Objects First with Java

A Practical Introduction using BlueJ

David J. Barnes
Michael Kölling

Take control of your own learning

- Lecture
- Classes
- Exercises
- Book
- Web page
- Discussion forum
- Study groups
- Practice, practice, practice!

Course Contents

- Introduction to object-oriented programming…
- …with a strong software engineering foundation…
- …aimed at producing and maintaining large, high-quality software systems.

Buzzwords

responsibility-driven design

inheritance

encapsulation

iterators

overriding

coupling

cohesion

javadoc

interface

collection classes

mutator methods

polymorphic method calls

Goals

- Sound knowledge of programming principles
- Sound knowledge of object-orientation
- Able to critically assess the quality of a (small) software system
- Able to implement a small software system in Java

Book

David J. Barnes & Michael Kölling

Objects First with Java
A Practical Introduction using BlueJ

5th edition,
Pearson Education, 2012
ISBN 0-13-249266-0
978-0-13-249266-9

Course overview (1)

- Objects and classes
- Understanding class definitions
- Object interaction
- Grouping objects
- More sophisticated behavior - libraries
- Designing classes
- Well-behaved objects - testing, maintaining, debugging

Course overview (2)

- Inheritance
- Polymorphism
- Extendable, flexible class structures
- Building graphical user interfaces
- Handling errors
- Designing applications

Demo

Fundamental concepts

- object
- class
- method
- parameter
- data type

Objects and classes

- objects
  - represent ‘things’ from the real world, or from some problem domain (example: “the red car down there in the car park”)
- classes
  - represent all objects of a kind (example: “car”)

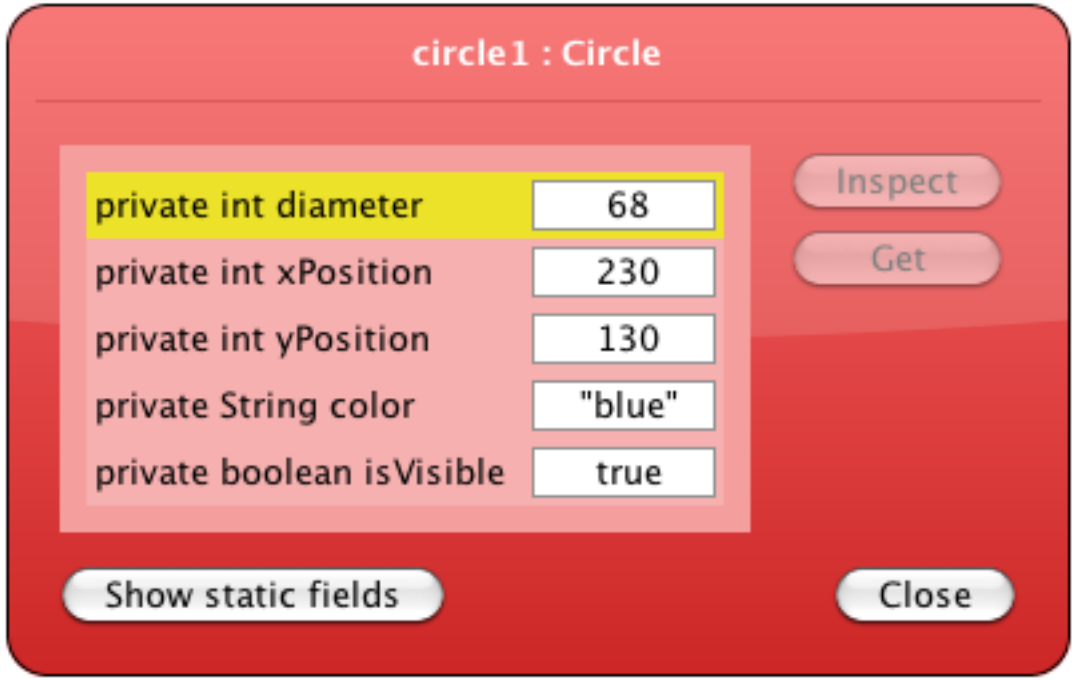
Methods and parameters

- Objects have operations which can be invoked (Java calls them *methods*).
- Methods may have parameters to pass additional information needed to execute.

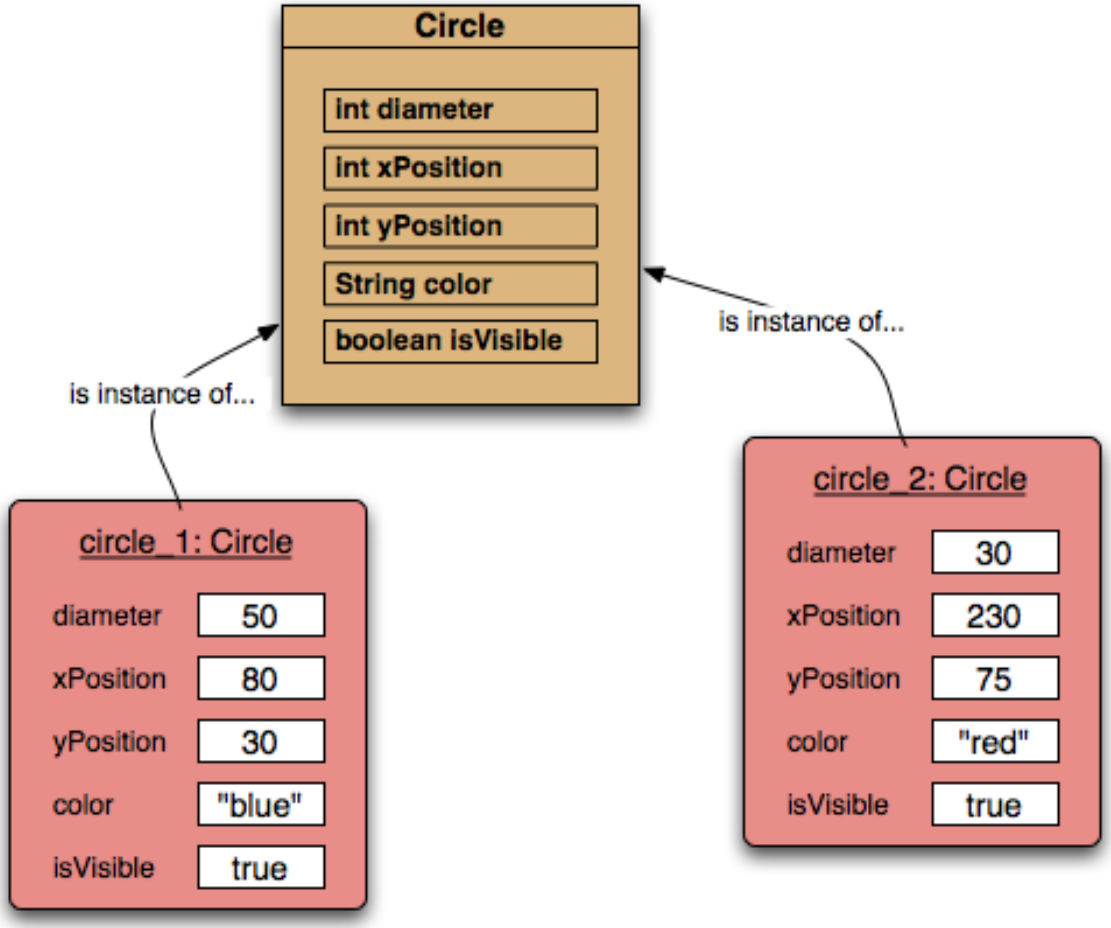
Other observations

- 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


circle1 : Circle
private int diameter
68
private int xPosition
230
private int yPosition
130
private String color
"blue"
private boolean isVisible
true
Inspect
Get
Show static fields
Close

Two circle objects



Source code

- Each class has source code (Java code) associated with it that defines its details (fields and methods).

Return values

- All the methods in the *figures* project have **`void`** return types; but …
- … methods may return a result via a return value.
- Such methods have a non-**`void`** return type.
- More on this in the next chapter.