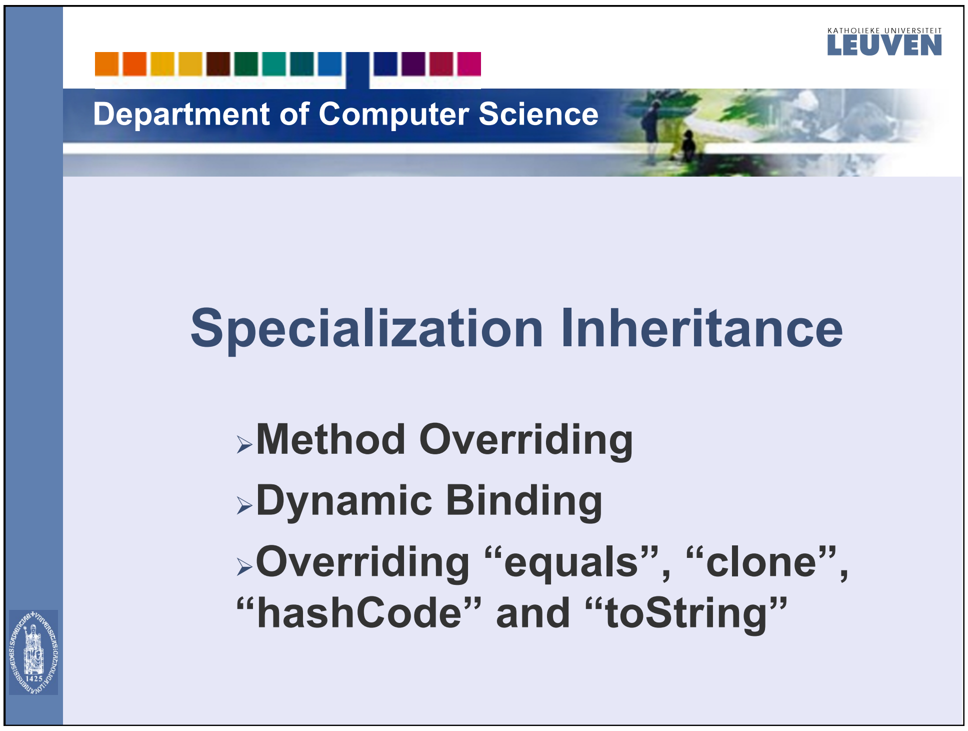
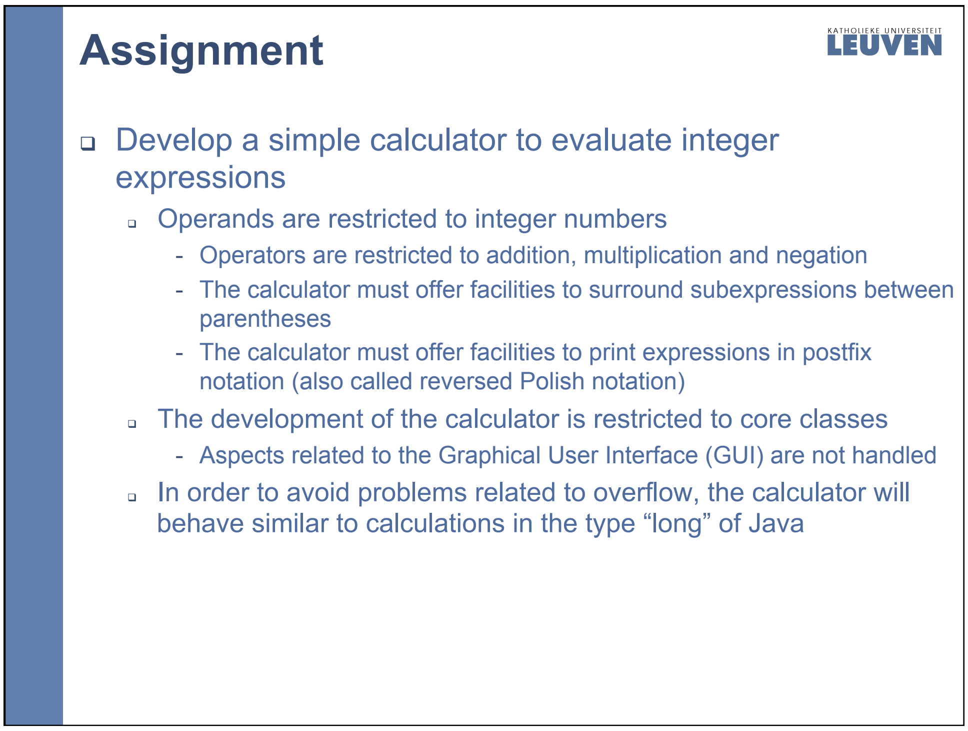


KATHOLIEKE UNIVERSITEIT LEUVEN

Department of Computer Science

Specialization Inheritance

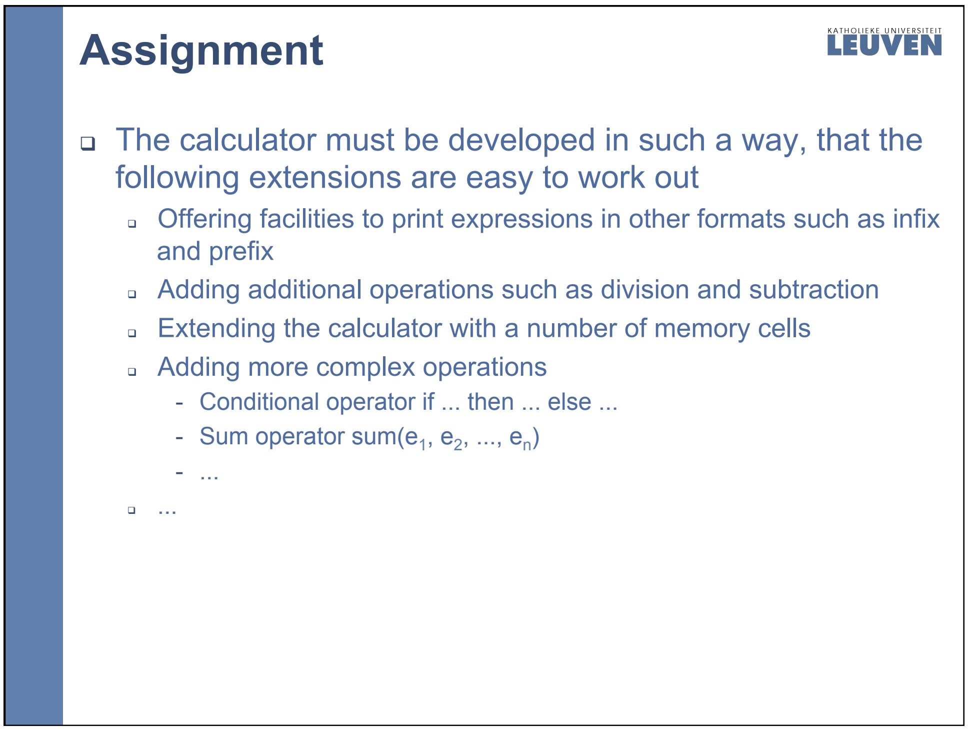
- Method Overriding
- Dynamic Binding
- Overriding “equals”, “clone”, “hashCode” and “toString”



Assignment

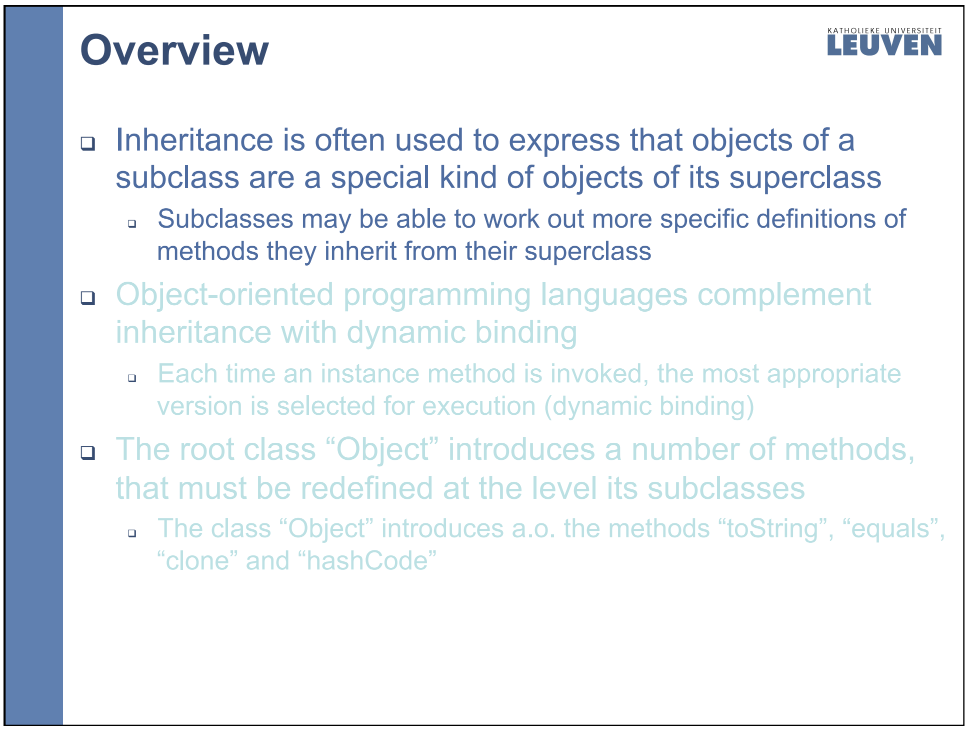
KATHOLIEKE UNIVERSITEIT LEUVEN

- Develop a simple calculator to evaluate integer expressions
  - Operands are restricted to integer numbers
    - Operators are restricted to addition, multiplication and negation
    - The calculator must offer facilities to surround subexpressions between parentheses
    - The calculator must offer facilities to print expressions in postfix notation (also called reversed Polish notation)
  - The development of the calculator is restricted to core classes
    - Aspects related to the Graphical User Interface (GUI) are not handled
  - In order to avoid problems related to overflow, the calculator will behave similar to calculations in the type “long” of Java



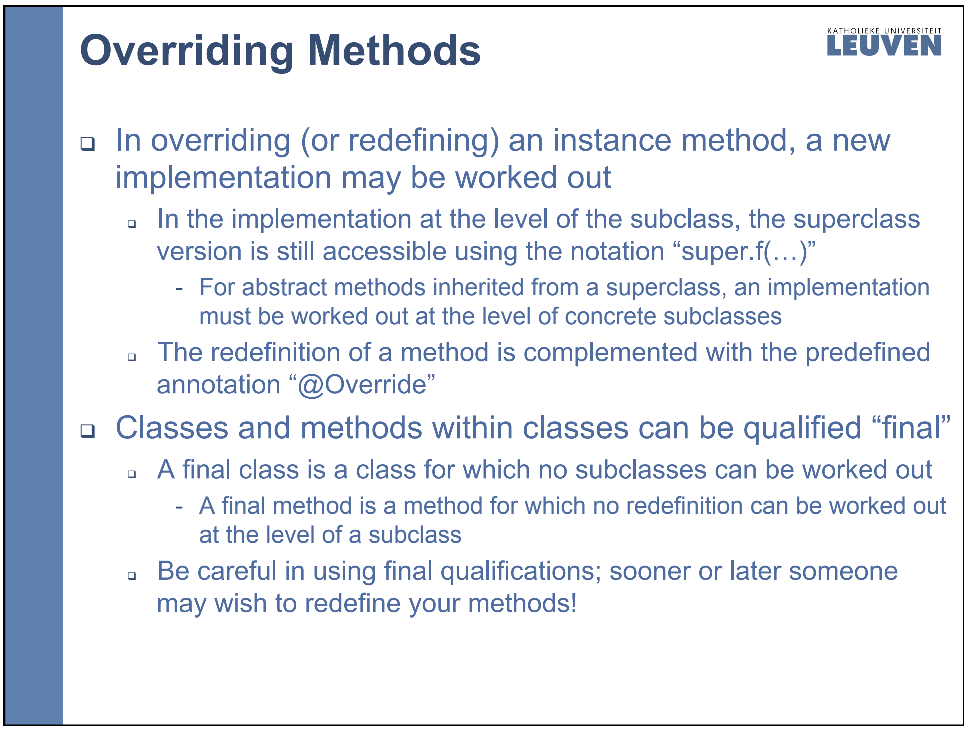
Assignment

- The calculator must be developed in such a way, that the following extensions are easy to work out
  - Offering facilities to print expressions in other formats such as infix and prefix
  - Adding additional operations such as division and subtraction
  - Extending the calculator with a number of memory cells
  - Adding more complex operations
    - Conditional operator if ... then ... else ...
    - Sum operator sum(e_1, e_2, ..., e_n)
    - ...
  - ...



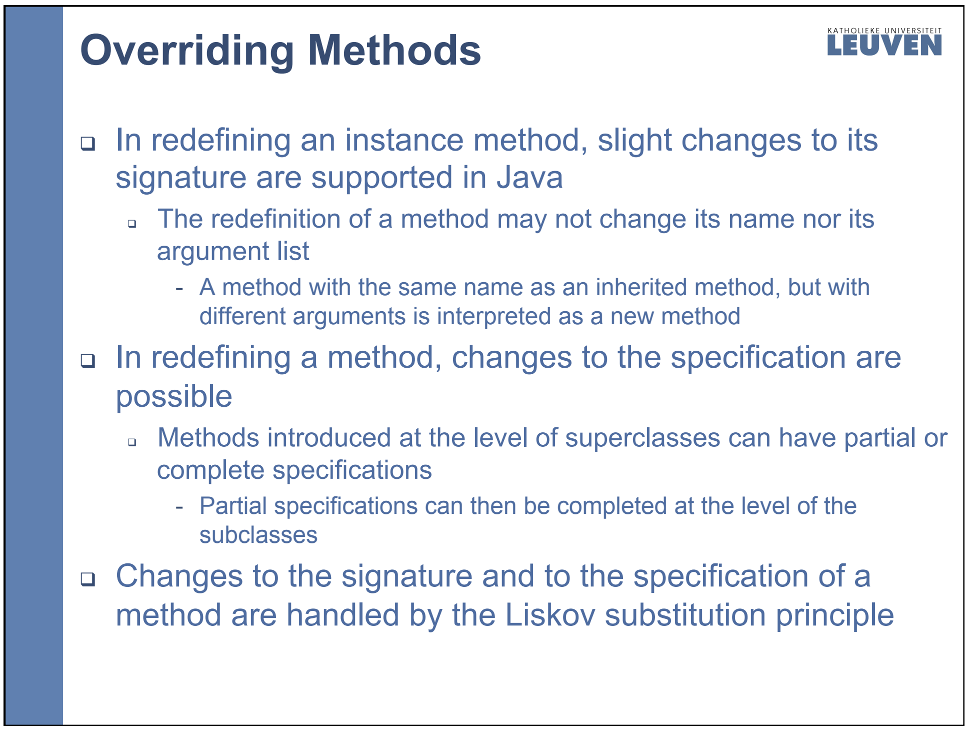
Overview

- Inheritance is often used to express that objects of a subclass are a special kind of objects of its superclass
  - Subclasses may be able to work out more specific definitions of methods they inherit from their superclass
- Object-oriented programming languages complement inheritance with dynamic binding
  - Each time an instance method is invoked, the most appropriate version is selected for execution (dynamic binding)
- The root class “Object” introduces a number of methods, that must be redefined at the level its subclasses
  - The class “Object” introduces a.o. the methods “toString”, “equals”, “clone” and “hashCode”



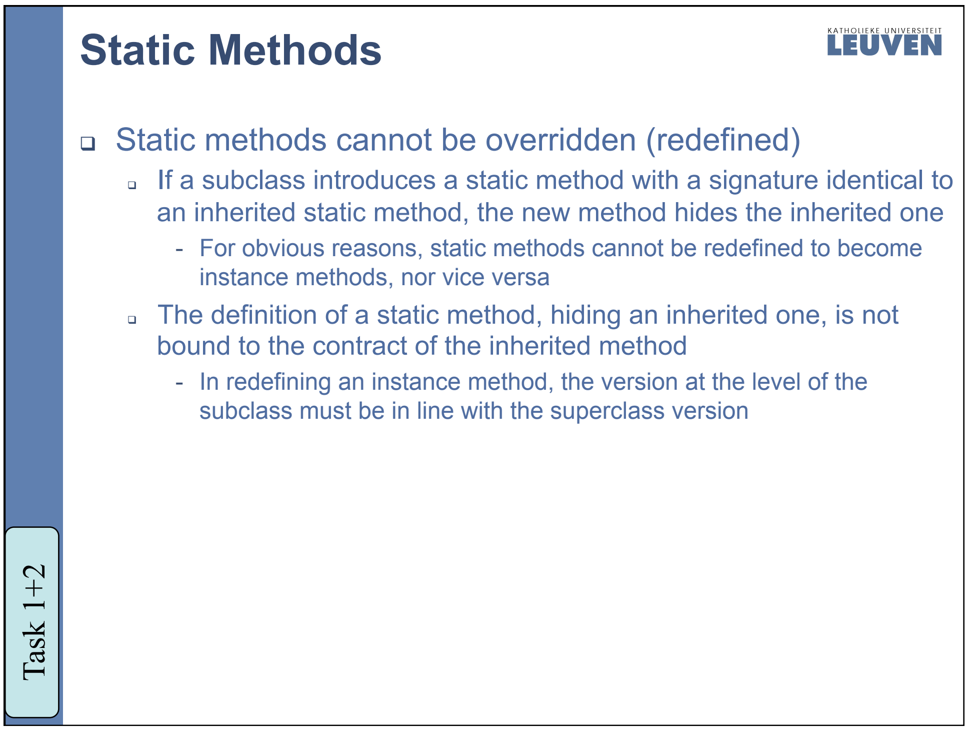
Overriding Methods

- In overriding (or redefining) an instance method, a new implementation may be worked out
  - In the implementation at the level of the subclass, the superclass version is still accessible using the notation “super.f(…)”
    - For abstract methods inherited from a superclass, an implementation must be worked out at the level of concrete subclasses
  - The redefinition of a method is complemented with the predefined annotation “@Override”
- Classes and methods within classes can be qualified “final”
  - A final class is a class for which no subclasses can be worked out
    - A final method is a method for which no redefinition can be worked out at the level of a subclass
  - Be careful in using final qualifications; sooner or later someone may wish to redefine your methods!



Overriding Methods

- In redefining an instance method, slight changes to its signature are supported in Java
  - The redefinition of a method may not change its name nor its argument list
    - A method with the same name as an inherited method, but with different arguments is interpreted as a new method
- In redefining a method, changes to the specification are possible
  - Methods introduced at the level of superclasses can have partial or complete specifications
    - Partial specifications can then be completed at the level of the subclasses
- Changes to the signature and to the specification of a method are handled by the Liskov substitution principle

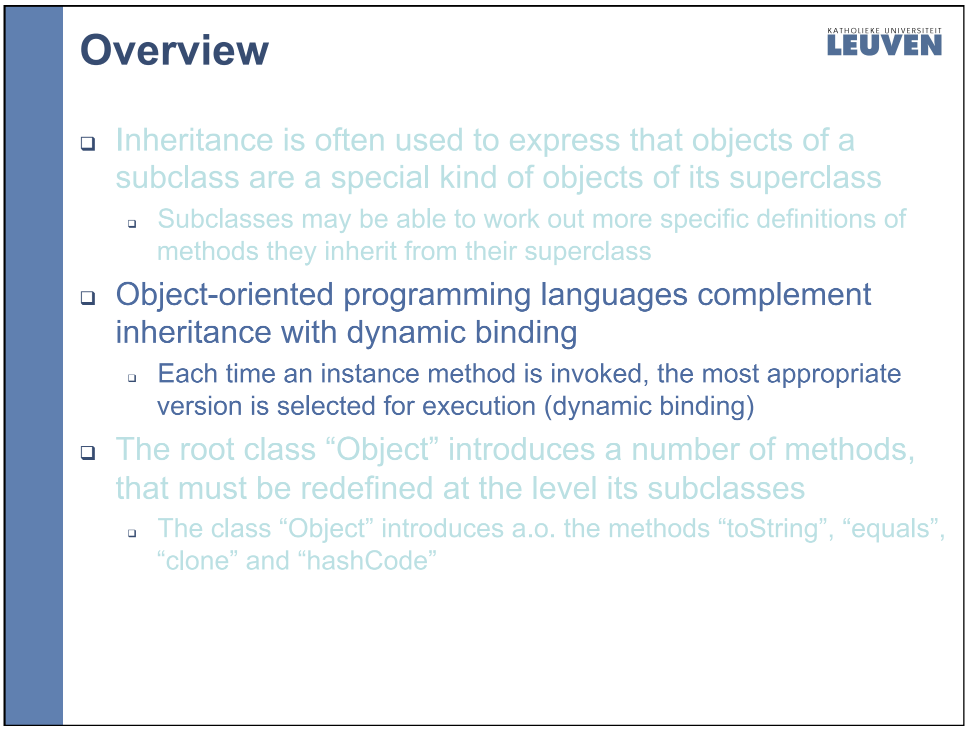


Static Methods

KATHOLIEKE UNIVERSITEIT LEUVEN

- Static methods cannot be overridden (redefined)
  - If a subclass introduces a static method with a signature identical to an inherited static method, the new method hides the inherited one
    - For obvious reasons, static methods cannot be redefined to become instance methods, nor vice versa
  - The definition of a static method, hiding an inherited one, is not bound to the contract of the inherited method
    - In redefining an instance method, the version at the level of the subclass must be in line with the superclass version

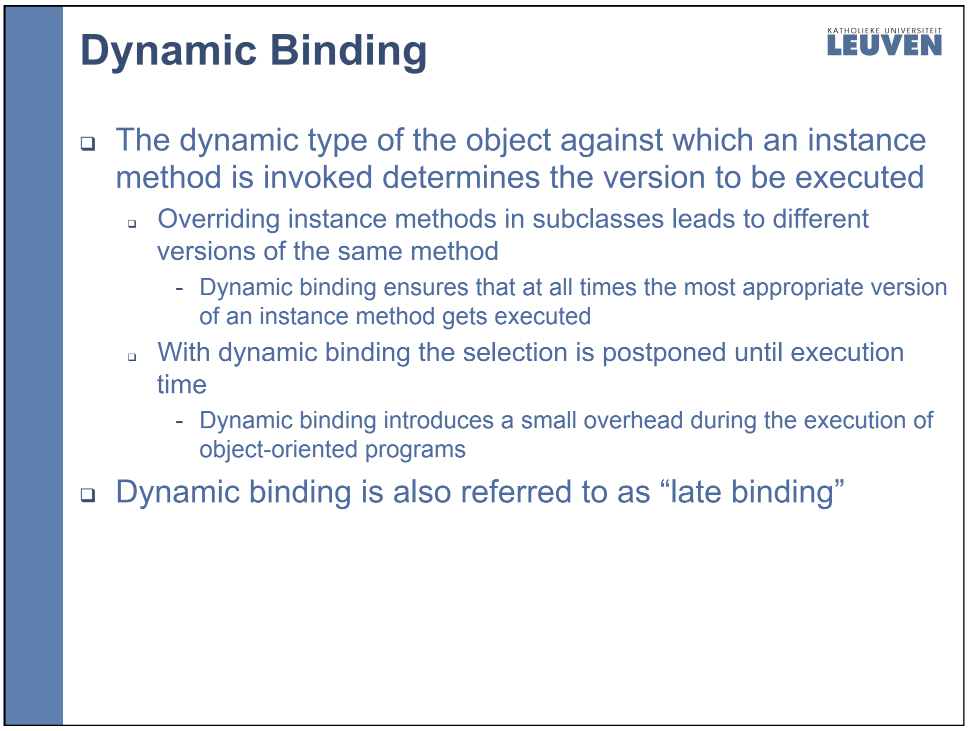
Task 1+2



Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

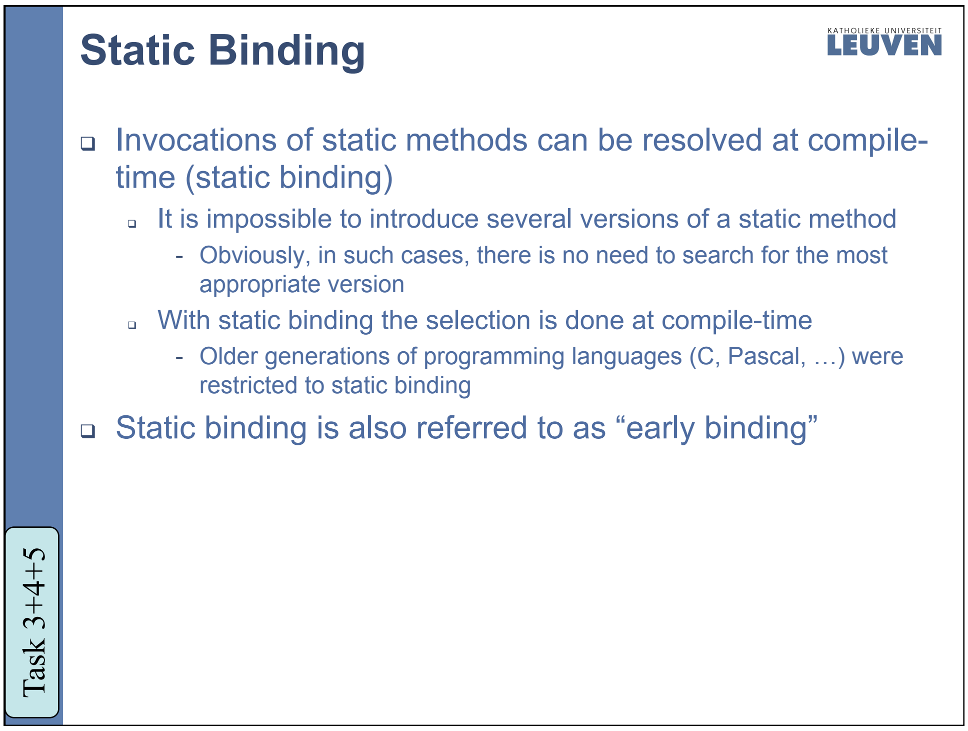
- Inheritance is often used to express that objects of a subclass are a special kind of objects of its superclass
  - Subclasses may be able to work out more specific definitions of methods they inherit from their superclass
- Object-oriented programming languages complement inheritance with dynamic binding
  - Each time an instance method is invoked, the most appropriate version is selected for execution (dynamic binding)
- The root class “Object” introduces a number of methods, that must be redefined at the level its subclasses
  - The class “Object” introduces a.o. the methods “toString”, “equals”, “clone” and “hashCode”



Dynamic Binding

KATHOLIEKE UNIVERSITEIT LEUVEN

- The dynamic type of the object against which an instance method is invoked determines the version to be executed
  - Overriding instance methods in subclasses leads to different versions of the same method
    - Dynamic binding ensures that at all times the most appropriate version of an instance method gets executed
  - With dynamic binding the selection is postponed until execution time
    - Dynamic binding introduces a small overhead during the execution of object-oriented programs
- Dynamic binding is also referred to as “late binding”

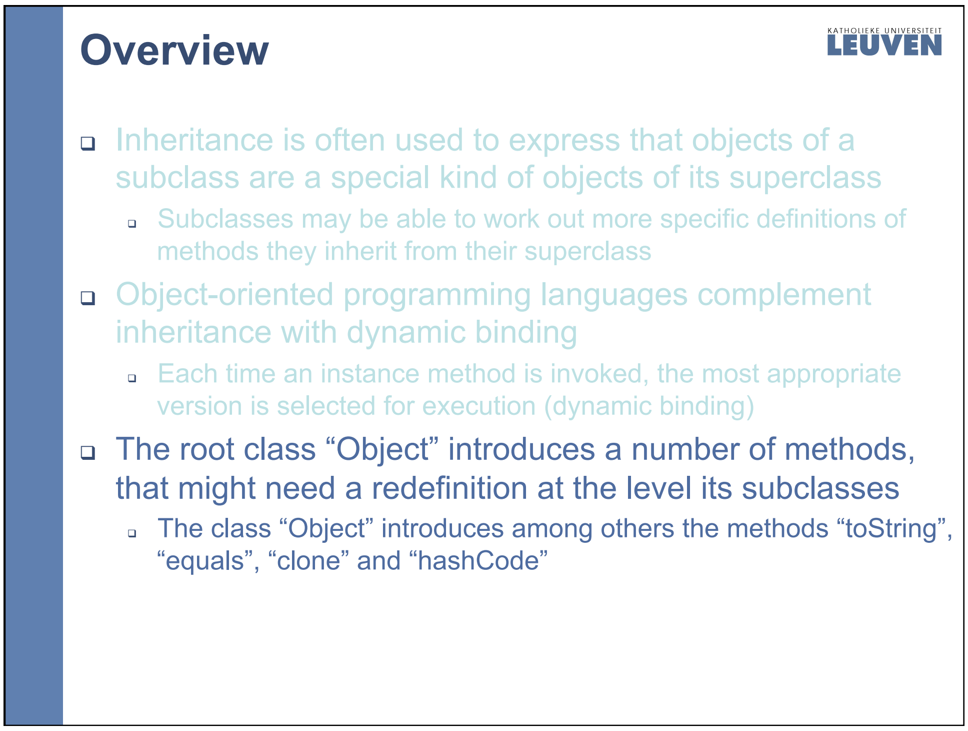


Static Binding

KATHOLIEKE UNIVERSITEIT LEUVEN

- Invocations of static methods can be resolved at compile-time (static binding)
  - It is impossible to introduce several versions of a static method
    - Obviously, in such cases, there is no need to search for the most appropriate version
  - With static binding the selection is done at compile-time
    - Older generations of programming languages (C, Pascal, …) were restricted to static binding
- Static binding is also referred to as “early binding”

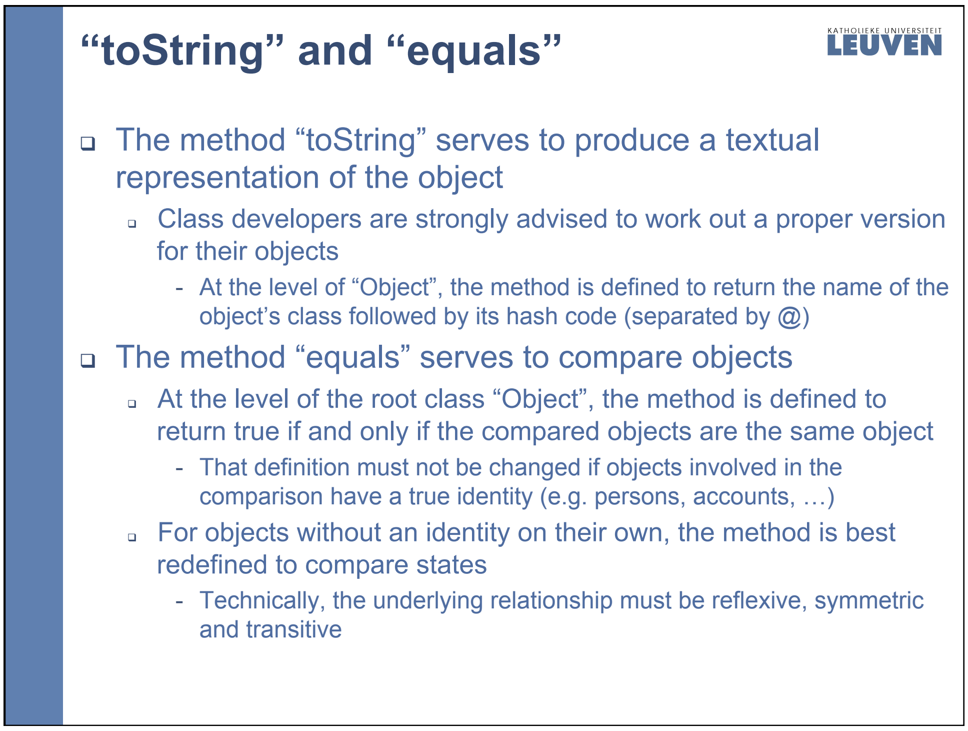
Task 3+4+5



Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

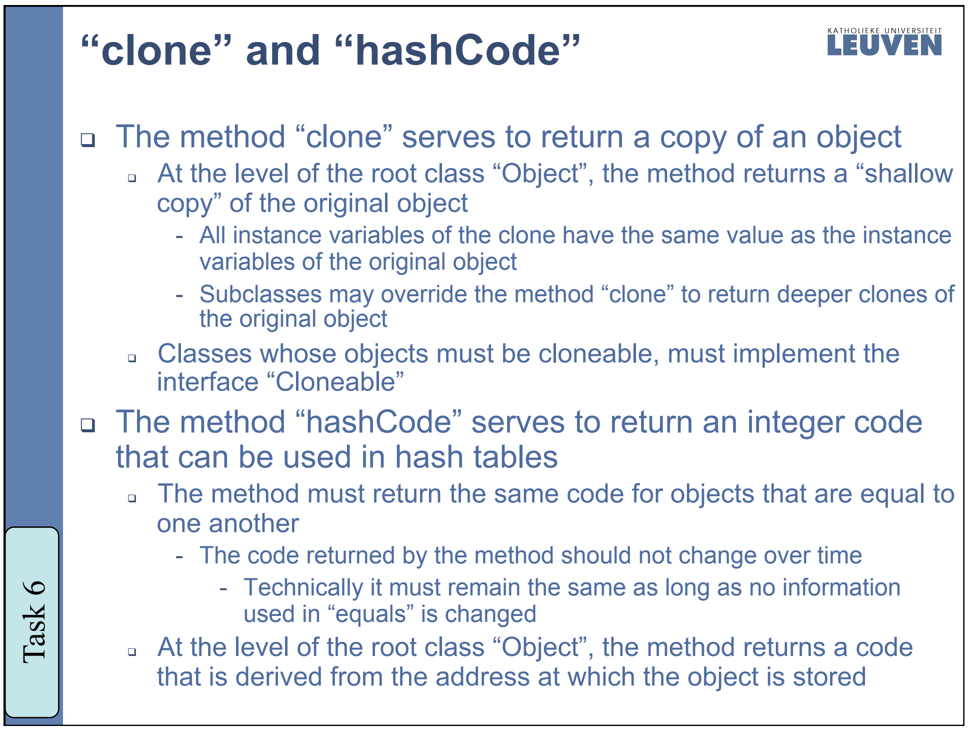
- Inheritance is often used to express that objects of a subclass are a special kind of objects of its superclass
  - Subclasses may be able to work out more specific definitions of methods they inherit from their superclass
- Object-oriented programming languages complement inheritance with dynamic binding
  - Each time an instance method is invoked, the most appropriate version is selected for execution (dynamic binding)
- The root class “Object” introduces a number of methods, that might need a redefinition at the level its subclasses
  - The class “Object” introduces among others the methods “toString”, “equals”, “clone” and “hashCode”



“toString” and “equals”

KATHOLIEKE UNIVERSITEIT LEUVEN

- The method “toString” serves to produce a textual representation of the object
  - Class developers are strongly advised to work out a proper version for their objects
    - At the level of “Object”, the method is defined to return the name of the object’s class followed by its hash code (separated by @)
- The method “equals” serves to compare objects
  - At the level of the root class “Object”, the method is defined to return true if and only if the compared objects are the same object
    - That definition must not be changed if objects involved in the comparison have a true identity (e.g. persons, accounts, …)
  - For objects without an identity on their own, the method is best redefined to compare states
    - Technically, the underlying relationship must be reflexive, symmetric and transitive

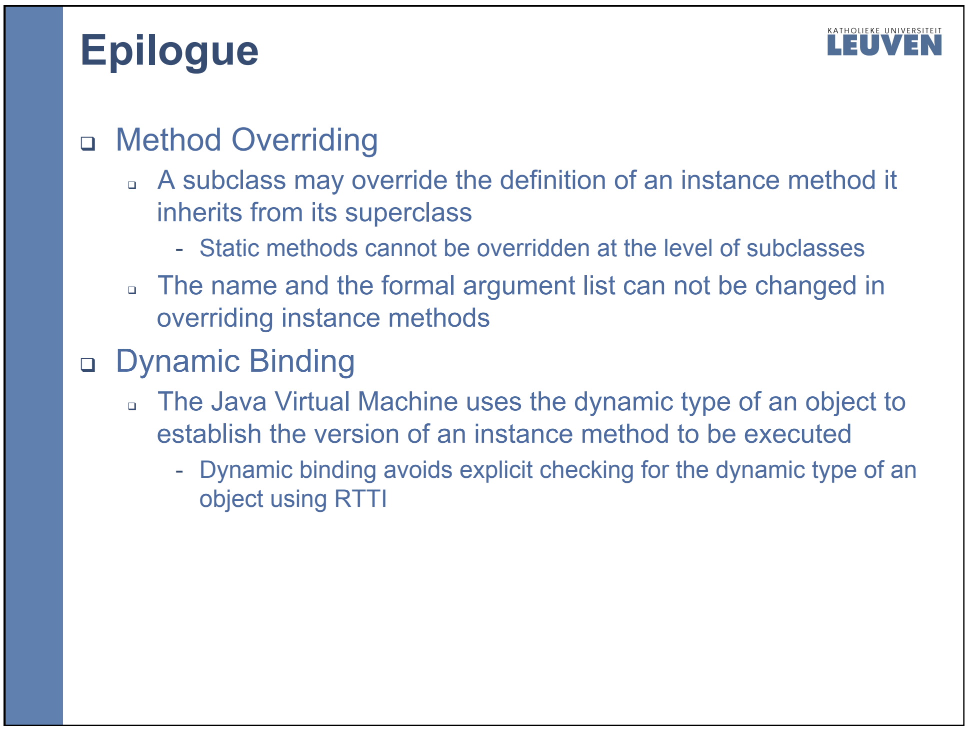


“clone” and “hashCode”

KATHOLIEKE UNIVERSITEIT LEUVEN

Task 6

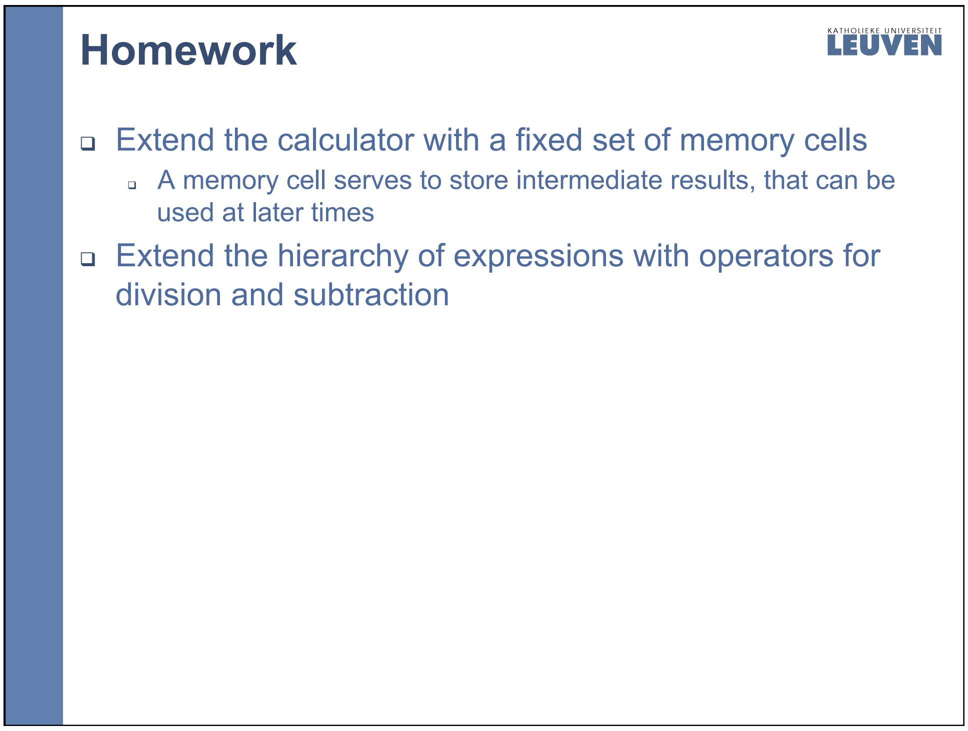
- The method “clone” serves to return a copy of an object
  - At the level of the root class “Object”, the method returns a “shallow copy” of the original object
    - All instance variables of the clone have the same value as the instance variables of the original object
    - Subclasses may override the method “clone” to return deeper clones of the original object
  - Classes whose objects must be cloneable, must implement the interface “Cloneable”
- The method “hashCode” serves to return an integer code that can be used in hash tables
  - The method must return the same code for objects that are equal to one another
    - The code returned by the method should not change over time
      - Technically it must remain the same as long as no information used in “equals” is changed
  - At the level of the root class “Object”, the method returns a code that is derived from the address at which the object is stored



Epilogue

KATHOLIEKE UNIVERSITEIT LEUVEN

- Method Overriding
  - A subclass may override the definition of an instance method it inherits from its superclass
    - Static methods cannot be overridden at the level of subclasses
  - The name and the formal argument list can not be changed in overriding instance methods
- Dynamic Binding
  - The Java Virtual Machine uses the dynamic type of an object to establish the version of an instance method to be executed
    - Dynamic binding avoids explicit checking for the dynamic type of an object using RTTI



Homework

KATHOLIEKE UNIVERSITEIT LEUVEN

- Extend the calculator with a fixed set of memory cells
  - A memory cell serves to store intermediate results, that can be used at later times
- Extend the hierarchy of expressions with operators for division and subtraction