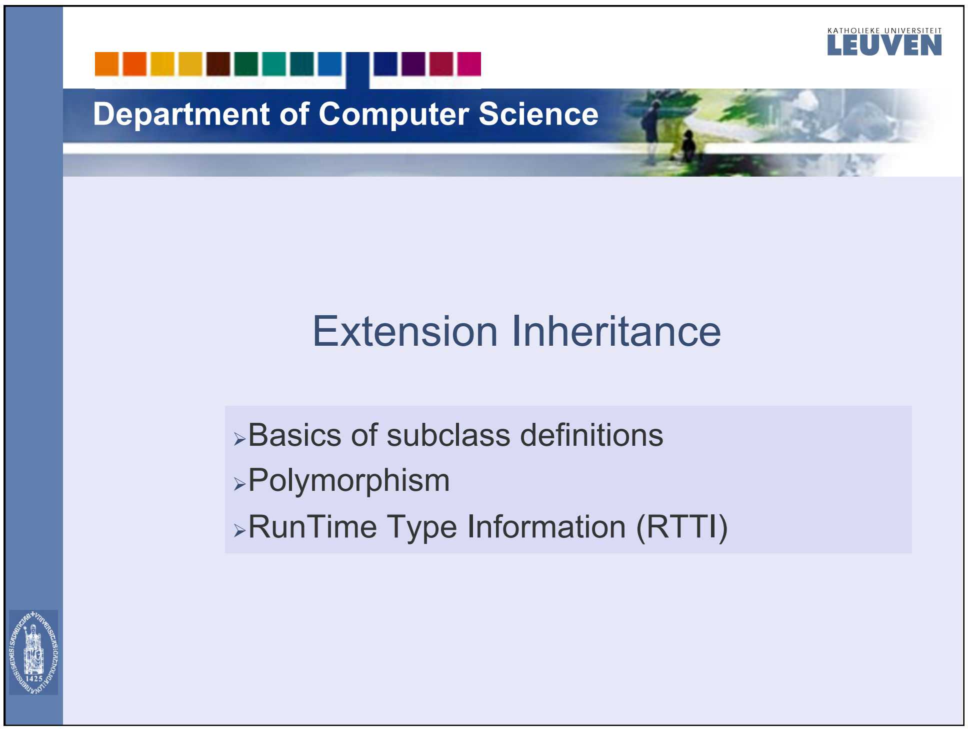
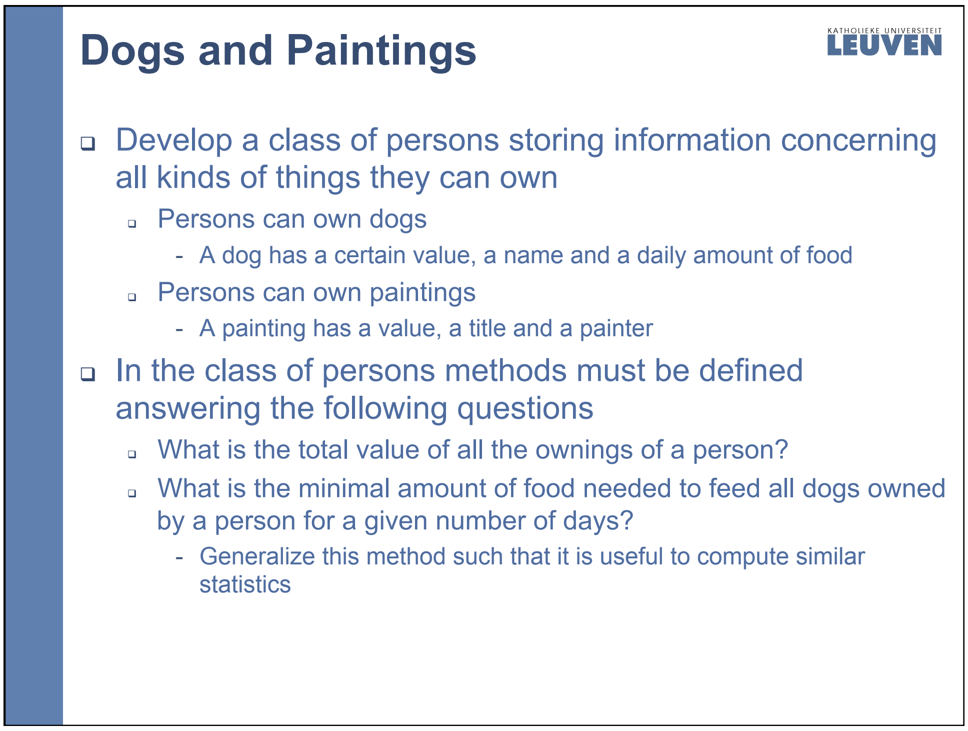


KATHOLIEKE UNIVERSITEIT LEUVEN

Department of Computer Science

Extension Inheritance

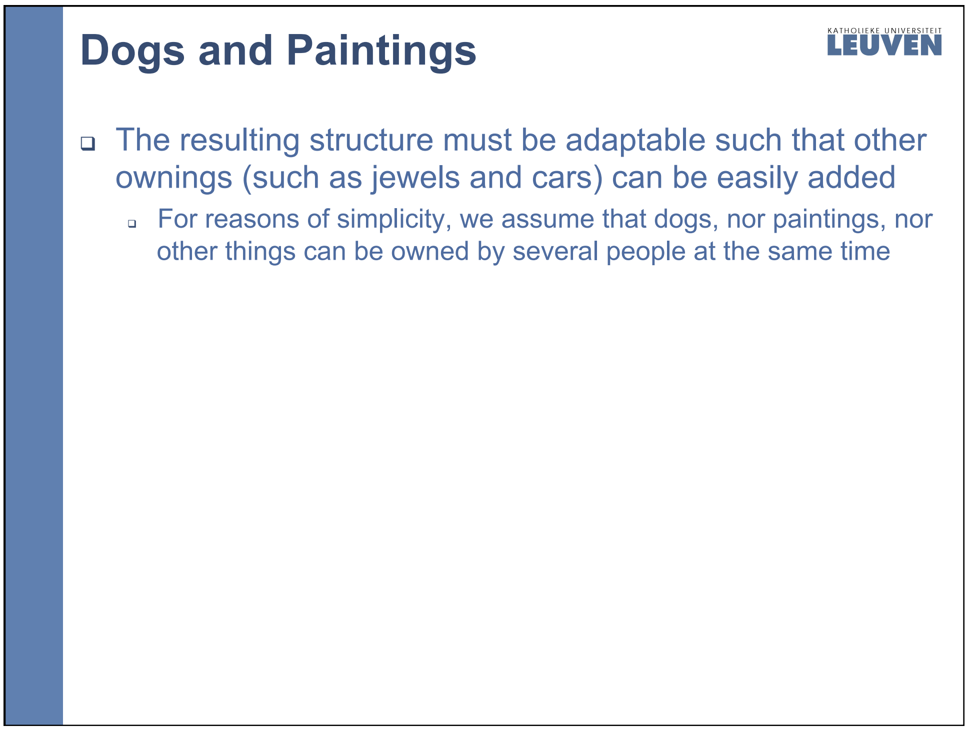
- Basics of subclass definitions
- Polymorphism
- RunTime Type Information (RTTI)



KATHOLIEKE UNIVERSITEIT LEUVEN

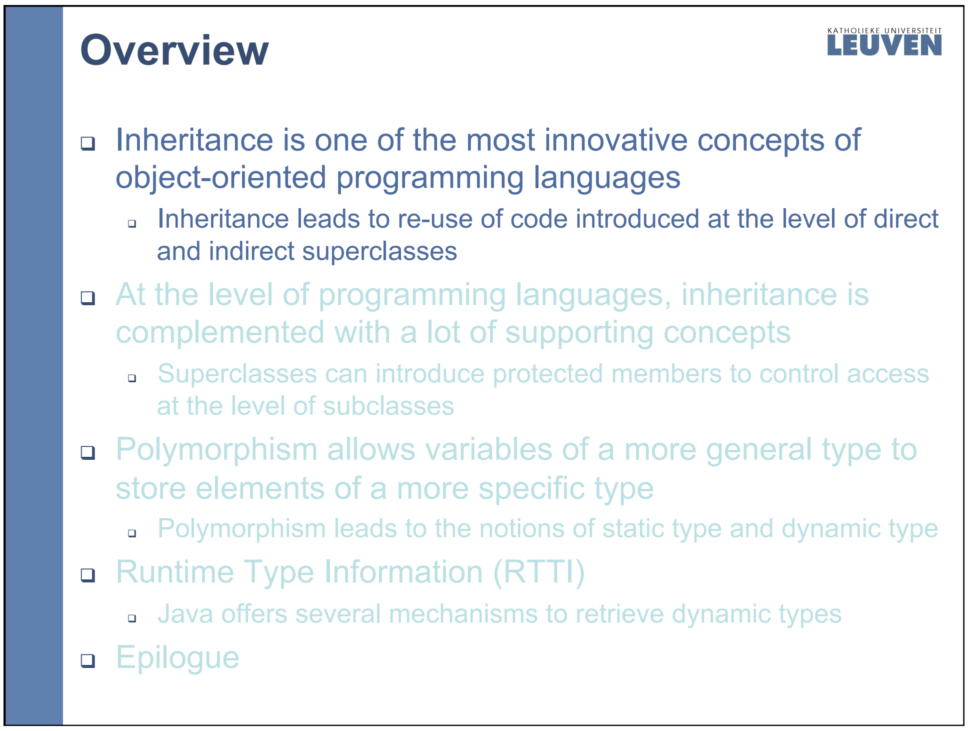
Dogs and Paintings

- Develop a class of persons storing information concerning all kinds of things they can own
  - Persons can own dogs
    - A dog has a certain value, a name and a daily amount of food
  - Persons can own paintings
    - A painting has a value, a title and a painter
- In the class of persons methods must be defined answering the following questions
  - What is the total value of all the ownings of a person?
  - What is the minimal amount of food needed to feed all dogs owned by a person for a given number of days?
    - Generalize this method such that it is useful to compute similar statistics



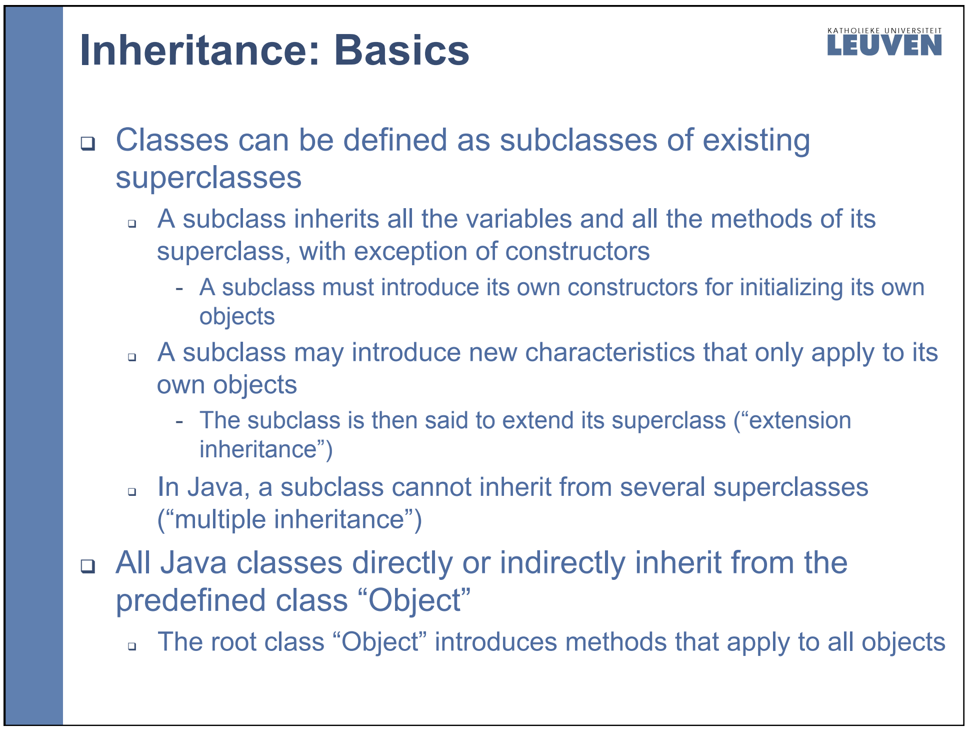
Dogs and Paintings

- The resulting structure must be adaptable such that other ownings (such as jewels and cars) can be easily added
  - For reasons of simplicity, we assume that dogs, nor paintings, nor other things can be owned by several people at the same time



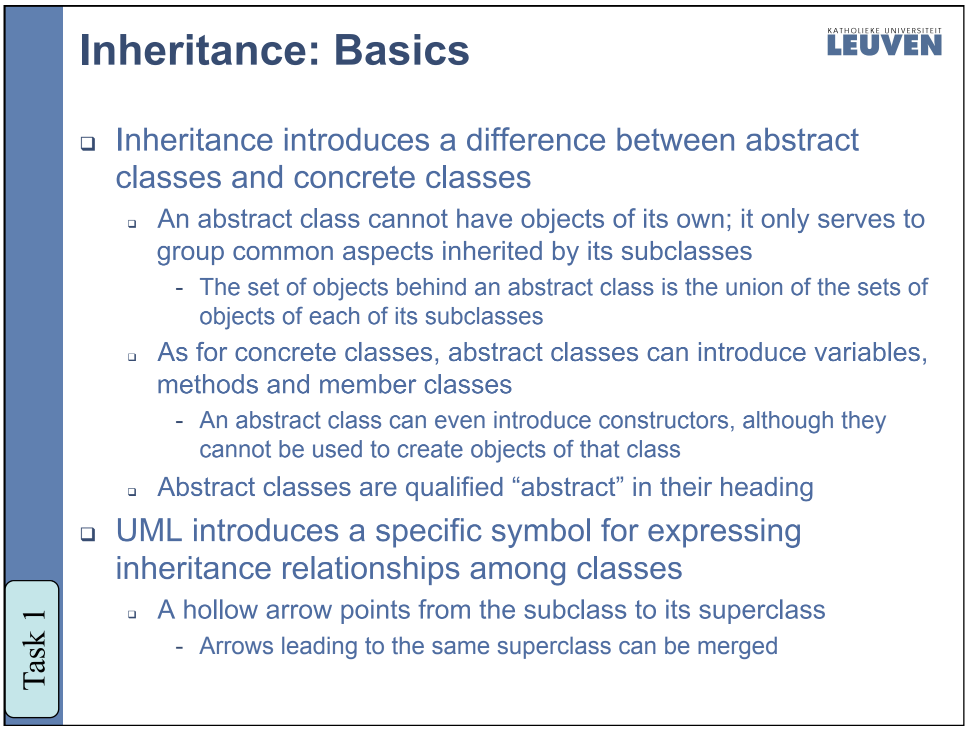
Overview

- Inheritance is one of the most innovative concepts of object-oriented programming languages
  - Inheritance leads to re-use of code introduced at the level of direct and indirect superclasses
- At the level of programming languages, inheritance is complemented with a lot of supporting concepts
  - Superclasses can introduce protected members to control access at the level of subclasses
- Polymorphism allows variables of a more general type to store elements of a more specific type
  - Polymorphism leads to the notions of static type and dynamic type
- Runtime Type Information (RTTI)
  - Java offers several mechanisms to retrieve dynamic types
- Epilogue



Inheritance: Basics

- Classes can be defined as subclasses of existing superclasses
  - A subclass inherits all the variables and all the methods of its superclass, with exception of constructors
    - A subclass must introduce its own constructors for initializing its own objects
  - A subclass may introduce new characteristics that only apply to its own objects
    - The subclass is then said to extend its superclass (“extension inheritance”)
  - In Java, a subclass cannot inherit from several superclasses (“multiple inheritance”)
- All Java classes directly or indirectly inherit from the predefined class “Object”
  - The root class “Object” introduces methods that apply to all objects



Inheritance: Basics

- Inheritance introduces a difference between abstract classes and concrete classes
  - An abstract class cannot have objects of its own; it only serves to group common aspects inherited by its subclasses
    - The set of objects behind an abstract class is the union of the sets of objects of each of its subclasses
  - As for concrete classes, abstract classes can introduce variables, methods and member classes
    - An abstract class can even introduce constructors, although they cannot be used to create objects of that class
  - Abstract classes are qualified “abstract” in their heading
- UML introduces a specific symbol for expressing inheritance relationships among classes
  - A hollow arrow points from the subclass to its superclass
    - Arrows leading to the same superclass can be merged

Task 1

Overview

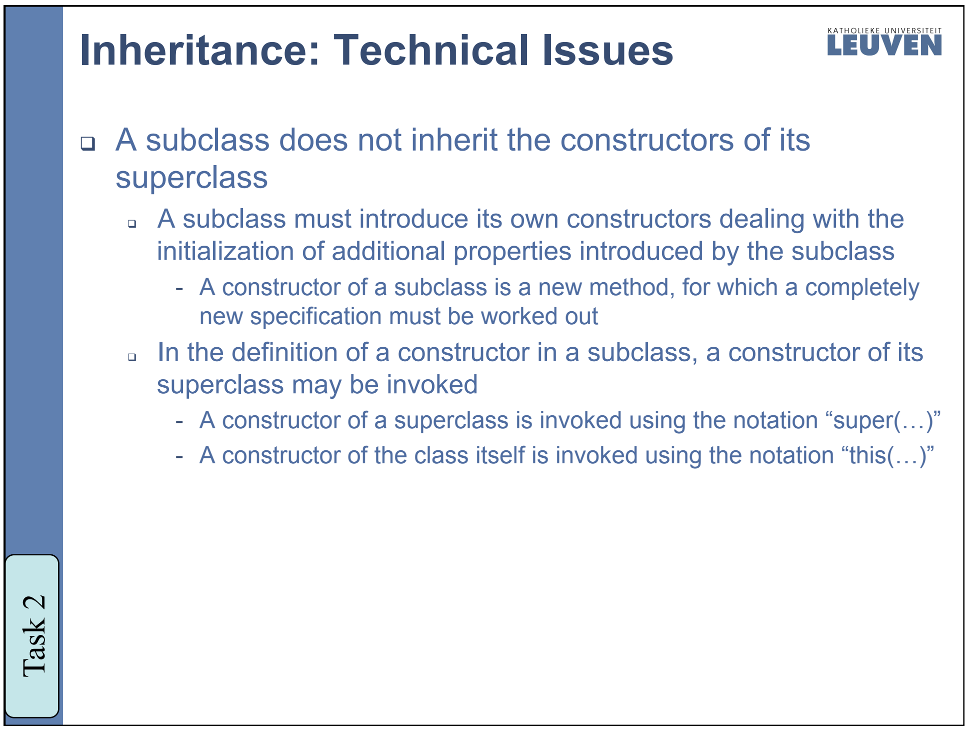
KATHOLIEKE UNIVERSITEIT LEUVEN

- Inheritance is one of the most innovative concepts of object-oriented programming languages
  - Inheritance leads to re-use of code introduced at the level of direct and indirect superclasses
- At the level of programming languages, inheritance is complemented with a lot of supporting concepts
  - Superclasses can introduce protected members to control access at the level of subclasses
- Polymorphism allows variables of a more general type to store elements of a more specific type
  - Polymorphism leads to the notions of static type and dynamic type
- Runtime Type Information (RTTI)
  - Java offers several mechanisms to retrieve dynamic types
- Epilogue

Inheritance: Technical Issues

KATHOLIEKE UNIVERSITEIT LEUVEN

- Protected members of a class are accessible to all its subclasses
  - Protected members are also accessible to the class itself and to classes residing in the same package
    - Ordering of access rights in Java: private, package-accessible, protected, public
- Subclasses will behave as much as possible as ordinary clients of their superclass
  - A subclass will use as much as possible public methods to inspect and mutate inherited characteristics
  - A subclass will never have direct access to the representation established at the level of its superclass

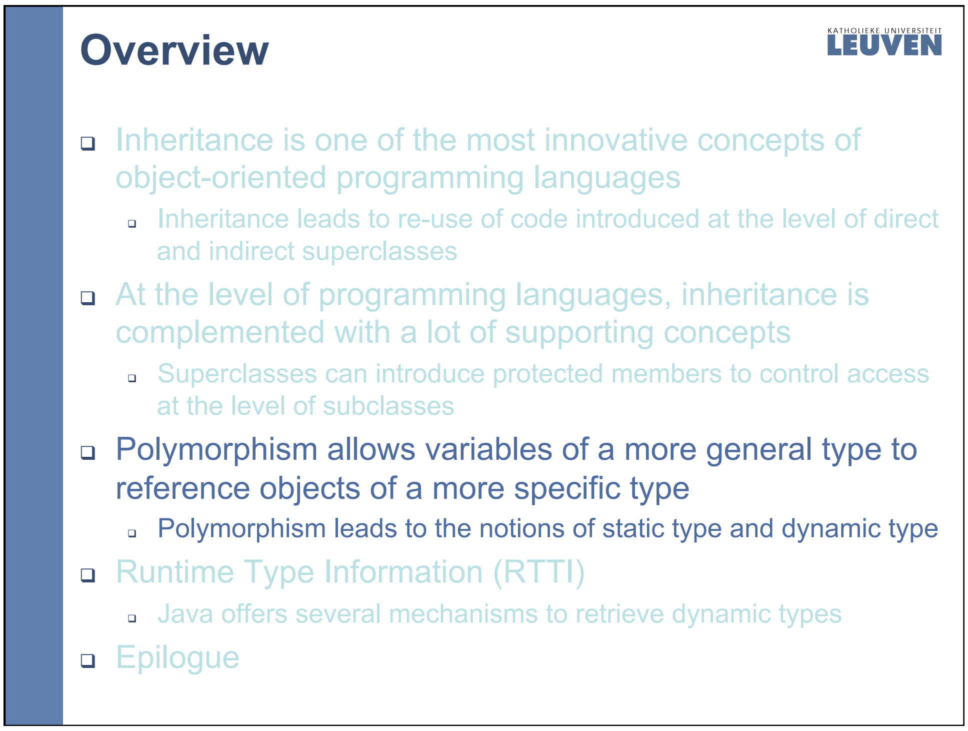


Inheritance: Technical Issues

KATHOLIEKE UNIVERSITEIT LEUVEN

- A subclass does not inherit the constructors of its superclass
  - A subclass must introduce its own constructors dealing with the initialization of additional properties introduced by the subclass
    - A constructor of a subclass is a new method, for which a completely new specification must be worked out
  - In the definition of a constructor in a subclass, a constructor of its superclass may be invoked
    - A constructor of a superclass is invoked using the notation “super(…)”
    - A constructor of the class itself is invoked using the notation “this(…)”

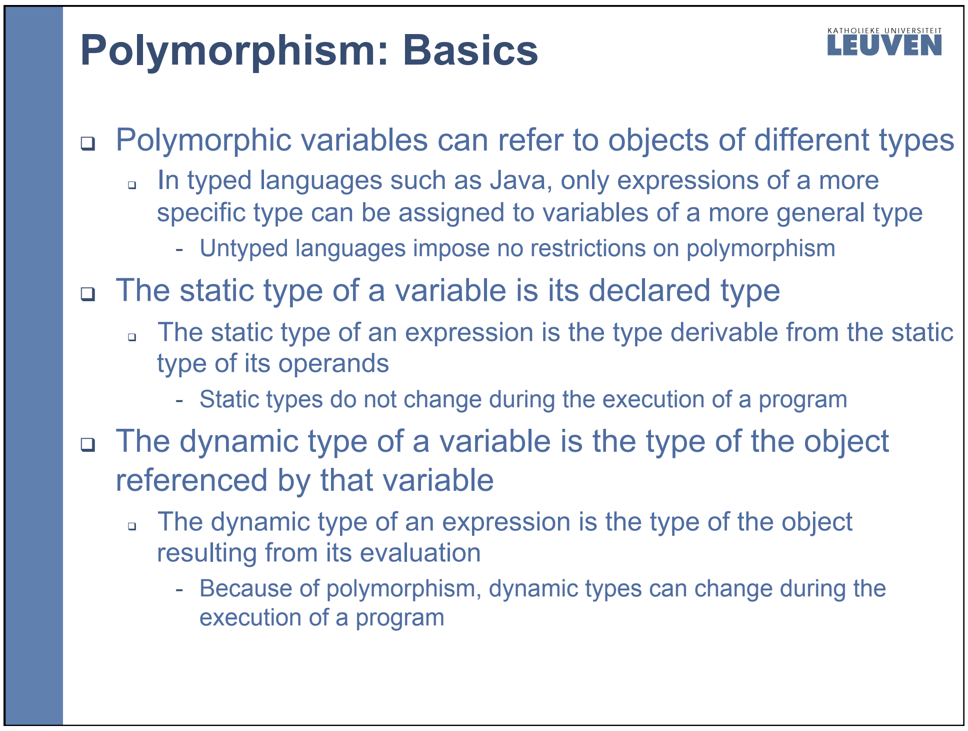
Task 2



Overview

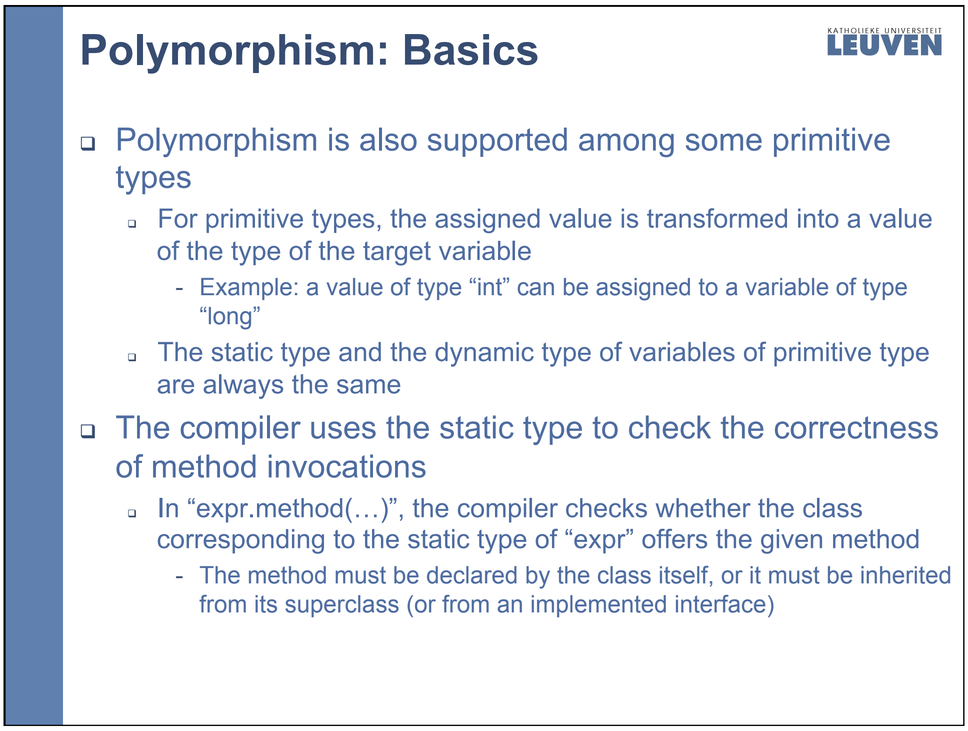
KATHOLIEKE UNIVERSITEIT LEUVEN

- Inheritance is one of the most innovative concepts of object-oriented programming languages
  - Inheritance leads to re-use of code introduced at the level of direct and indirect superclasses
- At the level of programming languages, inheritance is complemented with a lot of supporting concepts
  - Superclasses can introduce protected members to control access at the level of subclasses
- Polymorphism allows variables of a more general type to reference objects of a more specific type
  - Polymorphism leads to the notions of static type and dynamic type
- Runtime Type Information (RTTI)
  - Java offers several mechanisms to retrieve dynamic types
- Epilogue



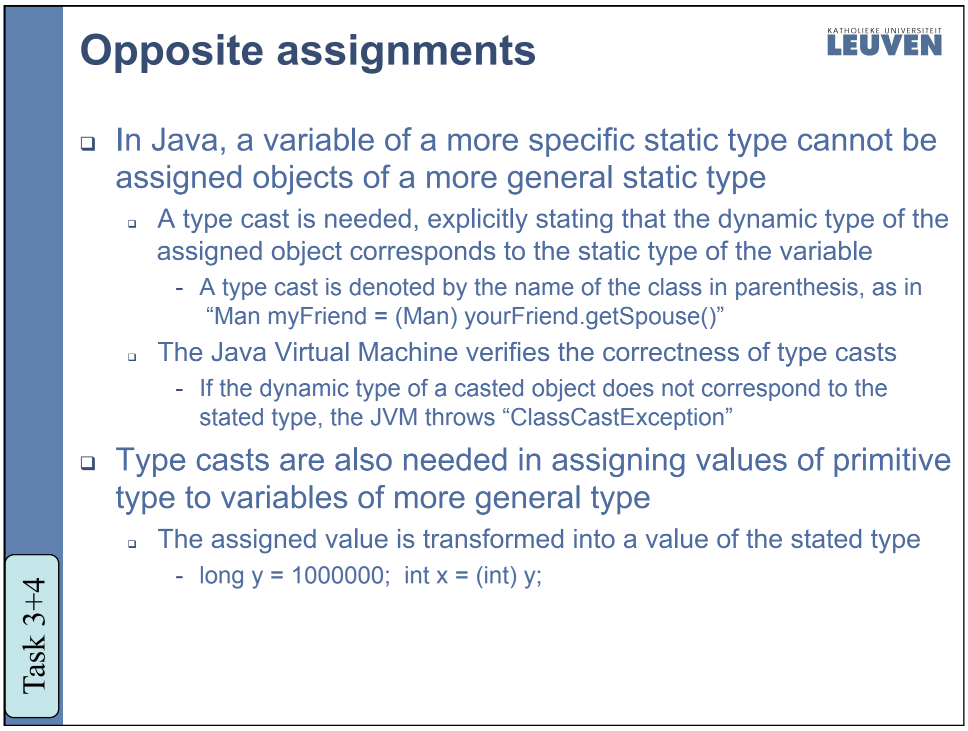
Polymorphism: Basics

- Polymorphic variables can refer to objects of different types
  - In typed languages such as Java, only expressions of a more specific type can be assigned to variables of a more general type
    - Untyped languages impose no restrictions on polymorphism
- The static type of a variable is its declared type
  - The static type of an expression is the type derivable from the static type of its operands
    - Static types do not change during the execution of a program
- The dynamic type of a variable is the type of the object referenced by that variable
  - The dynamic type of an expression is the type of the object resulting from its evaluation
    - Because of polymorphism, dynamic types can change during the execution of a program



Polymorphism: Basics

- Polymorphism is also supported among some primitive types
  - For primitive types, the assigned value is transformed into a value of the type of the target variable
    - Example: a value of type “int” can be assigned to a variable of type “long”
  - The static type and the dynamic type of variables of primitive type are always the same
- The compiler uses the static type to check the correctness of method invocations
  - In “expr.method(…)”, the compiler checks whether the class corresponding to the static type of “expr” offers the given method
    - The method must be declared by the class itself, or it must be inherited from its superclass (or from an implemented interface)

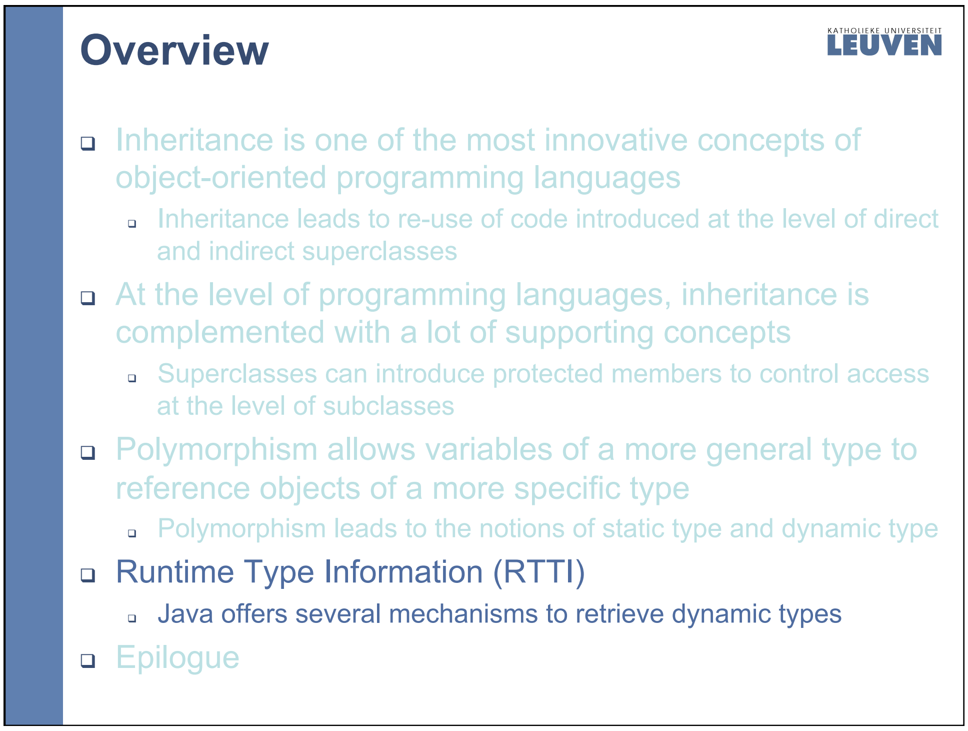


Opposite assignments

KATHOLIEKE UNIVERSITEIT LEUVEN

- In Java, a variable of a more specific static type cannot be assigned objects of a more general static type
  - A type cast is needed, explicitly stating that the dynamic type of the assigned object corresponds to the static type of the variable
    - A type cast is denoted by the name of the class in parenthesis, as in "Man myFriend = (Man) yourFriend.getSpouse()"
  - The Java Virtual Machine verifies the correctness of type casts
    - If the dynamic type of a casted object does not correspond to the stated type, the JVM throws "ClassCastException"
- Type casts are also needed in assigning values of primitive type to variables of more general type
  - The assigned value is transformed into a value of the stated type
    - long y = 1000000; int x = (int) y;

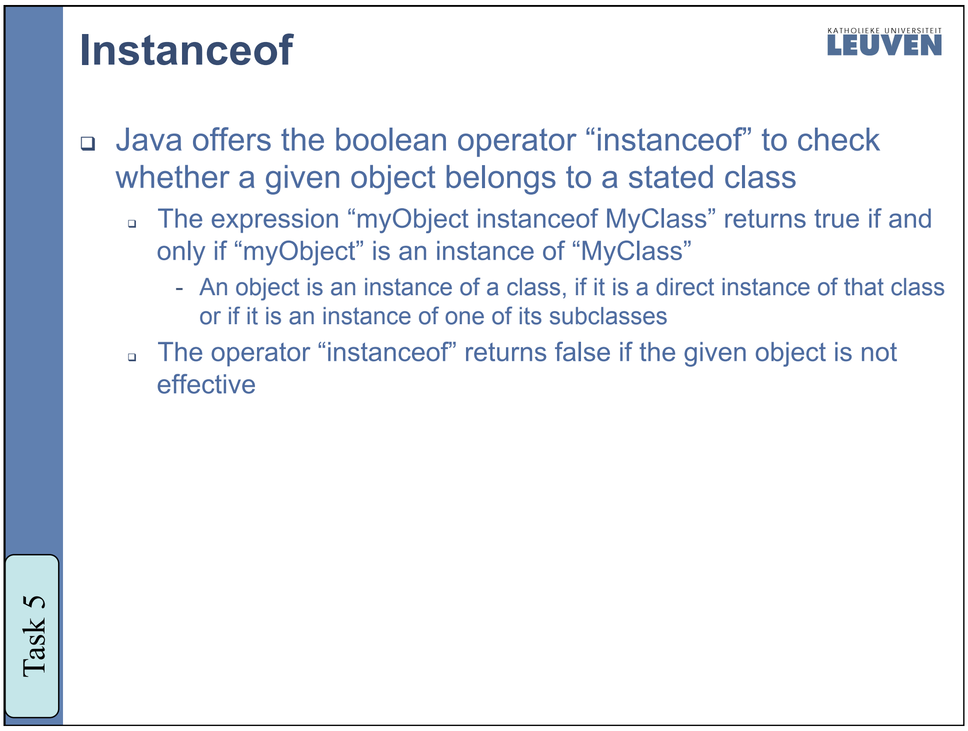
Task 3+4



Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

- Inheritance is one of the most innovative concepts of object-oriented programming languages
  - Inheritance leads to re-use of code introduced at the level of direct and indirect superclasses
- At the level of programming languages, inheritance is complemented with a lot of supporting concepts
  - Superclasses can introduce protected members to control access at the level of subclasses
- Polymorphism allows variables of a more general type to reference objects of a more specific type
  - Polymorphism leads to the notions of static type and dynamic type
- Runtime Type Information (RTTI)
  - Java offers several mechanisms to retrieve dynamic types
- Epilogue

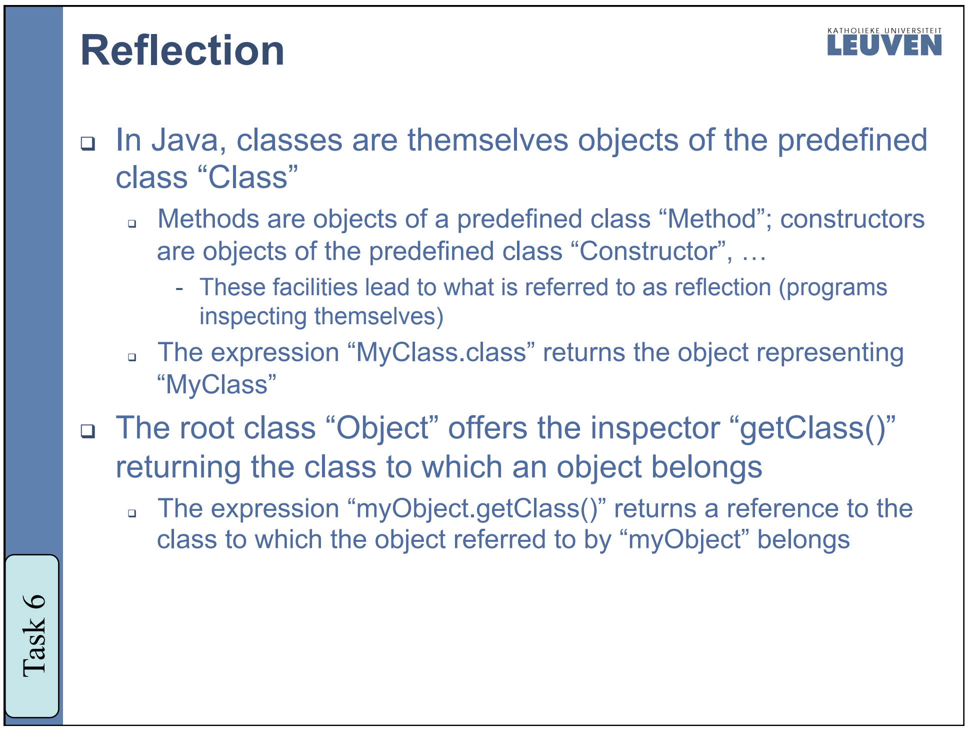


Instanceof

KATHOLIEKE UNIVERSITEIT LEUVEN

- Java offers the boolean operator "instanceof" to check whether a given object belongs to a stated class
  - The expression "myObject instanceof MyClass" returns true if and only if "myObject" is an instance of "MyClass"
    - An object is an instance of a class, if it is a direct instance of that class or if it is an instance of one of its subclasses
  - The operator "instanceof" returns false if the given object is not effective

Task 5



Reflection

KATHOLIEKE UNIVERSITEIT LEUVEN

- In Java, classes are themselves objects of the predefined class "Class"
  - Methods are objects of a predefined class "Method"; constructors are objects of the predefined class "Constructor", …
    - These facilities lead to what is referred to as reflection (programs inspecting themselves)
  - The expression "MyClass.class" returns the object representing "MyClass"
- The root class "Object" offers the inspector "getClass()" returning the class to which an object belongs
  - The expression "myObject.getClass()" returns a reference to the class to which the object referred to by "myObject" belongs

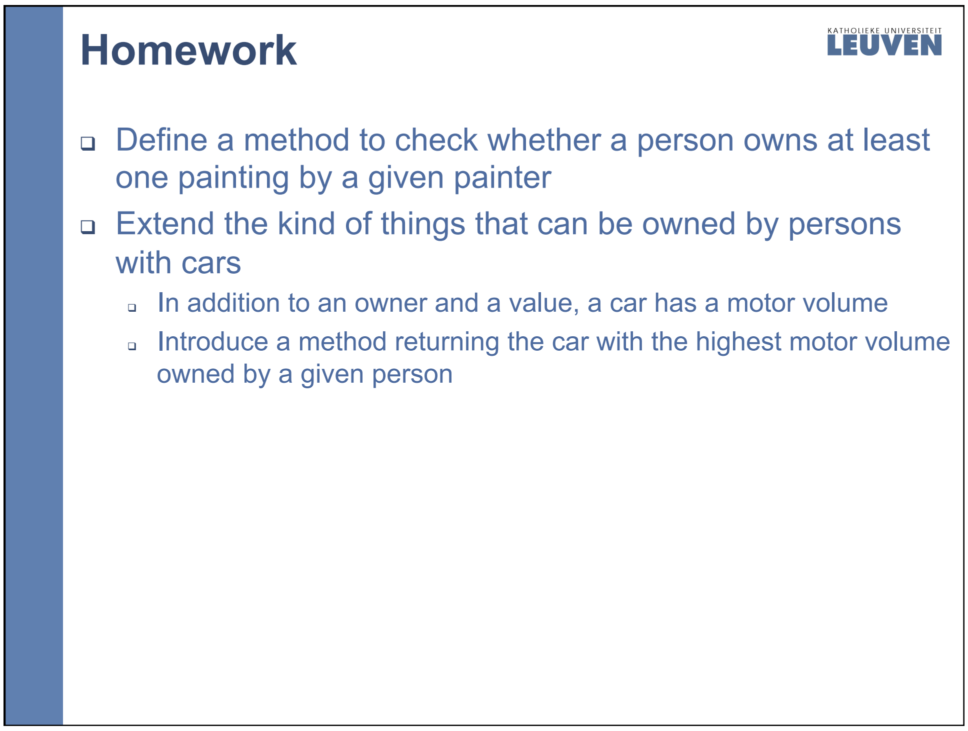
Task 6

Overview

- Inheritance is one of the most innovative concepts of object-oriented programming languages
  - Inheritance leads to re-use of code introduced at the level of direct and indirect superclasses
- At the level of programming languages, inheritance is complemented with a lot of supporting concepts
  - Superclasses can introduce protected members to control access at the level of subclasses
- Polymorphism allows variables of a more general type to store elements of a more specific type
  - Polymorphism leads to the notions of static type and dynamic type
- Epilogue

Summary

- Inheritance
  - Subclasses inherit all instance variables and instance methods from their superclass
    - Subclasses do not inherit constructors from their superclass
- Polymorphism
  - Variables of a more general type can be assigned objects of a more specific type
    - The static type of a variable is its declared type; the dynamic type of a variable is the type of the stored object
- Runtime Type Information (RTTI)
  - Type casts are explicit statements concerning the dynamic type of variables and expressions
  - The boolean operator “instanceof” checks whether a given object directly or indirectly belongs to a given class
  - The instance method “getClass()” returns the class to which an object belongs



Homework

KATHOLIEKE UNIVERSITEIT LEUVEN

- Define a method to check whether a person owns at least one painting by a given painter
- Extend the kind of things that can be owned by persons with cars
  - In addition to an owner and a value, a car has a motor volume
  - Introduce a method returning the car with the highest motor volume owned by a given person