?
    - n inserts each of O(log n) => O(n log n)
    - Traversal O(n)

For Wednesday

View the Lynda.com video's mentioned in the readme.md file

Exercise B.10 (and redo those you did not get to do)