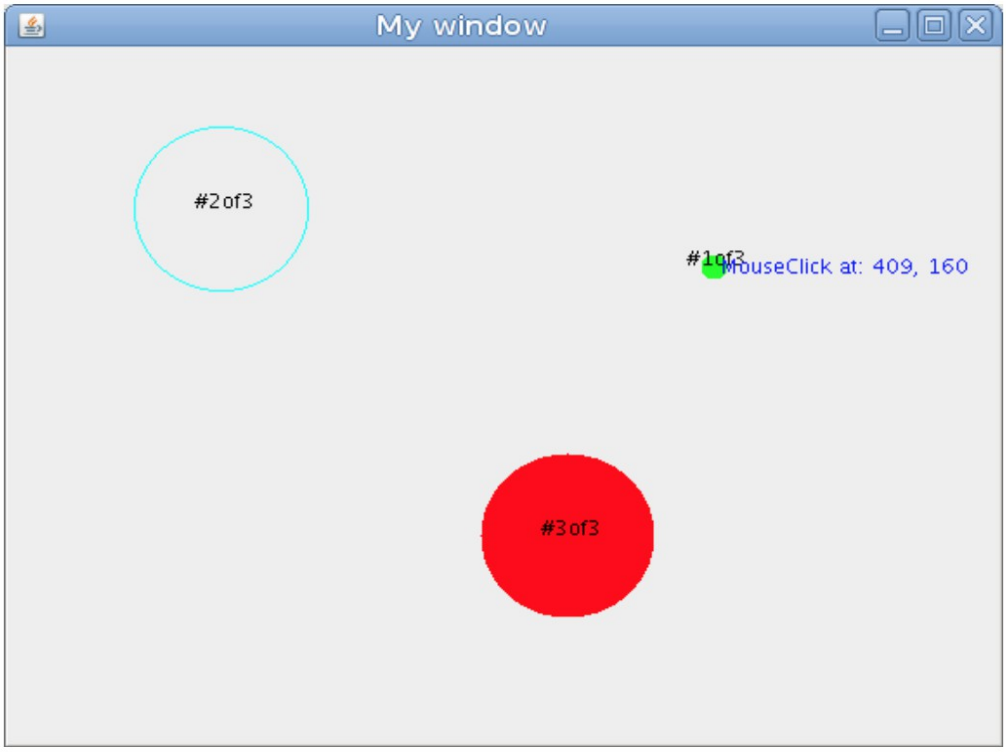
**Uebung 1. Java Wiederholung.**

**1. Ziel:** Erstellen wir ein Projekt mit graphischer Oberfläche. Bauen wir ein einfaches Spiel.

**2. Aufgabenstellung**

2.1. Erstellen wir einen Projekt, wo verschiedene Arten von Objekten (Kreis, gefüllte Kreis, Quadrat, etc.) dargestellt werden. Jeder Objekt muss in der Lage sein, auf Mausklicks auf dem Fenster zu reagieren und entsprechend seine Größe zu ändern. Wenn der Klick-Punkt nah liegt, muss ein Objekt seine Abmessungen reduzieren. Der Objekt, der von den Klick-Punkt weit entfernt wurde, muss die Abmessungen vergrössern.




**2.2. Source Code Example**

```
/*****************************************************************************/
Klasse Main
        package ue_1;
        public class Main {
            public static void main(String[] args) {
                WindowApp w1 = new WindowApp("MyWindow");
                w1.setVisible(true);
                w1.setSize(800, 600);
            }
        }
Klasse Coord
        package ue_1;
        class Coord {
            private double x;
            private double y;
            double getX() { return this.x; }
            double getY() { return this.y; }
            void setX(double x) { this.x = x; }
            void setY(double y) { this.y = y; }
            public Coord() { x = 0.0; y = 0.0; }
            public Coord(double x, double y) { this.x = x; this.y = y; }
            public Coord(Coord p) { this.x = p.getX(); this.y = p.getY(); }
            public double getDistance(Coord that) {
                double dx, dy;
                dx = this.x - that.getX();
                dy = this.y - that.getY();
                return Math.sqrt(dx * dx + dy * dy);
            }
        }
Klasse GraphicObject
        package ue_1;
        import java.awt.Graphics;
        abstract class GraphicObject {
            static int cnt = 0;
            static Coord maxCoordinate = new Coord(1.0, 1.0);
            int exemplarID = 0;
            Coord coordinate;
            protected Coord step = new Coord();
```

```
            protected int size = 15;
            protected Color c = Color.BLUE;
            private Random rnd = new Random();
            public GraphicObject() {
                coordinate = new Coord(rnd.nextInt((int)maxCoordinate.getX()),
                       rnd.nextInt((int)maxCoordinate.getY()));
                size = 15;
                cnt++;
                exemplarID = cnt;
                step.setX(rnd.nextInt(60)/10 - 3.0) ;
                step.setY(rnd.nextInt(60)/10 - 3.0) ;
                int r1 = rnd.nextInt(255);
                int g1 = rnd.nextInt(255);
                int b1 = rnd.nextInt(255);
                c = new Color(r1, g1, b1);
                size=30;
            }
            public GraphicObject(Coord p, int size) {
                this();
                coordinate.setX(p.getX());
                coordinate.setY(p.getY());
                this.size = size;
            }
            public void doStep(){
                coordMove((int)step.getX(), (int)step.getY());
                //coordCheck();
            }
            public void coordCheck(){
                //Der Objekt muss am Arbeitsfenster bleiben!
            }
            public void coordReplace(Coord p) {
                coordinate.setX(p.getX());
                coordinate.setY(p.getY());
            }
            public void coordMove(int x, int y) {
                coordinate.setX(coordinate.getX() + x);
                coordinate.setY(coordinate.getY() + y);
            }
            public double getX() { return coordinate.getX(); }
            public double getY() { return coordinate.getY(); }
            public Coord coordGet() { return coordinate; }
            public void setSize(int size) { this.size = size; }
            public int getSize() { return size;     }
            /**- bu Methode – wir sind ängstlich, aber brav! :-D  ----------*/
            protected void bu(Coord p) {
                double distance = p.getDistance(this.coordinate);
                if (distance > 3 * this.size) {
                    size *= 2;
                } else if (distance < this.size / 2) {
                    size /= 2;
                }
            }
            public abstract void draw(Graphics g);
        }
```

Klasse Dot

```
        package ue_1;
        import java.awt.Color;
        import java.awt.Graphics;
        /**=Klasse Dot -  ein gefültes Zirkel =========================*/
        class Dot extends GraphicObject {
            public Dot() { super(); }
            public Dot(Coord p, int size) { super(p, size); }
            public void draw(Graphics g) {
                g.setColor(c);
                g.fillOval((int)this.coordinate.getX() - size / 2, (int)coordinate.getY() - size / 2,
                        size, size);
                g.setColor(Color.BLACK);
                g.drawString("#" + exemplarID + "of" + cnt,
                        (int)coordinate.getX() - 15, (int)coordinate.getY());
            }
        }
```

Klasse Circle

```
        package ue_1;
        import java.awt.Color;
        import java.awt.Graphics;
        /**= Klasse Circle - ein leeres Zirkel ==================*/
        class Circle extends GraphicObject {
            public Circle() { super(); }
            public Circle(Coord p, int size) { super(p, size); }
            public void draw(Graphics g) {
                g.drawOval((int)this.coordinate.getX() - size / 2, (int)coordinate.getY() - size / 2,
                        size, size);
                g.setColor(Color.BLACK);
                g.drawString("#" + exemplarID + "of" + cnt,
                        (int)coordinate.getX() - 15, (int)coordinate.getY());
            }
        }
```

**Klasse WindowApp**

```
package ue_1;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JFrame;
import javax.swing.Timer;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class WindowApp extends JFrame {
    private Timer t1;
    private ActionListener doIt;
    private Coord clickCoord;
    private Dot d1 = new Dot();
    private Circle c1 = new Circle(new Coord(125, 125), 100);
    private Dot d2 = new Dot(new Coord(325, 325), 100);
    private Dot[] d;
    private class MyWindowsListener extends WindowAdapter {
        public void windowClosing(WindowEvent ev) {
            System.exit(0);
        }
    }
    private class MyMouseListener implements MouseListener {
        public void mouseClicked(MouseEvent ev) {        }
        public void mousePressed(MouseEvent ev) {
            clickCoord.setX(ev.getX());
            clickCoord.setY(ev.getY());
            repaint();
        }
        public void mouseReleased(MouseEvent ev) {         }
        public void mouseExited(MouseEvent ev) {         }
        public void mouseEntered(MouseEvent ev) {         }
    }
    public WindowApp(String s) {
        super(s);
        addWindowListener(new MyWindowsListener());
        addMouseListener(new MyMouseListener());
        setSize(400, 400);
        setVisible(true);
        GraphicObject.maxCoordinate.setX(400);
        GraphicObject.maxCoordinate.setY(400); //Spielplatz Abmessungen Definierung
        clickCoord = new Coord();
        d = new Dot[5];
        for(int i =0; i<d.length; i++){
            d[i] = new Dot();
        }
        doIt = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                d2.coordMove(-1, -2);
                GraphicObject.maxCoordinate.setX(getWidth());
                GraphicObject.maxCoordinate.setY(getHeight());
                for(int i =0; i<d.length; i++){
                    d[i].doStep();
                }
                repaint();
            }
        };
        t1 = new Timer(500, doIt);
        t1.start();
    }
    public void paint(Graphics g) {
        //clear window
        super.paint(g);
        //object state determination
        c1.bu(clickCoord);
        d2.bu(clickCoord);
        d1.coordReplace(clickCoord);
        //show objects
        g.setColor(Color.CYAN);
        c1.draw(g);
        g.setColor(Color.RED);
        d2.draw(g);
        g.setColor(Color.GREEN);
        d1.draw(g);
        //show a string
        g.setColor(Color.BLUE);
        g.drawString("MouseClick at: " + clickCoord.getX() + ", "
                + clickCoord.getY(), (int)clickCoord.getX() + 5, (int)clickCoord.getY() + 5);
        for(int i =0; i<d.length; i++){
            d[i].draw(g);
        }
    }
}
```

**2.3. Ihre Aufgabe:**

- Untersuchen Sie den gegebenen Beispiel.
- Analysieren Sie das Zusammenspiel von allen Objekten in der Anwendung.
- Starten Sie den Projekt.
- Analysieren und beschreiben Sie, wie das Objekt Timer in der Anwendung funktioniert.
- Ändern Sie den Projekt so, das Objekte immer am Arbeitsfeld bleiben.
- Ändern Sie den Projekt so, das am Arbeitsfeld 5 Exemplare von Circle und 5 Exemplare von Dot bewegt werden.
- * Machen Sie eine Anwendung, die einen Feuerwerk simuliert. 100 Objekte müssen am Klick-Punkt generiert werden und in allen Richtungen sich bewegen. Objekte müssen von Fenstergrenzen abgespiegelt werden.
- * Simulieren Sie eine Gravitation in Ihrem Feuerwerk.
- * Objekte müssen ihre kinetische Energie beim Einstoss am Hindernis teilweise verlieren.