CSSE220 Day 9

Communities of interacting objects; UML
BallWorlds Intro
Work on BallWorlds

Announcements

- Swing Warmup due now
- But I will allow a grace period until tonight, 11:59 PM. (There are other things to do, so you’re a bit behind, but it won’t be late if you submit it by then.)
- Today:
  - Interacting communities of Objects
Intro to BallWorlds (due Friday, 3/28)
- Tuesday: Finish GUIs

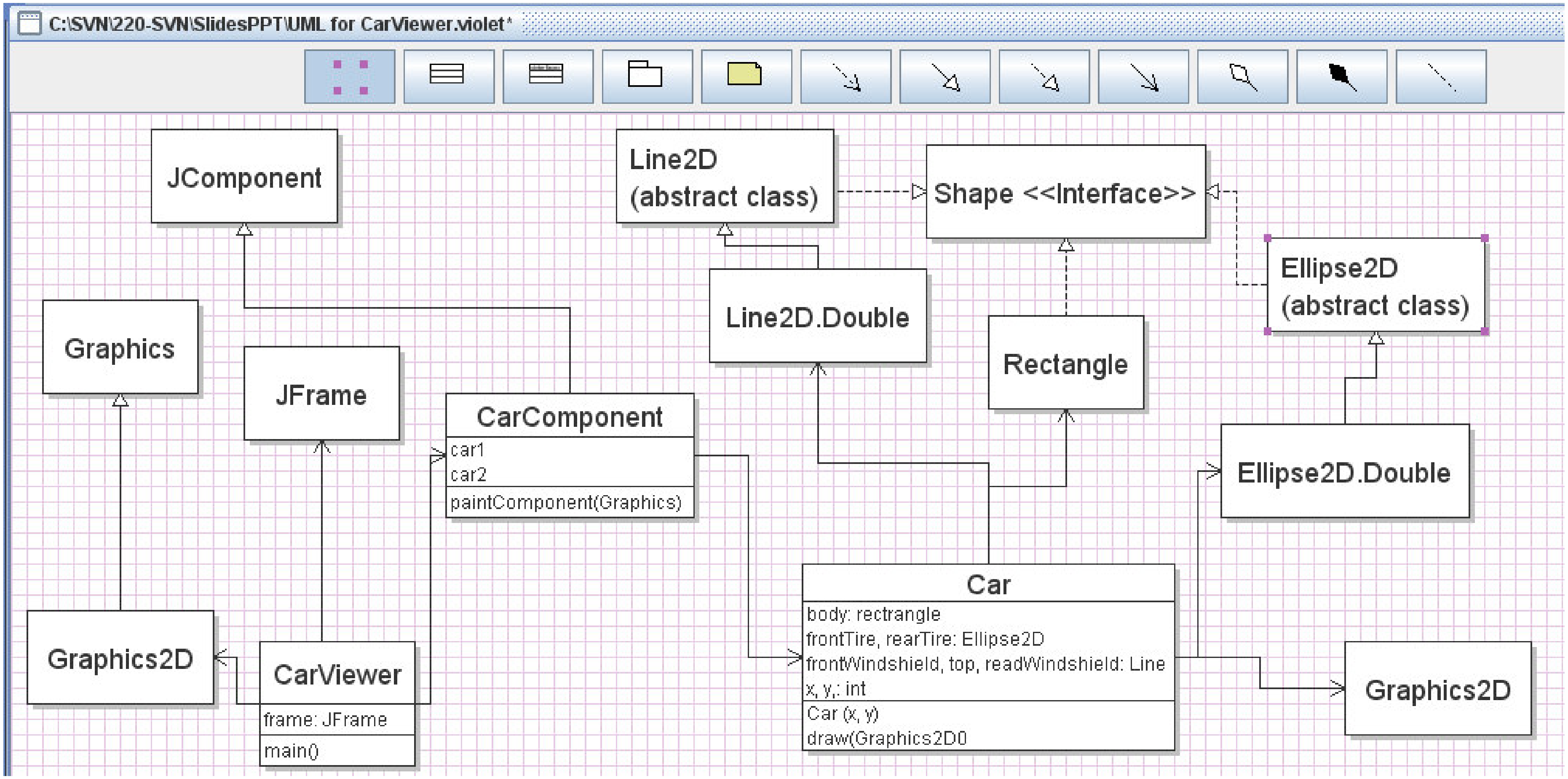
Event Handling Recap

- For a given event type X, a GUI component c, and an XListener object xLis,
  - the call `c.addXListener(xLis);` says to the c object,
  - "Whenever an event of type X happens, notify object xLis by calling its appropriate 'X handler' method."

Interaction in UML Class Diagrams

- So far, each of the programs we have written has involved at most three new classes that we wrote, plus a handful of classes from the Java library.
- Many "real" programs involve dozens or hundreds of classes, with complex interactions among objects from those classes.
- For large programs can't just start writing code and hope it works out!
- UML Class Diagrams can help us to visualize the classes and their interactions before we write the code.

A UML diagram for our Cars Program


C:\SVN\220-SVN\SlidesPPT\UML for CarViewer.violet*

BallWorlds Intro

- So far, we have written "from scratch" programs.

- Most programmers do not get that luxury.
  - They write a small part of a program that is designed/written by a larger team.
  - Their part has to "fit" with the other parts.
  - They have to understand enough of the other parts to be able to make their part work.

- In BallWorlds, you will experience that.

Goals of BallWorlds

- Understand things about a program based on its UML Class Diagram
- Figure out which parts are relevant to what you have to do
- Experience the power of inheritance

- DEMO:
  - Demonstrate the program
  - How many worlds are there?

Creation of the Worlds

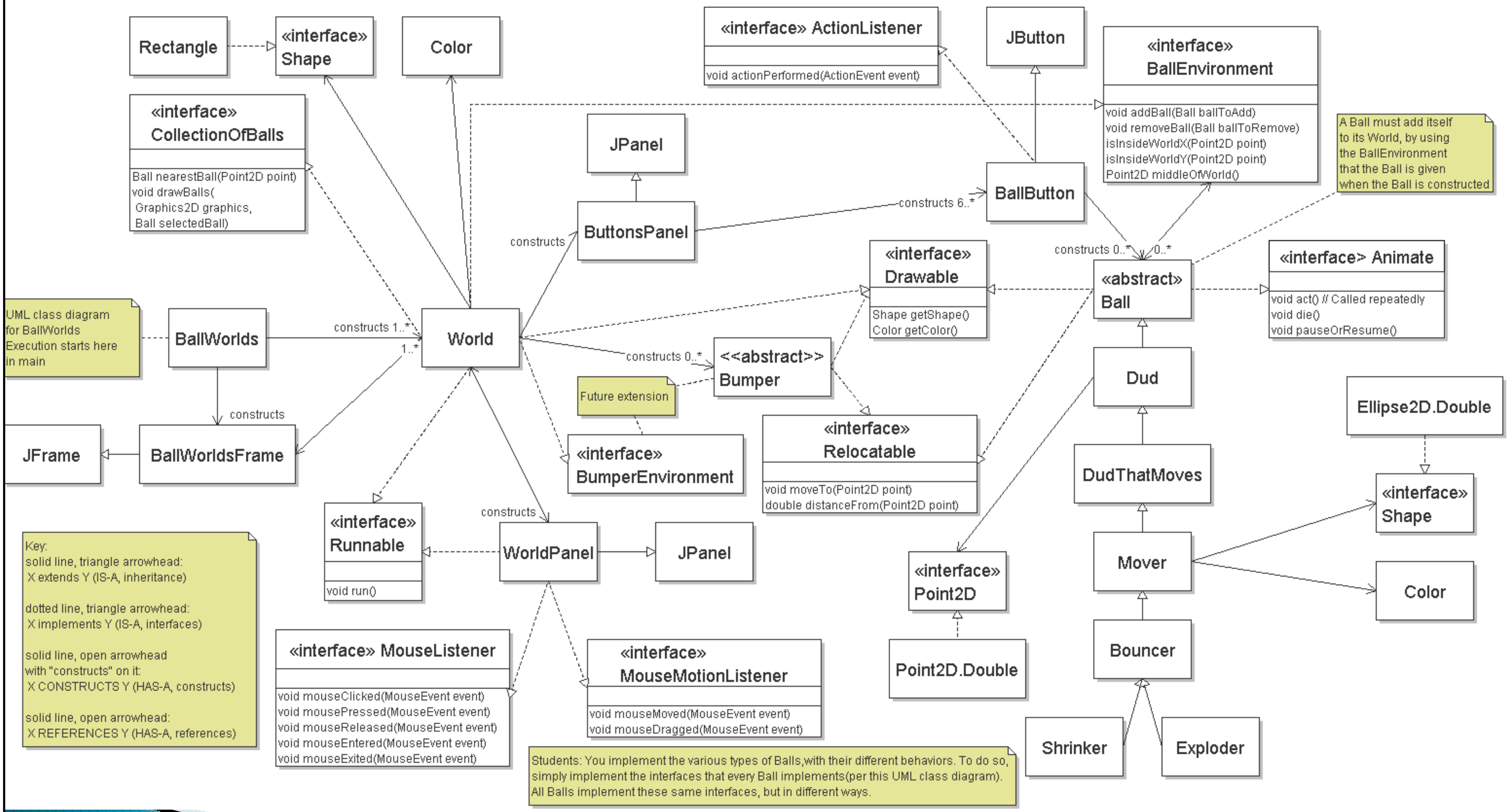
```
/*
 * Makes the given number of Worlds, giving each the given frame.
 * Rotates between 3 pre-assigned sizes and colors for the Worlds.
 */
private static void makeWorlds(int numberOfWorlds,
                               BallWorldsFrame frame) {
    ArrayList<Dimension> dimensions = new ArrayList<Dimension>();
    ArrayList<Color> colors = new ArrayList<Color>();

    dimensions.add(BallWorlds.world1Size);
    dimensions.add(BallWorlds.world2Size);
    dimensions.add(BallWorlds.world3Size);

    colors.add(BallWorlds.world1Color);
    colors.add(BallWorlds.world2Color);
    colors.add(BallWorlds.world3Color);

    for (int k = 0; k < numberOfWorlds; ++k) {
       new World(dimensions.get(k % 3), colors.get(k % 3), frame);
    }
}
```

UML Class Diagram for BallWorlds


Rectangle
«interface» Shape
Color
«interface» CollectionOfBalls
Ball nearestBall(Point2D point)
void drawBalls(
Graphics2D graphics,
Ball selectedBall)
«interface» ActionListener
void actionPerformed(ActionEvent event)
JButton
«interface» BallEnvironment
void addBall(Ball ballToAdd)
void removeBall(Ball ballToRemove)
isInsideWorldX(Point2D point)
isInsideWorldY(Point2D point)
Point2D middleOfWorld()
JPanel
ButtonsPanel
constructs
BallButton
constructs 6..*
constructs 0..*
0..*
«interface» Drawable
Shape getShape()
Color getColor()
«abstract» Ball
«interface» Animate
void act() // Called repeatedly
void die()
void pauseOrResume()
BallWorlds
constructs 1..*
1..*
World
constructs 0..*
<<abstract>> Bumper
Dud
constructs
JFrame
BallWorldsFrame
«interface» BumperEnvironment
«interface» Relocatable
void moveTo(Point2D point)
double distanceFrom(Point2D point)
DudThatMoves
Ellipse2D.Double
«interface» Shape
«interface» Runnable
void run()
constructs
WorldPanel
JPanel
«interface» Point2D
Mover
Color
Point2D.Double
Bouncer
«interface» MouseListener
void mouseClicked(MouseEvent event)
void mousePressed(MouseEvent event)
void mouseReleased(MouseEvent event)
void mouseEntered(MouseEvent event)
void mouseExited(MouseEvent event)
«interface» MouseMotionListener
void mouseMoved(MouseEvent event)
void mouseDragged(MouseEvent event)
Shrinker
Exploder

UML class diagram for BallWorlds
Execution starts here in main

A Ball must add itself to its World, by using the BallEnvironment that the Ball is given when the Ball is constructed

Future extension

Key:
solid line, triangle arrowhead:
X extends Y (IS-A, inheritance)

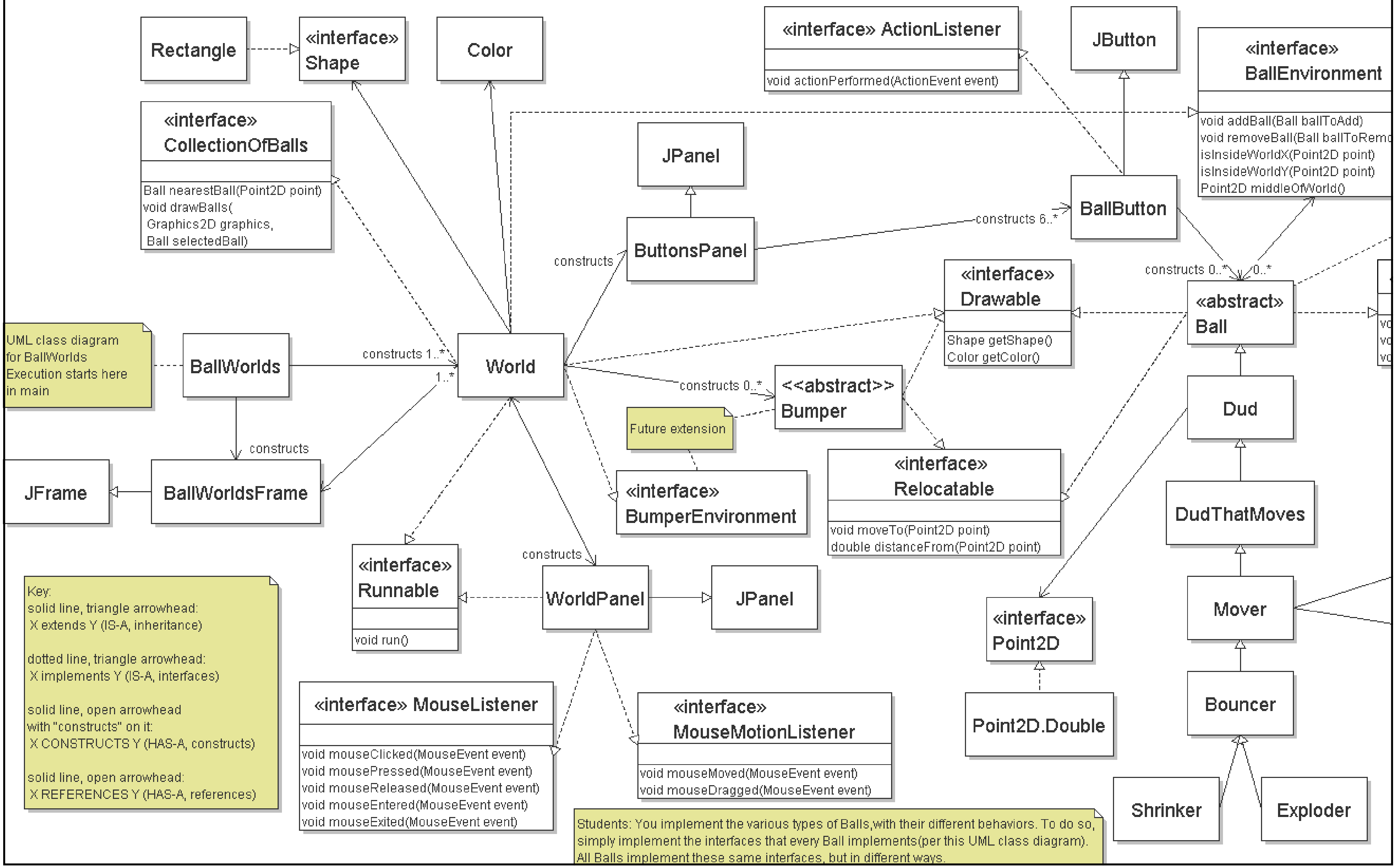
dotted line, triangle arrowhead:
X implements Y (IS-A, interfaces)

solid line, open arrowhead
with "constructs" on it:
X CONSTRUCTS Y (HAS-A, constructs)

solid line, open arrowhead:
X REFERENCES Y (HAS-A, references)

Students: You implement the various types of Balls,with their different behaviors. To do so, simply implement the interfaces that every Ball implements(per this UML class diagram). All Balls implement these same interfaces, but in different ways.

More details on Part of Diagram


Rectangle
«interface» Shape
Color
«interface» ActionListener
void actionPerformed(ActionEvent event)
JButton
«interface» BallEnvironment
void addBall(Ball ballToAdd)
void removeBall(Ball ballToRem
isInsideWorldX(Point2D point)
isInsideWorldY(Point2D point)
Point2D middleOfWorld()
«interface» CollectionOfBalls
Ball nearestBall(Point2D point)
void drawBalls(
Graphics2D graphics,
Ball selectedBall)
JPanel
ButtonsPanel
constructs
constructs 6..*
BallButton
constructs 0..*
0..*
«interface» Drawable
Shape getShape()
Color getColor()
«abstract» Ball
BallWorlds
constructs 1..*
1..*
World
constructs 0..*
<<abstract>> Bumper
Future extension
Dud
constructs
JFrame
BallWorldsFrame
«interface» BumperEnvironment
«interface» Relocatable
void moveTo(Point2D point)
double distanceFrom(Point2D point)
DudThatMoves
constructs
«interface» Runnable
void run()
WorldPanel
JPanel
«interface» Point2D
Mover
Point2D.Double
Bouncer
«interface» MouseListener
void mouseClicked(MouseEvent event)
void mousePressed(MouseEvent event)
void mouseReleased(MouseEvent event)
void mouseEntered(MouseEvent event)
void mouseExited(MouseEvent event)
«interface» MouseMotionListener
void mouseMoved(MouseEvent event)
void mouseDragged(MouseEvent event)
Shrinker
Exploder
vo
vo
vo

UML class diagram
for BallWorlds
Execution starts here
in main

Key:
solid line, triangle arrowhead:
X extends Y (IS-A, inheritance)

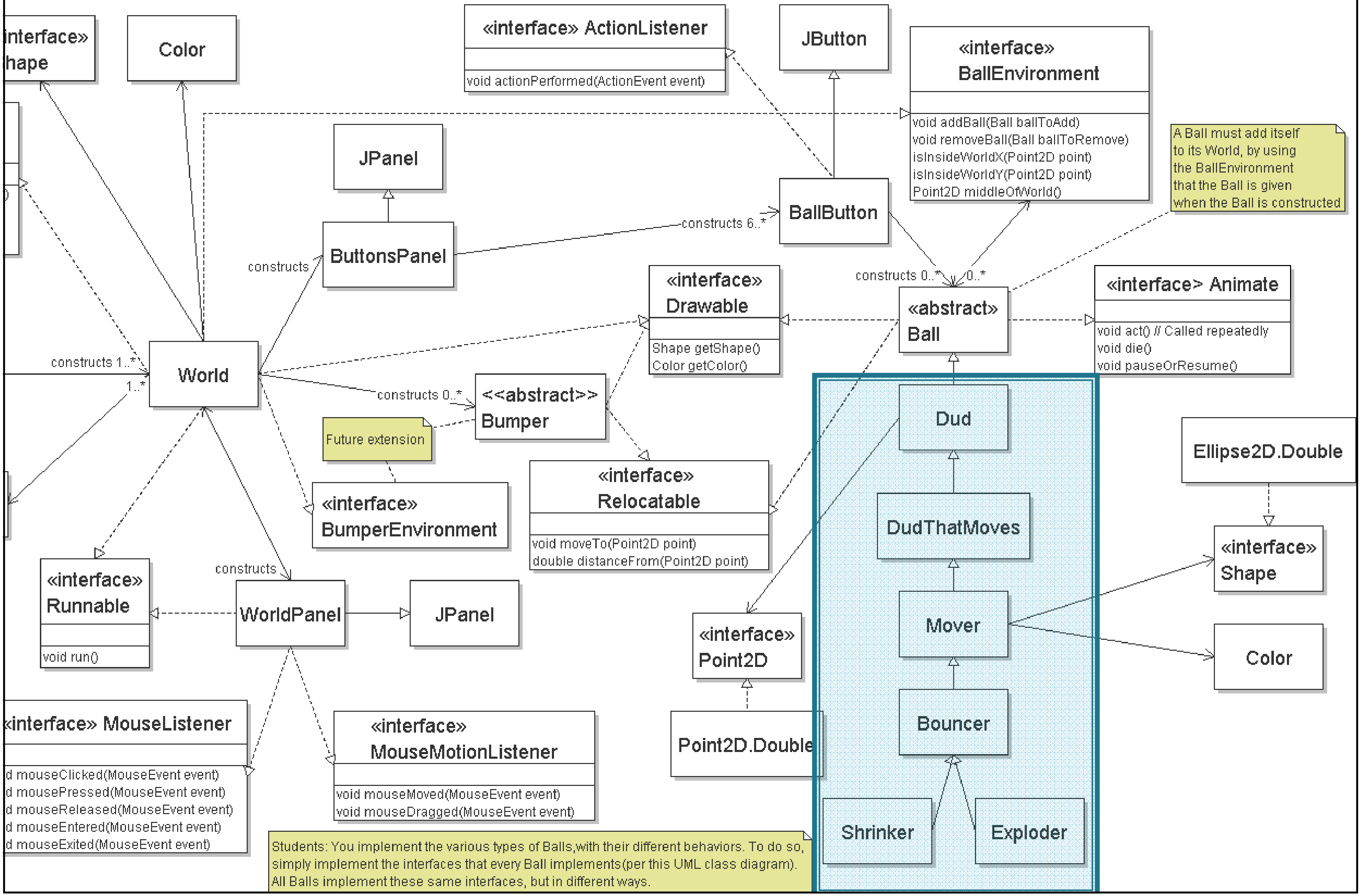
dotted line, triangle arrowhead:
X implements Y (IS-A, interfaces)

solid line, open arrowhead
with "constructs" on it:
X CONSTRUCTS Y (HAS-A, constructs)

solid line, open arrowhead:
X REFERENCES Y (HAS-A, references)

Students: You implement the various types of Balls,with their different behaviors. To do so, simply implement the interfaces that every Ball implements(per this UML class diagram). All Balls implement these same interfaces, but in different ways.

Focus on the Part You Will Implement



interface» hape

Color

«interface» ActionListener

void actionPerformed(ActionEvent event)

JButton

«interface» BallEnvironment

void addBall(Ball ballToAdd)
void removeBall(Ball ballToRemove)
isInsideWorldX(Point2D point)
isInsideWorldY(Point2D point)
Point2D middleOfWorld()

A Ball must add itself to its World, by using the BallEnvironment that the Ball is given when the Ball is constructed

JPanel

ButtonsPanel

BallButton

constructs 6..*

constructs

constructs 0..*

0..*

«interface» Drawable

Shape getShape()
Color getColor()

«abstract» Ball

«interface> Animate

void act() // Called repeatedly
void die()
void pauseOrResume()

constructs 1..*

1..*

World

constructs 0..*

<<abstract>> Bumper

Future extension

«interface» BumperEnvironment

«interface» Relocatable

void moveTo(Point2D point)
double distanceFrom(Point2D point)

Dud

DudThatMoves

Mover

Bouncer

Shrinker

Exploder

Ellipse2D.Double

«interface» Shape

Color

«interface» Point2D

Point2D.Double

constructs

«interface» Runnable

void run()

WorldPanel

JPanel

«interface» MouseListener

d mouseClicked(MouseEvent event)
d mousePressed(MouseEvent event)
d mouseReleased(MouseEvent event)
d mouseEntered(MouseEvent event)
d mouseExited(MouseEvent event)

«interface» MouseMotionListener

void mouseMoved(MouseEvent event)
void mouseDragged(MouseEvent event)

Students: You implement the various types of Balls,with their different behaviors. To do so, simply implement the interfaces that every Ball implements(per this UML class diagram). All Balls implement these same interfaces, but in different ways.

Ball Class

- Abstract
- Implements which interfaces?
- What data might be needed for every kind of Ball?

- Let's do a little bit of code exploration.
- Then write Dud together.