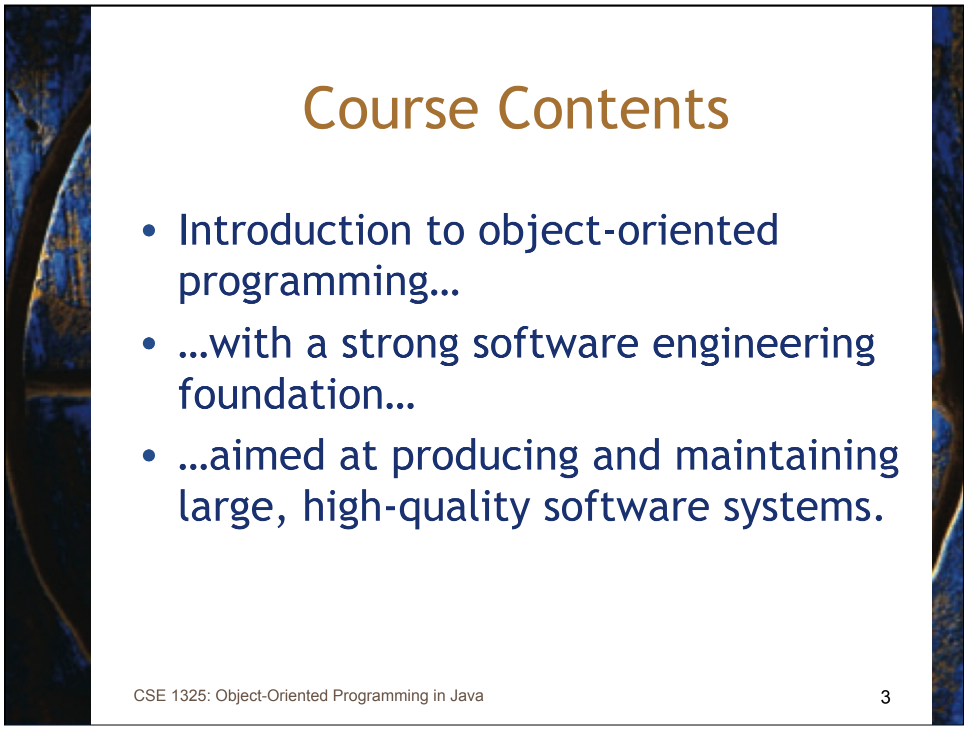
Object-Oriented Programming in Java

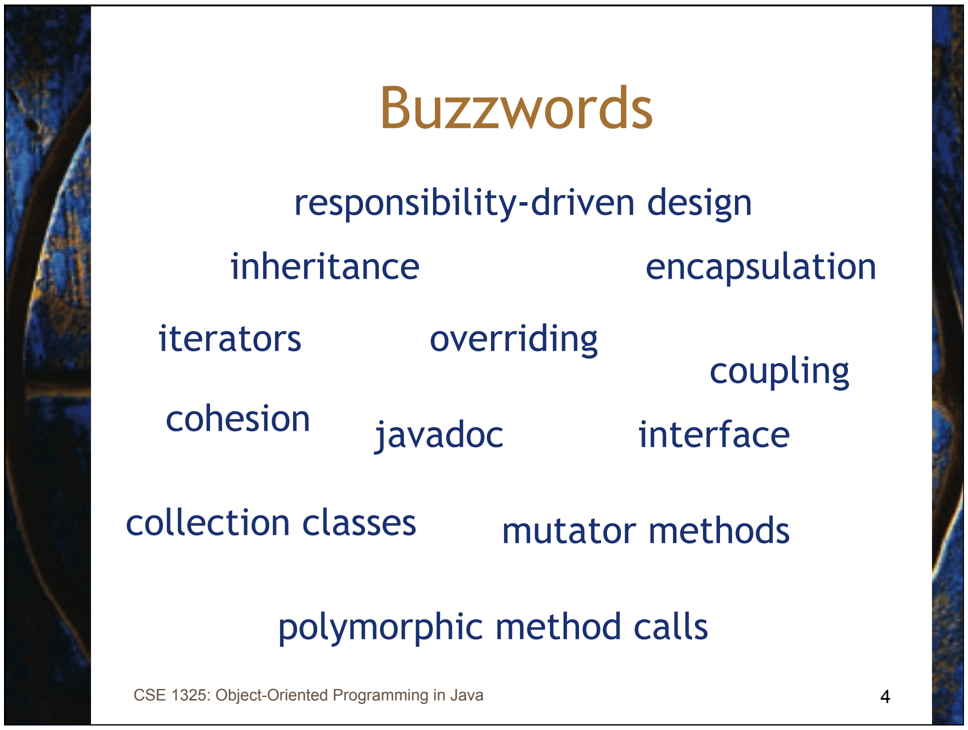
Take control of your own learning

- Lectures
- Textbook
- Exercises
- Internet
- 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

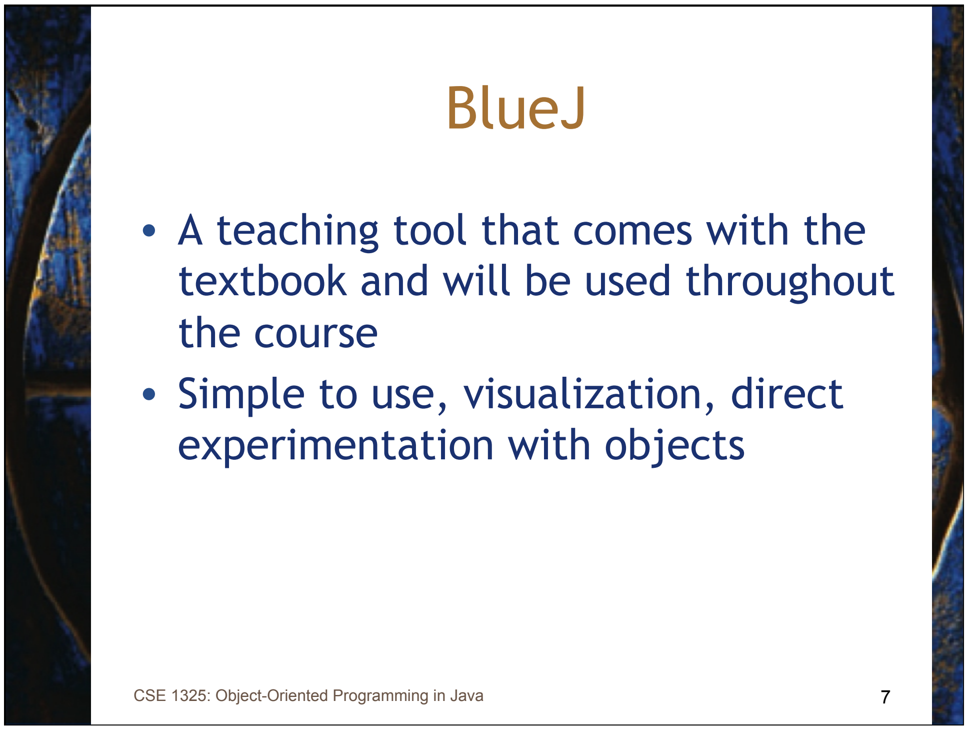
polymorphic method calls

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

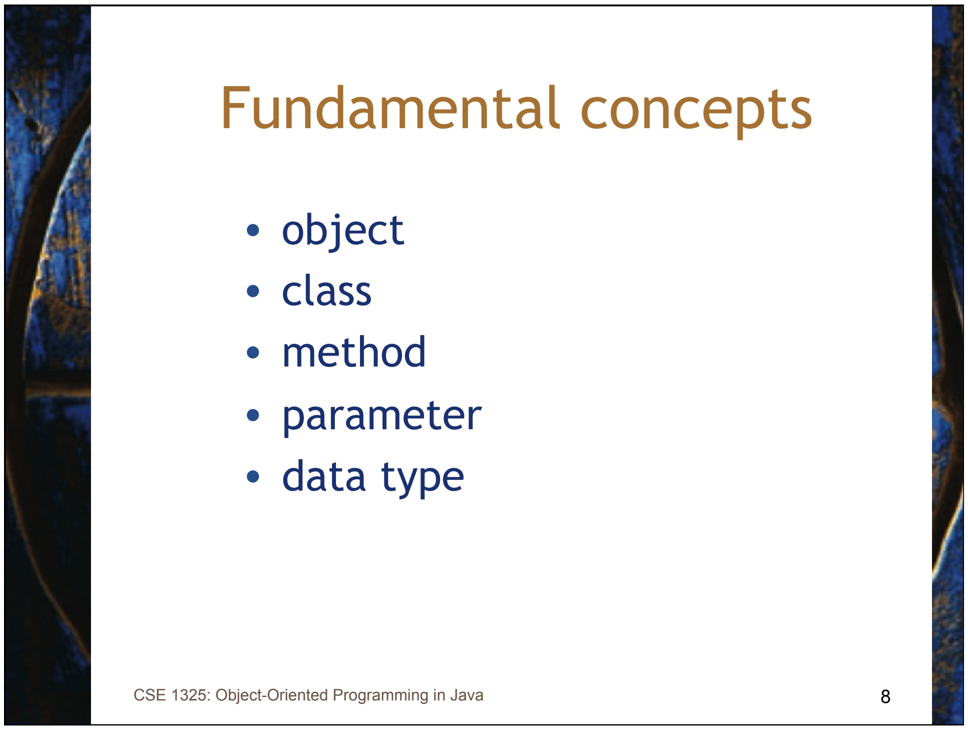
What to Expect

- Expect: fundamental concepts, principles, and techniques of object-oriented programming
- Not to expect: language details, all Java APIs, JSP/JavaServlet, use of specific tools



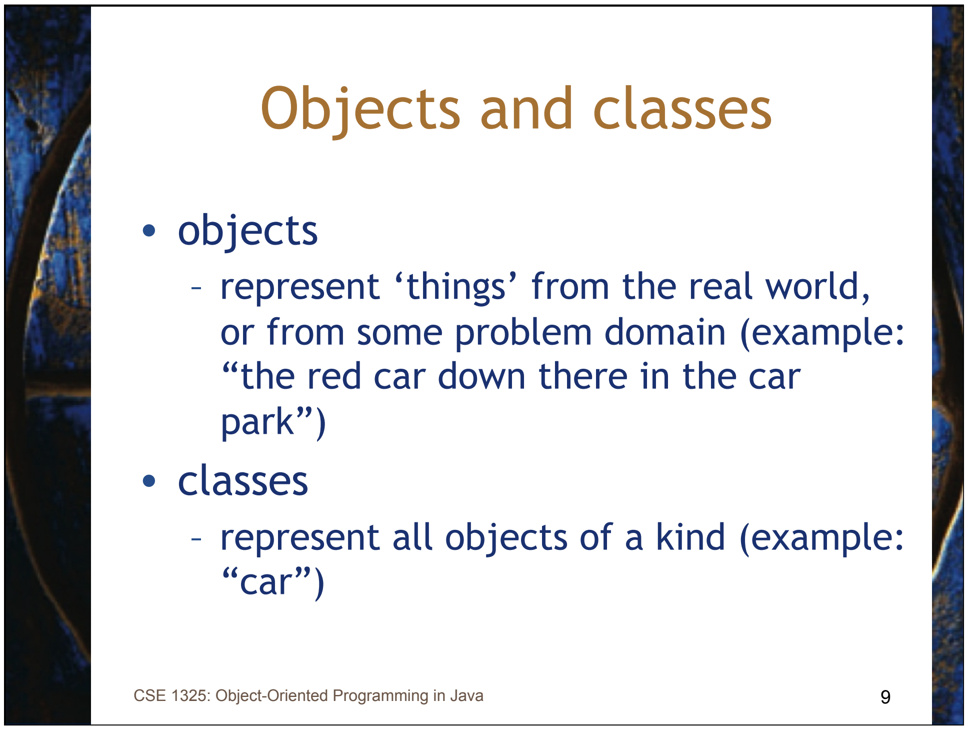
BlueJ

- A teaching tool that comes with the textbook and will be used throughout the course
- Simple to use, visualization, direct experimentation with objects



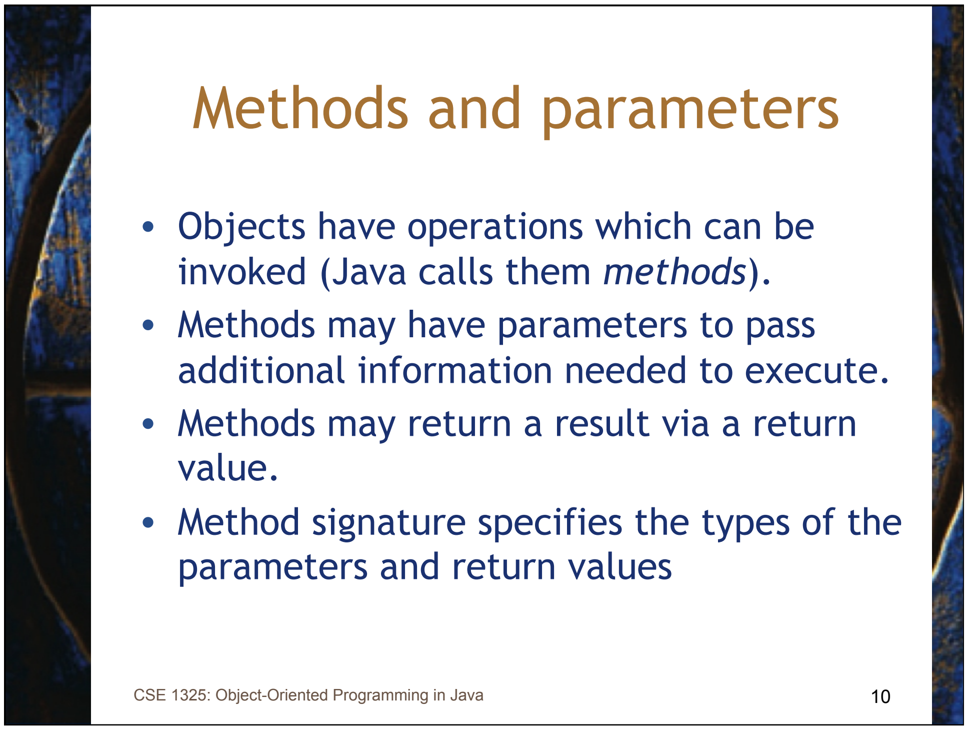
Fundamental concepts

- 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.
- Methods may return a result via a return value.
- Method signature specifies the types of the parameters and return values

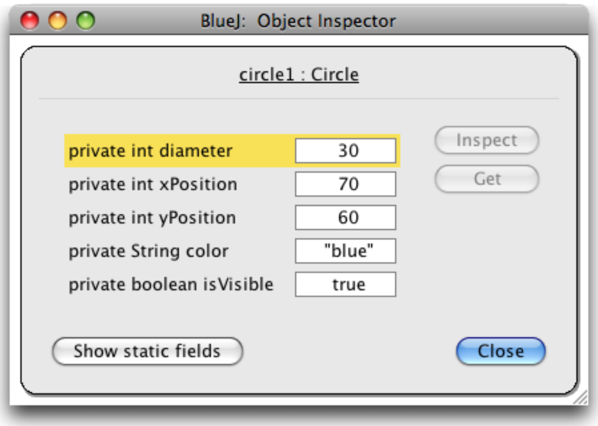
Data Types

- Parameters have data types, which specify what kind of data should be passed to a parameter.
- There are two general kinds of types: *primitive* types, where values are stored in variables directly, and *object* types, where references to objects are stored in variables.

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).

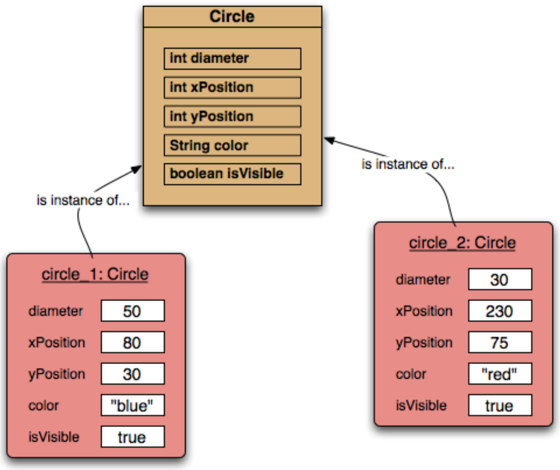
State

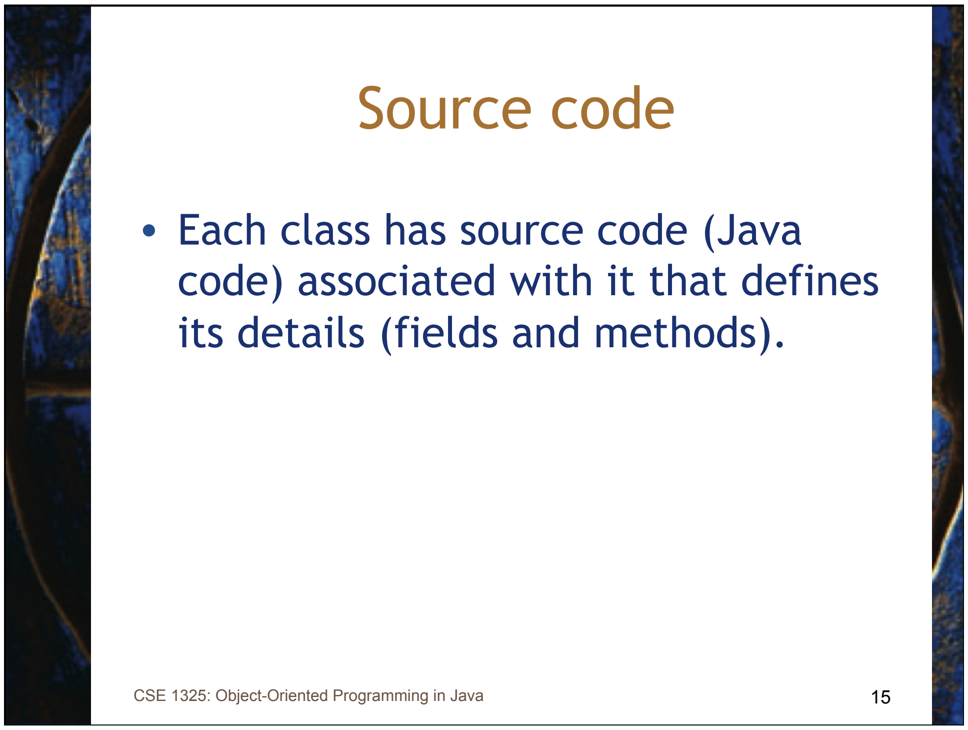

BlueJ: Object Inspector

circle1 : Circle

private int diameter	30
private int xPosition	70
private int yPosition	60
private String color	"blue"
private boolean isVisible	true

Two circle objects





Source code

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