# Plan

- single circle
- multiple circles
- resizable circles
- multiple moving circles, colliding with borders
- multiple moving circles, colliding with borders and with each other

# Short keys

- Ctrl + " / " - comment or uncomment code
- Ctrl + " f " – search in file
- alt + "enter" -  get a hint

# Single circle

# SingleCircle.java

```
22
23 @    private static void init(Panel panel) {
24          Circle circle = new Circle();
25
26          circle.x = 400;
27          circle.y = 300;
28          circle.radius = 50;
29          circle.color = color( red: 100,  green: 200,  blue: 40);
30
31          panel.addCircle(circle);
32      }
33  }
```

# Multiple circles

# MultipleCircles.java

```
22 @    private static void init(Panel panel) {
23
24          //circle-1
25          Circle circle1 = new Circle();
26          circle1.x = 70;
27          circle1.y = 90;
28          circle1.radius = 40;
29          circle1.color = color( red: 100,  green: 200,  blue: 40);
30          panel.addCircle(circle1);
31
32          //circle-2
33          Circle circle2 = new Circle();
34          circle2.x = 300;
35          circle2.y = 100;
36          circle2.radius = 60;
37          circle2.color = color( red: 200,  green: 40,  blue: 10);
38          panel.addCircle(circle2);
39
40          //circle-3
41          Circle circle3 = new Circle();
42          circle3.x = 700;
43          circle3.y = 300;
44          circle3.radius = 80;
45          circle3.color = color( red: 150,  green: 110,  blue: 80);
46          panel.addCircle(circle3);
47      }
48  }
```

# Multiple circles with cycle

# MultipleCircles.java

```
15
16      private static void init(ResizingPanel resizingPanel) {
17          for (int i = 1; i < 100; i++) {
18              Circle circle = new Circle();
19
20              circle.x = random(50, 750);
21              circle.y = random(50, 550);
22              circle.radius = random(10, 50);
23              circle.minimize = randomBoolean();
24              circle.color = color(random(0, 256), random(0, 256), random(0, 256));
25
26              resizingPanel.addCircle(circle);
27          }
28      }
29
```

# Multiple resizable circles

## ResizingPanel.java

```
29  @       private void resizeCircle(Circle circle) {
30              if (circle.radius == 100) {
31                  circle.minimize = true;
32              }
33
34              if (circle.radius == 10) {
35                  circle.minimize = false;
36              }
37
38              if (circle.minimize == true) {
39                  circle.radius = circle.radius - 1;
40              } else {
41                  circle.radius = circle.radius + 1;
42              }
43          }
```

# Moving circles

# CollisionPanel.java

```
25
26  @       private void move(Circle circle) {
27              circle.x = circle.x + circle.speed.x;
28              circle.y = circle.y + circle.speed.y;
29          }
```

# Circles colliding with borders

## CollisionPanel.java

```
25
26      private void move(Circle circle) {
27          collideWithBorders(circle);
28
29          circle.x = circle.x + circle.speed.x;
30          circle.y = circle.y + circle.speed.y;
31      }
32
33  @   private void collideWithBorders(Circle circle) {
34          boolean isBottomCollision = (circle.y + circle.radius) >= Const.MAX_HEIGHT;
35          boolean isTopCollision = (circle.y - circle.radius) <= 0;
36          boolean isLeftCollision = (circle.x - circle.radius) <= 0;
37          boolean isRightCollision = (circle.x + circle.radius) >= Const.MAX_WIDTH ;
38
39          if (isBottomCollision || isTopCollision) {
40              circle.speed.y = -circle.speed.y;
41              relocate(circle);
42          }
43
44          if (isLeftCollision || isRightCollision) {
45              circle.speed.x = -circle.speed.x;
46              relocate(circle);
47          }
48      }
```

# Circles colliding with each other

## CollisionPanel.java

```
26      private void move(Circle circle) {
27          collideWithBorders(circle);
28          collideWithOtherCircle(circle);
29
30          circle.x = circle.x + circle.speed.x;
31          circle.y = circle.y + circle.speed.y;
32      }
```

# Circles colliding with each other

## CollisionPanel.java

$$u=\frac{v_1(m_1-m_2)+2m_2v_2}{m_1+m_2}$$

$$newVelocityX=\frac{c1.speed.x*(c1.mass-c2.mass)+2*c2.mass*c2.speed.x}{c1.mass+c2.mass}$$

```
80  @   private double getNewVelocityX(Circle c1, Circle c2) {
81          double newVelocityX = (c1.speed.x * (c1.mass - c2.mass) + (2 * c2.mass * c2.speed.x)) / (c1.mass + c2.mass);
82
83          return newVelocityX;
84      }
```

# Circles colliding with each other

## CollisionPanel.java

$$u=\frac{v_1(m_1-m_2)+2m_2v_2}{m_1+m_2}$$

$$newVelocityY=\frac{c1.speed.y*(c1.mass-c2.mass)+2*c2.mass*c2.speed.y}{c1.mass+c2.mass}$$

```
86  @   private double getNewVelocityY(Circle c1, Circle c2) {
87          double newVelocityY = (c1.speed.y * (c1.mass - c2.mass) + (2 * c2.mass * c2.speed.y)) / (c1.mass + c2.mass);
88
89          return newVelocityY;
90      }
```