



# Multiple Inheritance

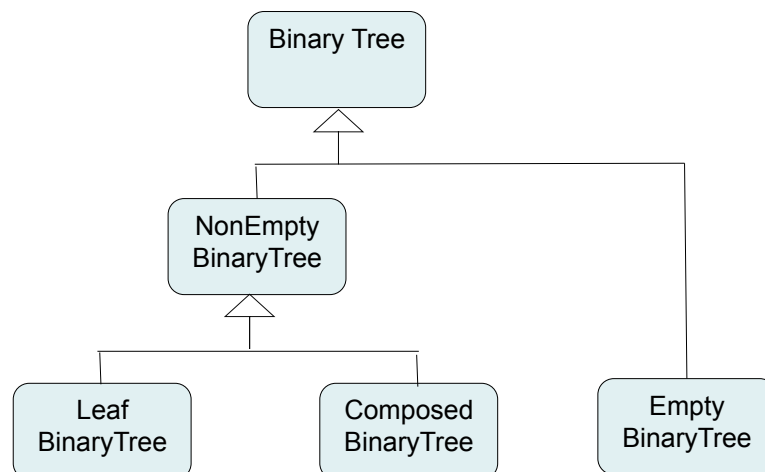
➤ Interfaces



## Binary Trees

- ❑ Binary trees are data structures in which information is spread over a root node, a left subtree and a right subtree
  - ❑ Binary trees store (a reference to) an object in each node
    - This is similar to other data structures in the Java API such as lists and sets
  - ❑ A hierarchy of classes (framework) covering binary trees in general is given
    - The hierarchy distinguishes between empty binary trees, binary leaf trees and composed binary trees
      - Memory requirements are optimized, because leaf nodes do not store references to a left subtree and a right subtree
    - The hierarchy is not parameterized in the type of objects to be stored in each tree
      - Generic classes and interfaces are discussed later

## Binary Trees

KATHOLIEKE UNIVERSITEIT  
LEUVEN

## Search Trees

KATHOLIEKE UNIVERSITEIT  
LEUVEN

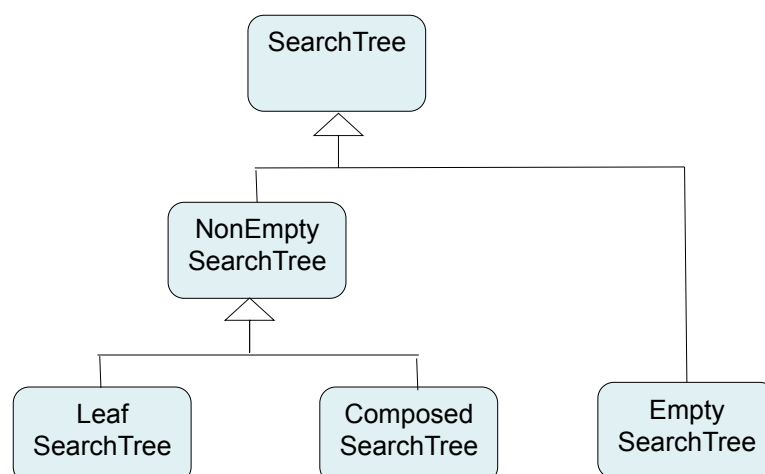
- ❑ Binary search trees are data structures in which elements are ordered
  - ❑ All elements in the left subtree are less than or equal to the root element
    - All elements in the right subtree are greater than the root element
  - ❑ Binary search trees can only store elements of classes that implement the predefined interface “Comparable”
    - The interface “Comparable” offers a single method “compareTo”
- ❑ In the Java API, binary search trees are offered as objects of the collection class “TreeSet”
  - ❑ Treesets are balanced trees, meaning that the left subtree and the right subtree (nearly) have the same number of elements

## Assignment

KATHOLIEKE UNIVERSITEIT  
LEUVEN

- ❑ Assignment 1
  - ❑ Work out implementations for the basic inspector “getNbOccurrencesOf” and for the mutator “addElement”
- ❑ Assignment 2
  - ❑ Work out a hierarchy of classes covering binary search trees
    - The hierarchy must also distinguish between empty search trees, leaf search trees and composed search trees
    - Classes introducing search trees will inherit as much as possible from the general hierarchy for binary trees (framework instantiation)

## Search Trees

KATHOLIEKE UNIVERSITEIT  
LEUVEN

## Interfaces

KATHOLIEKE UNIVERSITEIT  
**LEUVEN**

- ❑ An interface is a construct introducing specifications of methods
  - ❑ By definition, all the methods in an interface are abstract and public
    - An interface can be compared with a pure abstract class
  - ❑ Except for static final variables (constants), an interface cannot introduce aspects related to representation or implementation
- ❑ A class can implement an unrestricted number of interfaces
  - ❑ A concrete class must work out implementations of all the methods it inherits from all the interfaces it implements
    - An abstract class may postpone the implementation of inherited methods
- ❑ An interface can extend an unrestricted number of other interfaces
  - ❑ Methods inherited from super-interfaces equally apply to objects of the sub-interface

## Interfaces

KATHOLIEKE UNIVERSITEIT  
**LEUVEN**

- ❑ Identical methods inherited from different interfaces or from the superclass are joined
  - ❑ Inherited methods with differences in their signature are overloaded, as far as possible
    - A class or interface inheriting methods that only differ in their return type, must use the strongest type
    - A class or interface inheriting methods that only differ in their access right, must use the widest access type
    - A class or interface inheriting methods that only differ in their throws-list, must use a subset of all inherited throws-lists
- ❑ Polymorphism (and dynamic binding) also applies to interfaces
  - ❑ A variable declared to reference objects of an interface, can be assigned objects of a class implementing that interface