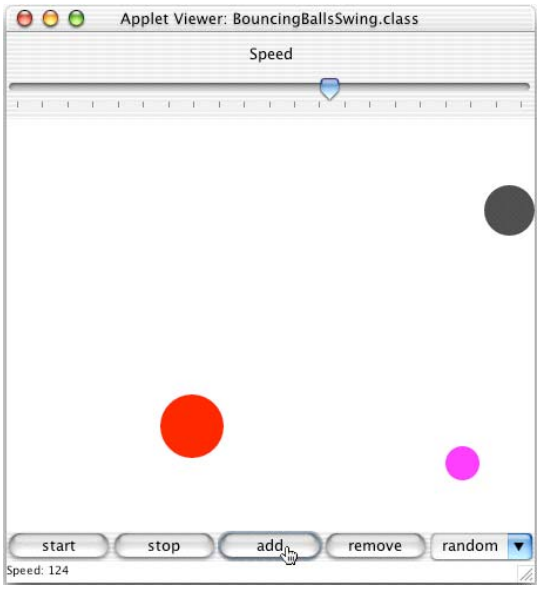
Opgave 8.2



Appletten er tilført følgende ny funktionalitet:

Knappen “add” tilføjer en ny bold på et tilfældigt sted i tegneområdet. Radius og hastighed vælges tilfældigt (inden for givne grænser).

Knappen ”remove” fjerner den sidst tilføjede bold.

Farven på bolden kan vælges ved hjælp valg-komponenten nederst til højre. Som vist på figuren er det muligt at angive, at farven skal vælges tilfældigt (blandt 12 farver).

En ny bold kan tilføjes på et ønsket sted i tegneområdet ved blot at klikke med musen på det ønskede sted.

Skyderen øverst i rammen kan benyttes til at ændre animationshastigheden.

```
import java.awt.*;
import javax.swing.*;

public class BouncingBallCanvas extends JPanel {
    public void initCanvas() {
        balls = new BallSet();
    }

    public void setBallColor(Color c) { ballColor = c; }

    protected double rand(double min, double max) {
        return min + Math.random() * (max - min);
    }

    public void addBall(double x, double y) {
        double radius = rand(minRadius, maxRadius);
        x = Math.max(radius, Math.min(x, d.width - radius));
        y = Math.max(radius, Math.min(y, d.height - radius));
        Color oldBallColor = ballColor;
        if (ballColor == null)
            ballColor = colors[(int) rand(0, colors.length)];
        Ball ball = new Ball(x, y, radius,
                             rand(minDx, maxDx), rand(minDy, maxDy),
                             ballColor, d);
        ballColor = oldBallColor;
        for (int i = 0; i < balls.size(); i++) {
            Ball b = (Ball) balls.get(i);
            if (ball.collidesWith(b)) {
                ball.dx = - b.dx;
                ball.dy = - b.dy;
            }
        }
        balls.add(ball);
        repaint();
    }

    public void addBall() {
        addBall(rand(0, d.width), rand(0, d.height));
    }

    public void paint(Graphics g) {
        d = getSize(d);
        g.setColor(Color.white);
        g.fillRect(0, 0, d.width, d.height);
        balls.move(d);
        balls.draw(g);
    }

    protected Color ballColor;
    protected double minRadius = 10, maxRadius = 30;
    protected double minDx = -10, maxDx = 10, minDy = -10, maxDy = 10;
    protected Dimension d;
    protected BallSet balls;
```

```
    public final Color[] colors =
        { Color.black, Color.blue, Color.cyan, Color.darkGray,
          Color.gray, Color.green, Color.lightGray, Color.magenta,
          Color.orange, Color.pink, Color.red, Color.yellow };
    public final String[] colorNames =
        { "black", "blue", "cyan", "dark gray",
          "gray", "green", "light gray", "magenta",
          "orange", "pink", "red", "yellow" };
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;
import javax.swing.event.*;

public class BouncingBalls extends JApplet {
    public BouncingBalls() {
        Container pane = getContentPane();
        pane.setLayout(new BorderLayout());
        controlPanel = new JPanel();
        controlPanel.setLayout(new GridLayout(1,0));
        controlPanel.setCursor(Cursor.getPredefinedCursor(
                               Cursor.HAND_CURSOR));
        canvas = new BouncingBallCanvas();
        canvas.addMouseListener(new MouseHandler());
        canvas.setCursor(Cursor.getPredefinedCursor(
                         Cursor.CROSSHAIR_CURSOR));
        pane.add("Center", canvas);
        animator = new Animator(canvas);

        JButton startButton = new JButton("start");
        startButton.addActionListener(new ButtonHandler(
                                        new StartCommand()));
        controlPanel.add(startButton);
        startButton.setToolTipText("Start the animation");

        JButton stopButton = new JButton("stop");
        stopButton.addActionListener(new ButtonHandler(
                                       new StopCommand()));
        controlPanel.add(stopButton);
        stopButton.setToolTipText("Stop the animation");

        JButton addButton = new JButton("add");
        addButton.addActionListener(new ButtonHandler(
                                      new AddCommand()));
        controlPanel.add(addButton);
        addButton.setToolTipText("Add a new ball");

        JComboBox choice = new JComboBox();
        choice.addItemListener(new ColorChoiceHandler());
        choice.addItem("random");
        for (int i = 0; i < canvas.colorNames.length; i++)
           combo.addItem(canvas.colorNames[i]);
        controlPanel.add(choice);
        choice.setToolTipText("Change the color of the next ball");

        pane.add("South", controlPanel);
```

```
    speedPanel = new JPanel();
    speedPanel.setLayout(new GridLayout(1,0));
    pane.add("North", speedPanel);
    speedSlider = new JSlider(
        SwingConstants.HORIZONTAL, 0, maxDelay, 0);
    speedSlider.addChangeListener(new SpeedSliderHandler());
    speedSlider.setMajorTickSpacing(10);
    speedSlider.setPaintTicks(true);
    speedPanel.add(speedSlider);
}

public void init() {
    animator.setDelay(maxDelay / 2);
    speedSlider.setValue(maxDelay / 2);
    showStatus("Speed: " +
               Integer.toString(animator.getDelay()));
    canvas.initCanvas();
}

public void start() {
    animator.start();
}

public void stop() {
    animator.stop();
}

protected BouncingBallCanvas canvas;
protected Animator animator;
protected JPanel controlPanel, speedPanel;
protected JSlider speedSlider;
protected final int maxDelay = 200;

protected class ButtonHandler implements ActionListener {
    private Command cmd;

    public ButtonHandler(Command cmd) {
        this.cmd = cmd;
    }

    public void actionPerformed(ActionEvent event) {
        if (cmd != null)
            cmd.execute();
    }
}
```

```
    protected class StartCommand implements Command {
        public void execute() {
            start();
        }
    }

    protected class StopCommand implements Command {
        public void execute() {
            stop();
        }
    }


    protected class AddCommand implements Command {
        public void execute() {
           canvas.addBall();
        }
    }

    protected class ColorChoiceHandler implements ItemListener {
        public void itemStateChanged(ItemEvent event)  {
            JComboBox choice = (JComboBox) event.getSource();
            if (choice != null) {
                canvas.setBallColor(null);
                for (int i = 0; i < canvas.colorNames.length; i++)
                    if (event.getItem().equals(canvas.colorNames[i]))
                        canvas.setBallColor(canvas.colors[i]);
            }
        }
    }

    protected class MouseHandler extends MouseAdapter {
        public void mouseClicked(MouseEvent e) {
            canvas.addBall(e.getX(), e.getY());
        }
    }

    protected class SpeedSliderHandler implements ChangeListener {
        public void stateChanged(ChangeEvent e) {
            animator.setDelay(maxDelay - speedSlider.getValue());
            showStatus("Speed: " +
                Integer.toString(maxDelay - animator.getDelay()));
        }
    }
}
```