Objects First with JAVA

A Practical Introduction using BlueJ

DAVID J. BARNES • MICHAEL KÖLLING

Produced by: Ms. Mairead Meagher,
Ms. Siobhán Roche.

Department of Computing and Mathematics
http://www.setu.ie/

Course Contents

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

Terminology

inheritance

responsibility-driven design

encapsulation

iterator

overriding

coupling

abstraction

interface

cohesion

javadoc

class

collection class

mutator

lambda

stream

instance

polymorphic method call

Goals

- Sound knowledge of programming principles
- Sound knowledge of object-orientation
- Able to implement a small software system in Java

Book

David J. Barnes & Michael Kölling

Objects First with Java
A Practical Introduction using BlueJ

7th edition, Pearson Education, 2025

Overview

- Objects and classes
- Understanding class definitions
- Object interaction
- Grouping objects
- Functional programming style
- More sophisticated behavior - libraries
- Designing classes
- Teamwork

Overview

- Data-oriented classes
- A brief history of Java
- Handling errors
- Designing applications

Classes and objects

- Fundamental to much of the early parts of this course.
- Class: category or type of ‘thing’. Like a template or blueprint.
- Object: belongs to a particular class and has individual characteristics.

- Explore through BlueJ …

Fundamental concepts

- object
- class
- method
- parameter
- data type

It is vital to understand these concepts as soon as possible.

Classes and Objects

- A class
  - represents all similar objects of a kind (example: “car”)
- objects
  - represent ‘things’ from the real world, or from some problem domain;
  - example: “that red car in the parking lot”.

Methods and Parameters

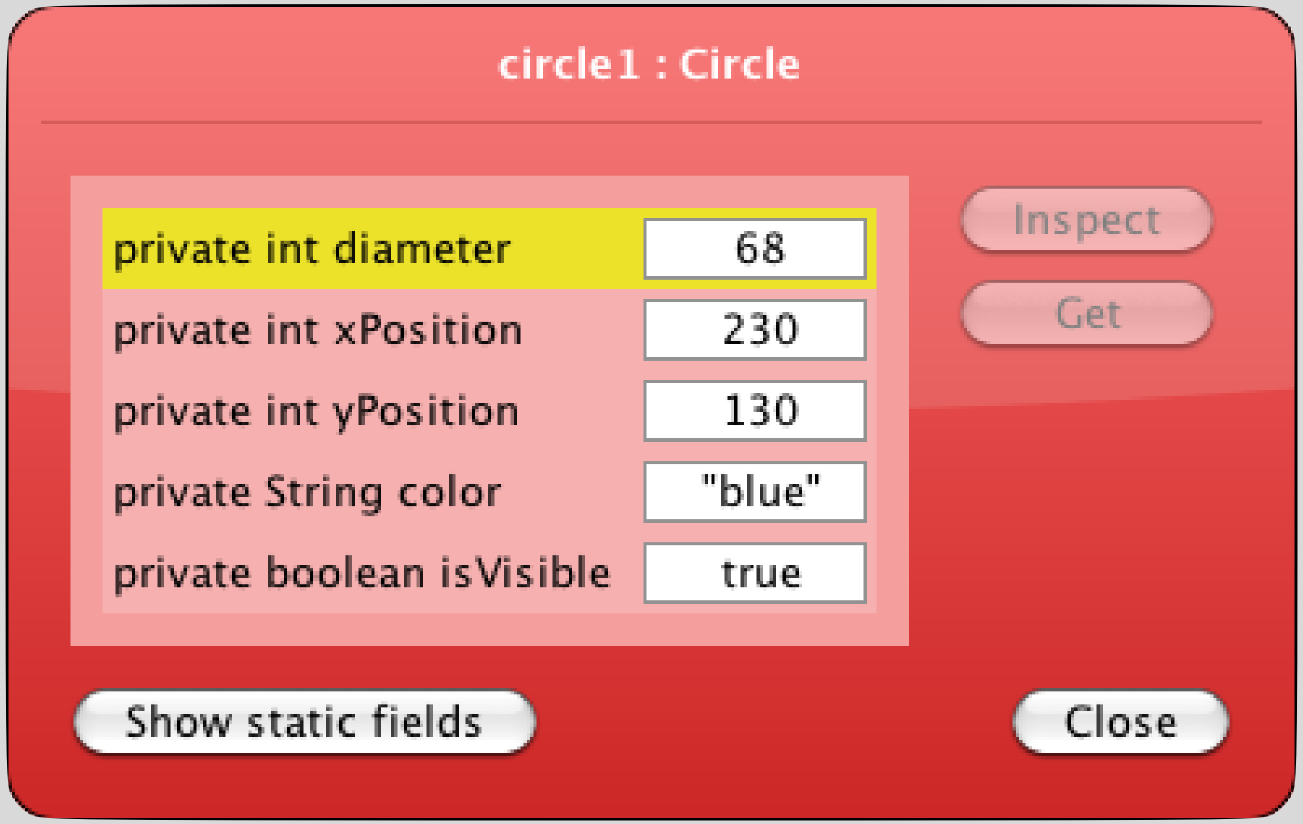
- Objects have operations which can be invoked (Java calls them *methods*).

- Methods may have parameters to receive additional information needed to execute.
  - Parameters introduce variation into the effect of method calls.

Other observations

- Many distinct ***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).

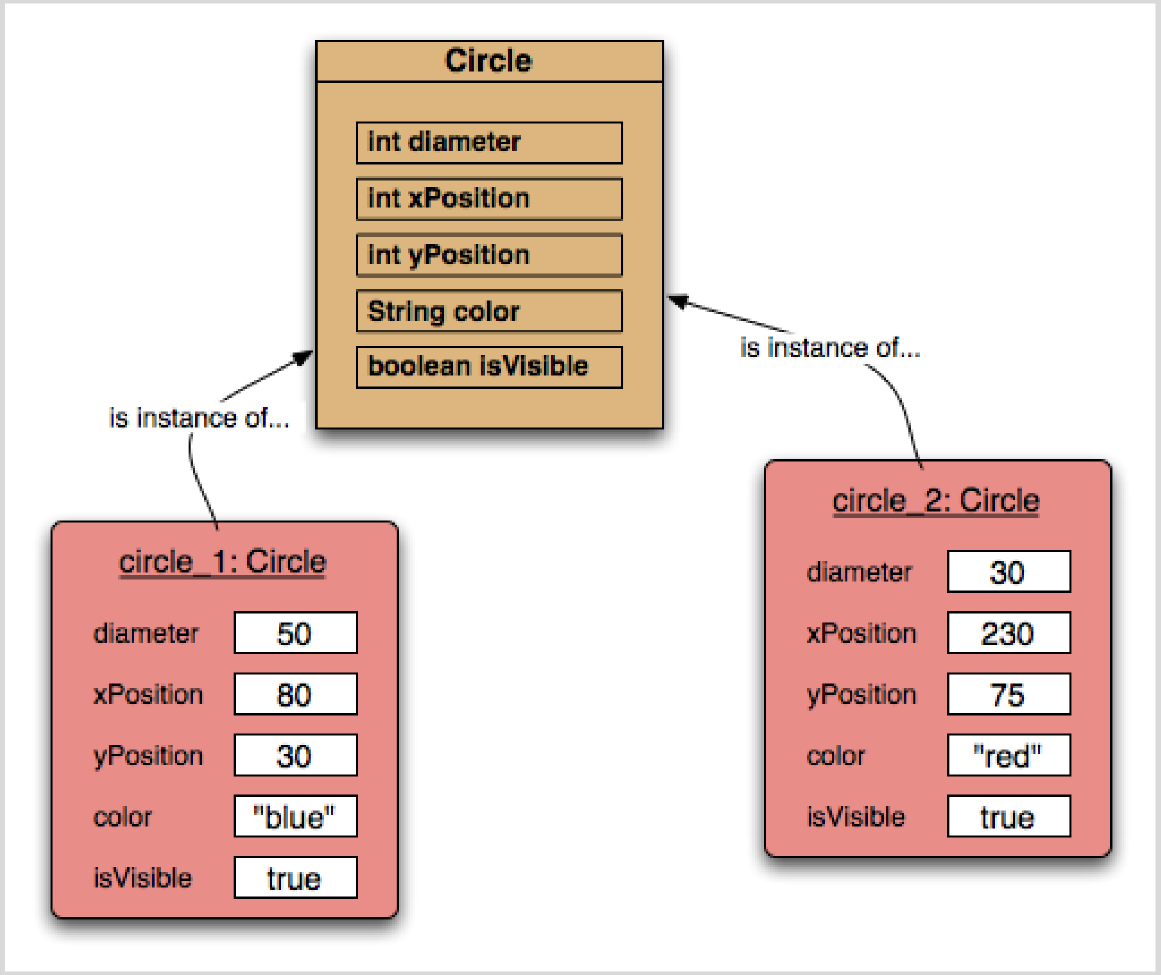
State



circle1 : Circle

private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

Two Circle objects



Source code

- Each class has source code associated with it that defines its details (attributes and methods).
- The source code is written to obey the rules of a particular programming language.

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 later..

Review

- Classes model concepts.
- Source code realises those concepts.
- Source code defines
    - What objects can do (methods).
    - What data they store (attributes).
- Objects come into existence with pre-defined attribute values.
- The methods determine what objects do with their data.

Review

- When a method is called an object:
  - Alters its state, and/or
  - Uses its data to decide what to do.
- Some methods take parameters that affect their actions.
- Methods without parameters typically use their state to decide what to do.
- Some methods return a value.

Review

- Most programs contain multiple classes.
- At runtime, objects interact with each other to realize the overall purpose of the program.

Questions?