Java Programming AP Edition
U3C9 Objects and Classes

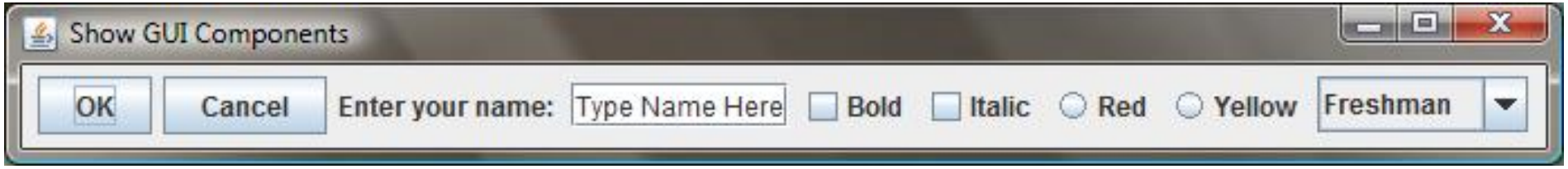
CLASS DEFINITION AND OBJECT CREATION

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

Introduction to Object-Oriented Programming

After learning the preceding chapters, you are capable of solving many programming problems using selections, loops, methods, and arrays. However, these Java features are not sufficient for developing graphical user interfaces and large scale software systems. Suppose you want to develop a graphical user interface as shown below. How do you program it?

Without Object-Oriented Programming, things shall still work. Why we need Object-Oriented Programming?



Motivation for Object-Oriented Programming

- More Compatible for Event-Driven Programming
- More Manageable for GUI Components
- More Organized Data and Methods related to a Certain Objects

Replacing:

(1) Library

(2) Data Records

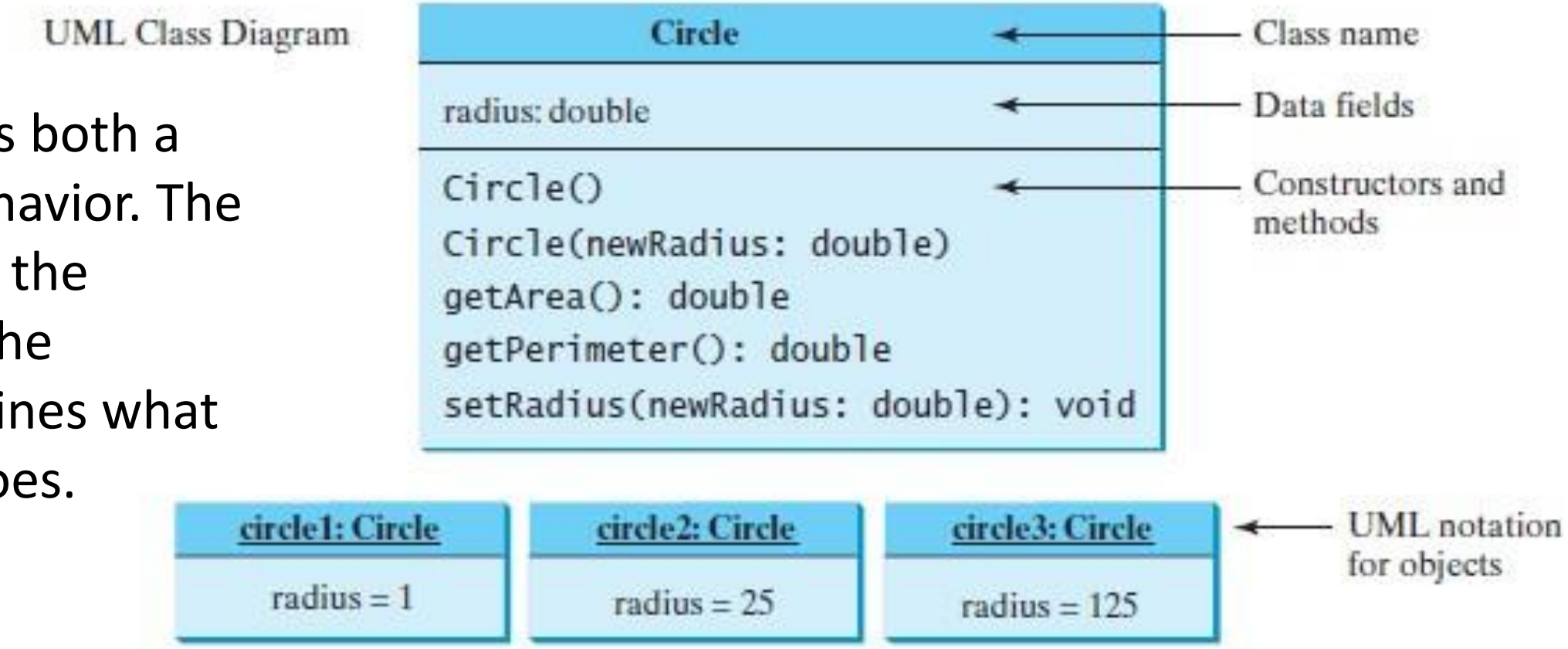
(3) Event-Loop (Execution Flow)

(4) Thread Execution Control

OO Programming Concepts

Object-oriented programming (OOP) involves programming using objects. An ***object*** represents an entity in the real world that can be distinctly identified. For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects. An object has a unique identity, state, and behaviors. The *state* of an object consists of a set of ***data fields*** (also known as ***properties***) with their current values. The *behavior* of an object is defined by a set of methods.

Objects: UML Class/Object Diagram



An object has both a state and behavior. The state defines the object, and the behavior defines what the object does.

Classes

Classes are constructs that define objects of the same type. A Java class uses variables to define data fields and methods to define behaviors. Additionally, a class provides a special type of methods, known as constructors, which are invoked to construct objects from the class.

Classes

```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Data field

Constructors

Method

Example: Defining Classes and Creating Objects

TestSimpleCircle.java

Objective: Demonstrate creating objects, accessing data, and using methods.