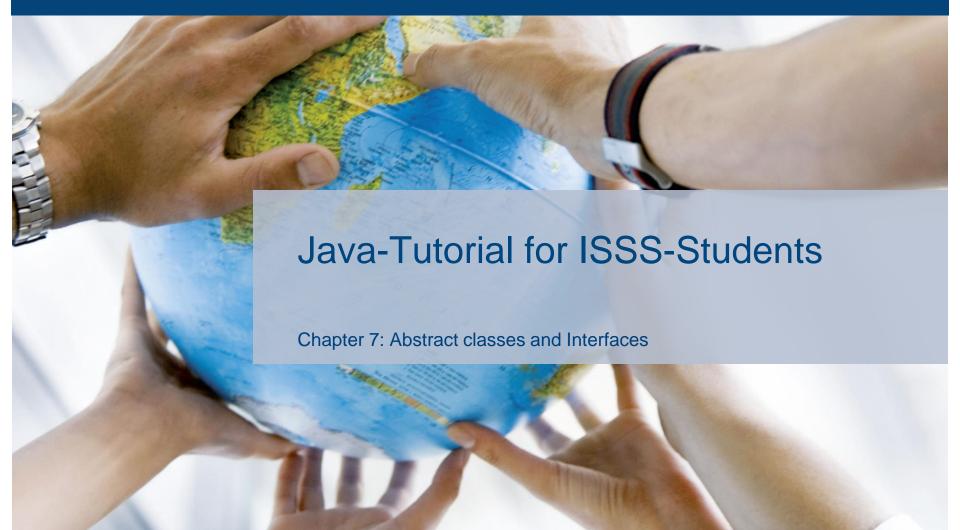
University of Bamberg







Chapter 7: Abstract classes and Interfaces

- Abstract classes
- 2. Interfaces
- 3. Constants

Warm-up: Animals

- Create the package *chapter7* with the following classes:
 - Class Animal with the attributes name (String), age (int) and height (double) and
 the method travelDistance() which takes a distance (double) as parameter and
 returns the time needed to travel the distance (double) return 0 as we don't know
 yet how to travel distances + constructor, getters, setters, toString
 - Class Horse (extends Animal) with the additional attribute runningSpeed (double) and the method runDistance() which takes a distance (double) as parameter and returns the time needed to travel the distance (double) + constructor, getters, setters, toString
 - Class Bird (extends Animal) with the additional attribute flyingSpeed (double) and the method flyDistance() which takes a distance (double) as parameter and returns the time needed to travel the distance (double) + constructor, getters, setters, toString
 - Class Main with the main-method which creates a new object of Animal, Horse and Bird and lets them travel/ run/ fly a distance of 20
 - Assume that distance is given in kilometers and runningSpeed/flyingSpeed in km/h

Abstract classes

- In a hierarchy of classes, sometimes it doesn't make sense that all classes can be instantiated
 - If you add the modifier abstract to the class declaration, the class cannot be instantiated any more
- In an abstract class, it is also possible to modify single methods as abstract.
 - modifier abstract returnValue methodName(parameters);
 - These methods cannot be implemented by the abstract class itself, but must be implemented by any subclass (unless the subclass is abstract itself)

Interfaces

- In Java Interfaces offer a way to define what instances of a certain category should be able to do without implementing this functionality themselves
 - Interfaces cannot be instantiated and have no constructor
 - All methods in Interfaces have to be abstract
 - Classes that claim to implement an Interface have to implement all abstract methods the Interface requires:
 - ➤ public class ClassName implements InterfaceName { ... }



Task: Pegasus

Add the attribute flyingMode (boolean) to the class
 Pegasus – override the travelDistance()-method so that
 it returns the time needed to run or to fly the given
 distance depending on wether the flyingMode of the
 Pegasus is active or not

Interfaces 2

- Unlike abstract classes, Interfaces can be used in more than one class hierarchy
- While a class can only extend one superclass, it can implement more than one Interfaces – therefore it can be used in any context where one of its Interfaces is required
- Interfaces can be structured in hierarchies of inheritance just like normal classes:
 - public interface SubInterface extends SuperInterface { ... }
- In the course of software engineering processes, Interfaces work like contracts between the involved parties that specify what functionality a certain software should provide

Task: Superheroes

- Create the following classes:
 - Class SuperHero with the attributes name (String) and strength (int) + constructor, getters, setters, toString
 - Class FlyingSuperHero (extends SuperHero) with the additional attributes flyingSpeed (double) and hasTeleportationPower (boolean) which implements the Interface FlyingCreature if the FlyingSuperHero has teleportation power, the time needed to fly a distance is 0, otherwise it depends on the flying speed (like in the Pegasus-class) + constructor, getters, setters, toString

Constants

- Constants can be defined in Interfaces (all variables are automatically public, static and final there)
- Better style: put them in their own constant classes
 - Declare the constructor of the class private and all variables public static final
 - Import the constant class into your other classes to use the constants or use static import for single constants
- Even better style: *Enums* to define groups of constants:
 - public enum EnumName { CONSTANT1, CONSTANT2, ... CONSTANTX }
 - Enums can have a constructor (private) and methods