Chapter 5: Ball Worlds

Features 2 classes, constant data fields, constructors, extension through inheritance, graphics

Constant data fields

- public static final int FWidth = 600;
  - Easy to edit/change source (single place)
  - Documentation
- Must be initialized! Usually public
- Behaves like a constant, as it cannot be modified while the program runs.

Constructing new objects

- BallWorld w = new BallWorld(Color.red)
- Operator "new" creates a new object, I.e instance of some class, takes a class name and some arguments.
- Ties together creation and initialization
- Prevents against common errors:
  - Object used before initialization
  - Object initialized several times

Constructors

- Like method, but no return type!
- Must have same name as class!
- Never called directly, only by "new"
- Does all the necessary initialization
- If we define none, a default constructor of no arguments is generated
- Can have several constructor with different parameter lists ("overloading")

Constructing applications

- Some prefer separation:

```
public class BallWorldProgram {
 public static void main(String[] args) {
     BallWorld world = new BallWorld(Color.red);
     world.show();
    }
 }
```

- 2 separate files, cumbersome, so we will have a "main" inside the application class

Inheritance

public class BallWorld extends Frame {

- means class BallWorld is a type of Frame, but will add additional behavior
- We don't have to reimplement general behavior defined by class Frame, like resizing, iconification, setTitle(…), setSize(…), show(), …

Graphics model: AWT

- Abstract windowing toolkit
- Example of a software "framework"
  - paint(Graphics g) method
    defines how to draw Frame & Ball
    (abused for simple animation here, too)
  - repaint(1)
    repaint after 1 millisecond
    repaint will call paint,
    but also clear stuff first

System.exit(0)

- Finishes execution, including possibly some cleanup
- Passes a return code to the operating system:
  - 0 is used to indicate normal completion
  - other integers are taken to be error codes

The class Ball

- Protected data members: allows subclasses to access these directly; (good idea?)
- Socalled "accessor" methods, for reading and writing data members (setLocation, radius, …)
- Use Rectangle class, and methods of that class as well as of class Graphics (setLocation, translate, setColor, fillOval, …)

Multiple objects of same class

- e.g. have 2 windows, or an array of balls:
  ballArray = new Ball [BallArraySize];
- Must be initialized appropriately:

```
for (int I = 0; I < BallArraySize; I++) {
    ballArray[I] = new Ball(10,15,5);
    ballArray[I].setColor(ballColor);
    ballArray[I].setMotion(3.0+I,6.0-I); }
```

- All have their own separate data members!

Summary 1

- final fields cannot be modified;
- static final denote symbolic constants
- new generates new objects
- new invokes a constructor
- Constructor must have same name as its class, must initialize properly!
- new must supply all arguments

Summary 2

- Classes can use "inheritance"
- Use class Frame for simple windows
- AWT provides a windowing framework
- Use paint method to create images
- Class Rectangle: useful library class
- Multiple instances of same class have their own separate data fields.