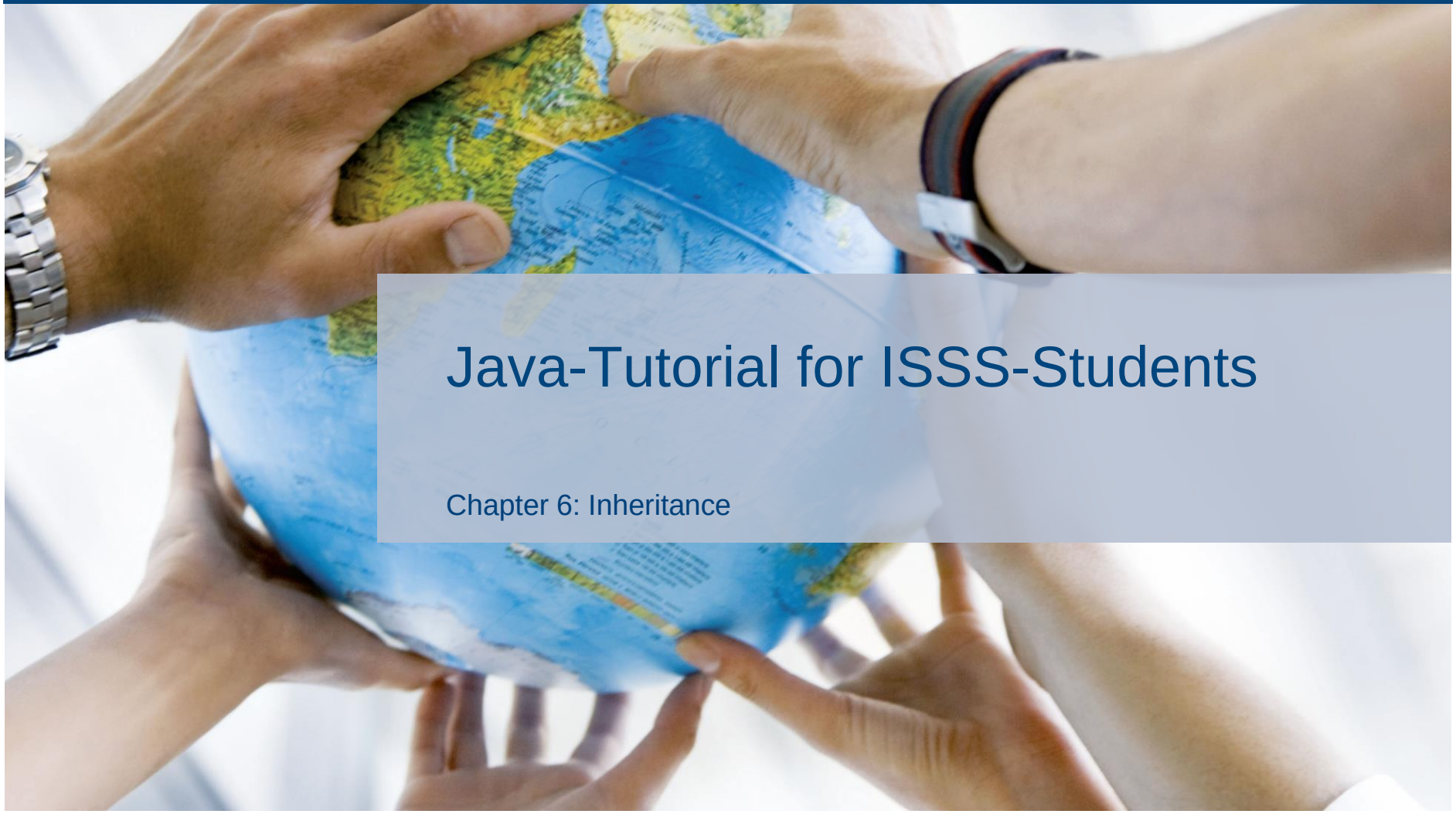
University of Bamberg



Java-Tutorial for ISSS-Students

Chapter 6: Inheritance

Chapter 6: Inheritance

1. Superclasses and subclasses
2. Inheritance and constructors
3. Inheritance and casting

Superclasses and subclasses

- If a class is derived from another class, it inherits all non-private fields and methods from the superclass:
  - *public class SuperClass { … }*
  - *public class SubClass extends SuperClass { … }*
- In the subclass you can add new fields and methods or overwrite inherited methods to specify their functionality
  - The return value has to be compatible with the return value of the overwritten method, the parameters have to be exactly the same
  - If you don‘t want a method to be overwritten, use the modifier *final* (this works for whole classes too)

Inheritance and constructors

- As soon as you implement your own constructor in a class, the default-constructor is not available any more
- This affects every single subclass too, so that they can‘t be compiled any more – you can solve this problem by:
  1. … writing a new default-constructor for the subclass that uses the constructor of the superclass with values = null
  2. … writing an additional default constructor for the superclass so that it can be used again
  3. … writing a new constructor for the subclass which uses the same (or more) parameters than the constructor of the superclass

Inheritance and casting

- *SuperClass variableName = new SubClass();*
  - The object that is referenced has the type *SubClass*, but the reference itself considers it to be an object of the type *SuperClass* – therefore it cannot invoke methods that are only available for objects of the type *SubClass*
  - This problem can be solved by *casting* the reference type of the object into the type *SubClass* (as the object itself already is an object of the type *SubClass* – and only in this case the *casting* is possible):
    - *SubClass newVariableName = (SubClass) variableName;*
  - Besides this way of *downcasting* there is the possibility of *upcasting* an object of the type *SubClass* into the reference type *SuperClass*:
    - *SuperClass veryNewVariableName = newVariableName;*
  - Before the casting you should use the *instanceof*-operator to check if the data types match properly (*downcasting* only)

Task: Band 2.0

- Implement a superclass for all band members and a subclass for each single member!
  - The vocalist has a name, an age, a heigth and a pitch of voice and can play music (“Lalala”) and announce the next song
  - The guitarist has a name, an age, a heigth and a guitar and can play music (“Guitar sound”)
  - The bassist has a name, an age, a heigth and a bass and can play music (“Bass sound”)
  - The drummer has a name, an age, a heigth and drums and can play music (“Drum sound”)
- Bonus task: implement a superclass for all instruments and try to imagine possible variables and methods – then implement a subclass for each instrument!