# // D for DATA

## interface EventHandler {

public void handleEvent(Event e);

}

## class MyPanel extends Panel

{

### public MyPanel() {

Panel();

eventHandler\_ = null;

frame\_ = new Frame("Bouncy");

frame\_.add("Center", this);

frame\_.resize(XSIZE, YSIZE);

frame\_.setVisible(true);

repaint()

}

int XSIZE = 1000;

int YSIZE = 600;

### public Frame frame() { return frame\_ }

private Frame frame\_;

private EventHandler eventHandler\_

public int xsize() { return XSIZE }

public int ysize() { return YSIZE }

### public void setEventHandler(EventHandler eh) {

eventHandler\_ = eh

}

### public boolean handleEvent(Event event) {

if (event.id == Event.MOUSE\_MOVE) {

if (eventHandler\_ != null) {

eventHandler\_.handleEvent(event)

}

}

return true

}

}

### 

## class Point {

int x\_, y\_;

### public Point(int x, int y) { x\_ = x.clone; y\_ = y.clone }

### public void setXY(int x, int y) { x\_ = x.clone; y\_ = y.clone }

### public int x() { return x\_ }

### public int y() { return y\_ }

}

## class BallObject extends Point {

public BallObject(int x, int y) {

Point(x, y);

velocity\_ = new Point(7, 7)

### }

Point velocity\_;

private int RADIUS = 15

### public int radius() { return RADIUS }

### public Point velocity() { return velocity\_ }

### public void setVelocity(Point velocity) { velocity\_ = velocity }

}

# // C for CONTEXT

## context Arena implements EventHandler {

### public Arena() {

MyPanel panel = new MyPanel();

THEPANEL = panel;

BALL = new BallObject(50, 50);

PADDLE= new Point(450, 560);

panel.setEventHandler(this)

}

### private boolean handleEvent(Event e) {

if (e.id == Event.MOUSE\_MOVE) {

Point newLoc = new Point(e.x, e.y);

PADDLE.moveTo(newLoc)

}

return true

}

### public void run() {

do {

THEPANEL.clear();

BALL.draw();

PADDLE.draw();

BALL.velocityAdjust();

BALL.step();

THEPANEL.refresh();

Thread.sleep(20)

} while (true)

}

# // I for INTERACTON

## role THEPANEL {

### public void drawCircle(int x, int y, int r) {

setForeground(Color.blue);

fillOval(x+r, y+r, r, r)

}

### public void drawPaddle(int xs, int ys, int h, int w) {

setForeground(new Color(32, 170, 64));

drawRect(xs, ys, h, w)

}

### public int maxX() { return xsize() }

### public int maxY() { return ysize() }

### public void refresh() { repaint() }

### public void clear() {

removeAll();

setForeground(Color.white);

drawRect(0, 0, xsize(), ysize());

setForeground(Color.white)

}

} requires {

void fillOval(int x, int y, int h, int w);

void drawRect(int x, int y, int h, int w);

int xsize();

int ysize();

void repaint();

void removeAll();

void setForeground(Color color)

}

## role PADDLE{

public int thickness() const { return 10 }

public int width() const { return 100 }

### public void draw() {

THEPANEL.drawPaddle(x() - (width() / 2), y(), width(), thickness())

}

### public void moveTo(Point p) {

setXY(p.x(), y())

}

### public boolean contains(int x) const {

return ((x > x() - (width() / 2)) &&

(x < x() + (width() / 2)))

}

### public int vertical() const {

return y() - thickness() }

} requires {

void setXY(int x, int y);

int x() const;

int y() const

}

## role BALL {

### public void draw() {

THEPANEL.drawCircle(x(), y(), radius())

}

### public void step() {

setXY(x() + velocity().x, y() + velocity().y);

}

### private boolean bouncingOffOfPaddle() {

if (y() + (radius() \* 2) > PADDLE.vertical()) {

return PADDLE.contains(x())

}

return false

}

### public void velocityAdjust() {

int maxX = THEPANEL.maxX(), maxY = THEPANEL.maxY();

int xv = velocity().x, yv = velocity().y;

int newXv = xv.clone, newYv = yv.clone;

if (xv > 0) { *//moving right*

if (x() + (radius() \* 2) >= maxX) newXv = -xv //bouncing right wall

} else *// moving left*

if (x() + radius() <= 0) newXv = -xv; //bouncing left wall

if (yv > 0) { *// Moving down*

if (bouncingOffOfPaddle()) newYv = -yv // normal bounce

// else continues moving down out of sight. Bounce at some y?

} else { *// moving up*

if (y() + (radius() \* 2) <= 0) newYv = -yv // bouncing off top

}

Point retval = new Point(newXv, newYv);

setVelocity(retval)

}

} requires {

void setXY(int x, int y);

int x();

int y();

Point velocity();

void setVelocity(Point velocity);

int radius()

}

}

# // Starting the program

new Arena().run()