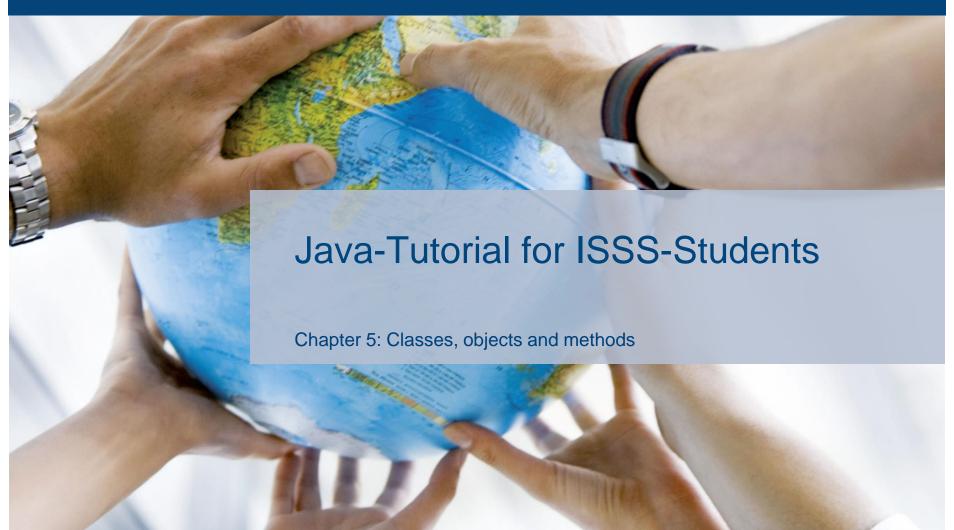
#### University of Bamberg







# Chapter 5: Classes, objects and methods

- 1. Using methods
- 2. Commenting the code
- 3. Classes and objects
- 4. Getters and setters
- 5. Constructors
- 6. The toString-method

## Using methods

- The main-method is the starting point of the program the rest of the functionality of the program should be implemented in separate methods
- Methods are created as follows:
  - modifiers returnValue methodName(parameters) {code}



## Commenting the code

```
* What does the method do?

* * @param firstParameter what is the first Parameter?

* @param secondParameter what is the second Parameter?

*...

* @param nthParameter what is the n-th Parameter?

* @return what is the return value?

*/

modifiers returnValue methodName (parameters) {code}
```



### Classes and objects

- In Java each object belongs to a class
- A class can be described as a category or pattern of how objects of this class should be constituted – an object is a concrete instance of this pattern

# Classes and objects 2

- Classes contain variables, which describe the attributes of objects of a certain class, and methods, which describe the functionality of these objects
  - If variables or methods contain the modifier static, these are class variables/ class methods, which are common to all objects of a class and can even be used without instantiating the class in concrete objects
  - Otherwise they are instance variables/ instance methods: the instance variables contain individual values for each object and describe its current state; instance methods can only be called by concrete objects of the class



# Controlling access to members of a class

Modifier	Same class	Package	Subclass	Other class
public	yes	yes	yes	yes
protected	yes	yes	yes	no
no modifier	yes	yes	no	no
private	yes	no	no	no

#### Getters and setters

- To control the access to *private* variables, use *getter*and *setter*-methods:
  - public dataTypeOfX getX() { return x }
  - public void setX(newValueOfX) { x = newValueOfX }
  - Getters and setters enable us to validate the parameters
- If a parameter has the same name as a variable of the class, use this.variableName to show that the class/ instance variable is meant:
  - public void setX(x) { this.x = x }

#### Constructors

- In addition to using the default constructor, it is possible to implement your own constructors:
  - public ClassName(parameters) { code }
- With constructors you can set default values for certain variables or build a new object out of given parameters
- The default constructor of the superclass (which is by default java.lang.Object) is automatically invoked by every constructor
  - See chapter 6

# The toString-method

- By default, an object itself is represented by its memory address, which is not interpretable by human viewers
- The toString-method of a class replaces this standardrepresentation by a String-representation of objects of this class:
  - public String toString() { return ... }