Objects First with Java
A Practical Introduction using BlueJ

David J. Barnes
Michael Kölling
Extensions by H.-J. Bungartz and T. Neckel
Course Contents

• Introduction to object-oriented programming…
• …with a strong software engineering foundation (cf. SWE course in CSE)…
• …aimed at producing and maintaining large, high-quality software systems…
• …but starting with the first steps.
Buzzwords to Come

- responsibility-driven design
- encapsulation
- inheritance
- coupling
- overriding
- cohesion
- interface
- iterators
- javadoc
- collection classes
- mutator methods
- polymorphic method calls
Goals

• Sound knowledge of programming principles
• Sound knowledge of object-orientation
• Ability to critically assess the quality of a (small) software system
• Ability to implement a small software system (not just a program ...) in Java
David J. Barnes & Michael Kölling

Objects First with Java
A Practical Introduction using BlueJ

Fourth edition,
Pearson Education, 2008
Further Literature

CSE Course Advanced Programming link:

http://www5.in.tum.de/wiki/index.php/Advanced_Programming_-_Winter_10

More references will be given in the Exercises!
Course Overview (1)

- Objects and classes
- Understanding class definitions
- Object interaction
- Grouping objects
- More sophisticated behaviour – libraries
- Well-behaved objects – testing, maintaining, debugging
Course Overview (2)

• Designing classes
• Inheritance
• Polymorphism
• Extendible, flexible class structures
• Building graphical user interfaces
• Handling errors
• Designing applications
Fundamental Concepts

- object
- class
- method
- parameter
- data type
Objects and Classes

• objects
  - represent ‘things’ from the real world, or from some problem domain
    (ex.: “the red car there in the car park”)

• classes
  - represent all objects of a kind
    (ex.: “car”: a vehicle with four wheels and an engine, that can carry a small number of passengers, cf. LONGMAN)
Methods and Parameters

- **objects** have operations which can be invoked (Java calls them *methods*)

- **methods**
  - may have parameters to pass additional information needed to execute
  - are no independent entities, but directly related to objects
Data Types

• specify what kind of data can be passed to a parameter

• examples:
  - integer – int (4, 6, 0, …)
  - character strings – string (“red”, “BPX”)
  - …
Other Observations

• many *instances* can be created from a single class

• an object has *attributes*: values stored in *fields*

• the class defines what fields an object has, but each object stores its own set of values (the *state* of the object)
State
Two Circle Objects

- circle 1: Diameter 50, xPosition 80, yPosition 30, color "blue", isVisible true
- circle 2: Diameter 30, xPosition 230, yPosition 75, color "red", isVisible true
Source Code

• ... in each class (Java code)
• ... defines details of a class (fields and methods).
Return Values

- Methods may return a result via a return value.
- Example: Parabola
  \[ y = -2.0 \times x^2 + 10.0 \times x - 4.5; \]
Java Reserved Words

abstract continue for new switch
assert default * goto package synchronized
boolean do if implements this
break double import private throw
byte else instanceof public throws
case enum * return transient
catch extends int short try
char finally interface static void
class float long strictfp volatile
const native super while

* not used ** added in java 1.2 *** added in java 1.4 **** added in java 5.0