EINFÜHRUNG IN DIE PROGRAMMIERUNG

GRAFIK- UND NETZWERKPROGRAMMIERUNG

Tobias Witt

28.03.2014

SWING

FENSTER

```
public class Main extends JFrame {
    public Main() {
        add(new Board());
        setTitle("Wurm");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(300, 280);
        setResizable(false);
        setVisible(true);
    }
    public static void main(String[] args) {
        new Main();
    }
}
```

SPIELFELD

```
public class Board extends JPanel {
    @Override
    public void paintComponent(Graphics g) {
      super.paintComponent(g);
      Graphics2D g2 = (Graphics2D) g;

      RenderingHints rh =
              new RenderingHints(RenderingHints.KEY_ANTIALIASING,
                                 RenderingHints.VALUE_ANTIALIAS_ON);
      rh.put(RenderingHints.KEY_RENDERING,
              RenderingHints.VALUE_RENDER_QUALITY);

      g2.setRenderingHints(rh);

      Ellipse2D e = new Ellipse2D.Double(0, 0, 80, 130);
      g2.setStroke(new BasicStroke(1));
      g2.setColor(Color.gray);
      g2.draw(e);
    }
}
```

Wird nicht vom Programmierer gerufen

BILDER

```
public class Board extends JPanel {

    private Image wurm;

    public Board() {
        ImageIcon ii = new ImageIcon("wurm.png");
        wurm = ii.getImage();
    }

    @Override
    public void paintComponent(Graphics g) {
        // ...
        Graphics2D g2d = (Graphics2D) g;
        g2d.drawImage(wurm, 10, 10, null);
    }
}
```

BILDER


IN SNOPPYQUOP LAND
WHERE NOTHING SEEMS QUEER
THE ANIMATED CARTOONIST
IDEAS

ANIMATION

VIEW

```
public abstract class View {
    protected final Image image;

    public View(String imageName) {
        ImageIcon ii = new ImageIcon(
                "res/" + imageName);
        image = ii.getImage();
    }

    public abstract void draw(Graphics2D g2d);
}
```

WURM VIEW

```
public class WurmView extends View {
    private Wurm wurm;

    public WurmView(Wurm wurm) {
        super("wurm.png");
        this.wurm = wurm;
    }

    @Override
    public void draw(Graphics2D g2d) {
        g2d.drawImage(image,
                wurm.getPosition().x,
                wurm.getPosition().y, null);
    }
}
```

MODEL

```
public class Wurm {
    private Point position;

    public Wurm(int x, int y) {
        this(new Point(x, y));
    }

    public Wurm(Point position) {
        this.position = position;
    }

    public Point getPosition() {
        return position;
    }
}
```

MODEL

```
public class Wurm {
    // ...
    public void moveX(int step) {
        position = new Point(position.x + step,
                position.y);
    }

    public void moveY(int step) {
        position = new Point(position.x,
                position.y + step);
    }
}
```

BOARD

```
public class Board extends JPanel {
    private WurmView wurmView;
    // ...
    public Board(WurmView wurmView) {
        this.wurmView = wurmView;
        // ...
        setBackground(Color.black);
        setDoubleBuffered(true);
    }

    @Override
    public void paintComponent(Graphics g) {
        // ...
        wurmView.draw((Graphics2D) g);
    }
}
```

SWING TIMER

```
import javax.swing.Timer;

public class Board extends JPanel {
    private Timer timer;
    public Board(WurmView wurmView) {
        // ...
        timer = new Timer(40, this); // 25 fps
        timer.start();
    }

    @Override
    public void paintComponent(Graphics g) {
        // ...
        wurmView.draw((Graphics2D) g);
    }
}
```

Type error: ActionListener expected

SWING TIMER

```
import javax.swing.Timer;

public class Board extends JPanel implements ActionListener {
    // ...
    @Override
    public void actionPerformed(ActionEvent e) {
        wurmView.getWurm().moveX(1);
        wurmView.getWurm().moveY(1);

        repaint();
    }
}
```

Weist Komponente dazu an, dass `repaintComponent(Graphics g)` gerufen werden soll

THREAD

```
public class Board extends JPanel implements Runnable {
    private Thread animator;
    // ...
        animator = new Thread(this); // 25 fps
        animator.start();
    // ...

    @Override
    public void run() {
        wurmView.getWurm().moveX(1);
        wurmView.getWurm().moveY(1);

        repaint();
    }
}
```

run() nur
einmal ausgeführt

THREAD

```
public class Board extends JPanel implements Runnable {
    // ...
        long before = System.currentTimeMillis(),
                        now, sleep;
        while (true) {
            wurmView.getWurm().moveX(1);
            wurmView.getWurm().moveY(1);
            repaint();

            now = System.currentTimeMillis();
            sleep = Math.max(0,
                    40 - (now - before)); // 25 fps
            try {
                Thread.sleep(sleep);
            } catch (InterruptedException e) {
            }
            before = System.currentTimeMillis();
    }
}
```

SPIEL

```
public class Star extends JFrame {

    public Star(Board board) {
        add(board);
        // ...
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            Wurm wurm = new Wurm(10, 10);
            WurmView view = new WurmView(wurm);
            Board board = new Board(view);
            Star star = new Star(board);
            star.setVisible(true);
        });
    }
}
```

BEWEGUNG

WURM

```
public class Wurm implements Model {
    // ...
    private int vx = 0, vy = 0;
    // ...
    public void setVx(int vx) {
        this.vx = vx;
    }

    public void setVy(int vy) {
        this.vy = vy;
    }
```

MODEL

```
public interface Model {
    public void move();
}
```

WURM

```
public class Wurm implements Model {
    // ...
    private int vx = 0, vy = 0;
    // ...
    public void setVx(int vx) {
        this.vx = vx;
    }

    public void setVy(int vy) {
        this.vy = vy;
    }

    @Override
    public void move() {
        position.x += vx;
        position.y += vy;
    }
}
```

VIEW

```
public abstract class View<M extends Model> implements KeyListener {
    protected final Image image;
    protected final M model;

    public View(String imageName, M model) {
        ImageIcon ii = new ImageIcon("res/" + imageName);
        image = ii.getImage();
        this.model = model;
    }

    public abstract void draw(Graphics2D g2d);

    public void move() {
        model.move();
    }
}
```

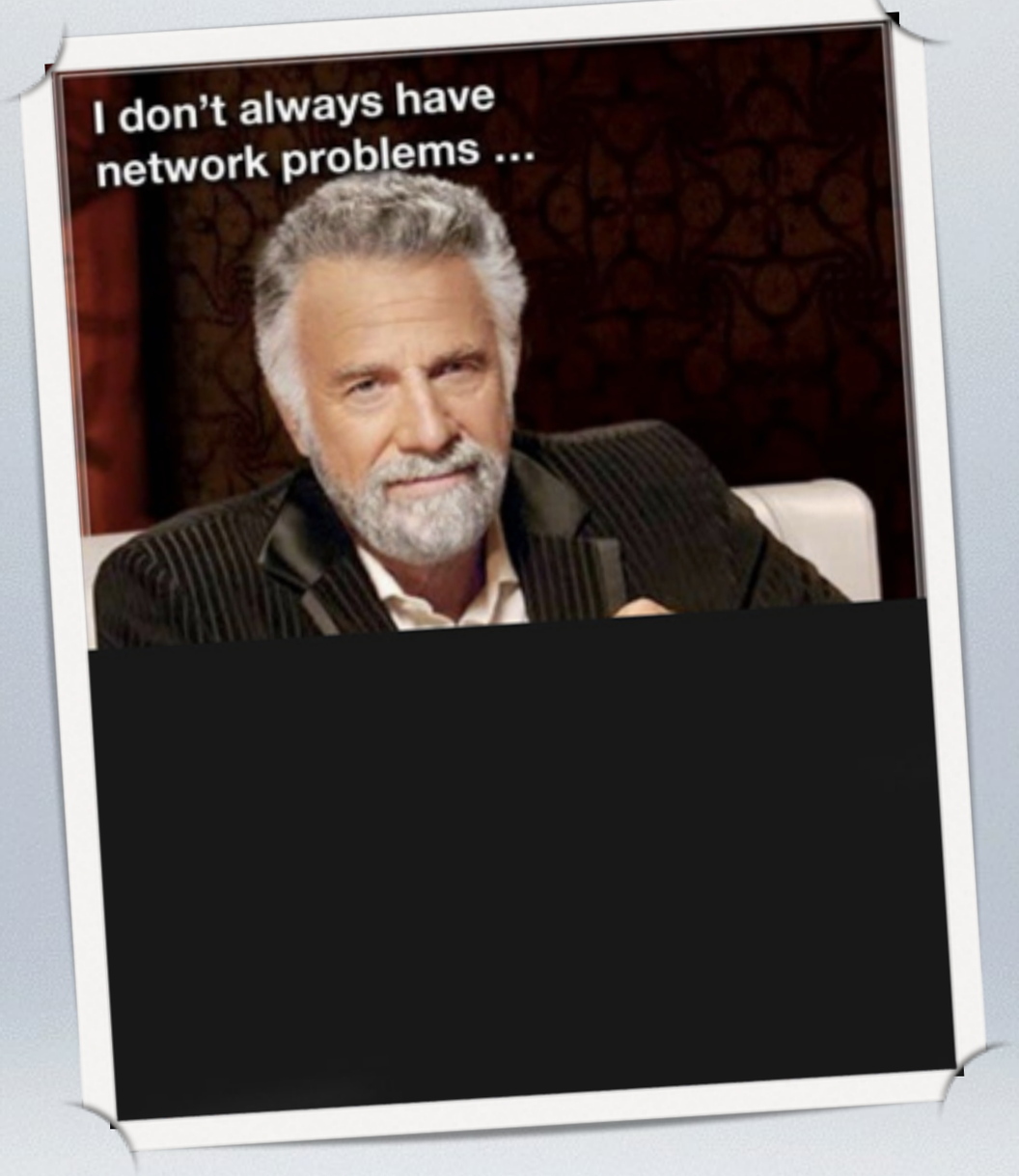
WURMVIEW

```
public class WurmView extends View<Wurm> {
    public WurmView(Wurm wurm) {
        super("wurm.png", wurm);
    }
    // ...
    @Override
    public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_LEFT: model.setVx(-1); break;
            case KeyEvent.VK_RIGHT: model.setVx(1); break;
            case KeyEvent.VK_UP: model.setVy(-1); break;
            case KeyEvent.VK_DOWN: model.setVy(1); break;
        }
    }
    @Override
    public void keyReleased(KeyEvent e) {
        switch (e.getKeyCode()) {
            // ...
            case KeyEvent.VK_LEFT: model.setVx(0);
        }
    }
}
```

BOARD

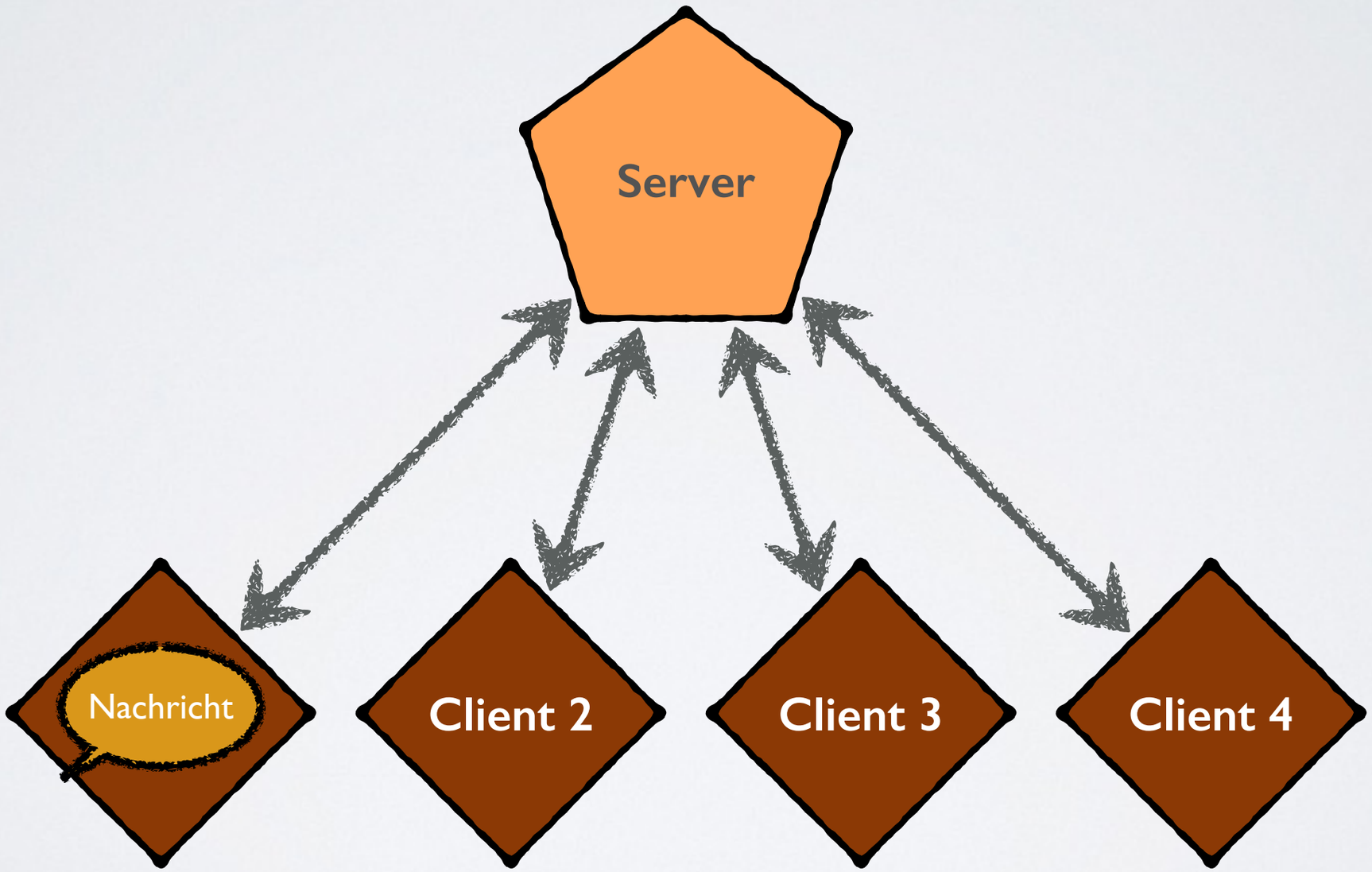
```
public class Board extends JPanel implements Runnable {
    private View<Wurm> wurmView;
        // ...
    public Board(View<Wurm> wurmView) {
        // ...
        addKeyListener(wurmView);
        setFocusable(true);
    }

    @Override
    public void run() {
        // ...
        while (true) {
            moveObjects();
            // ...
    private void moveObjects() {
        wurmView.move();
    }
}
```

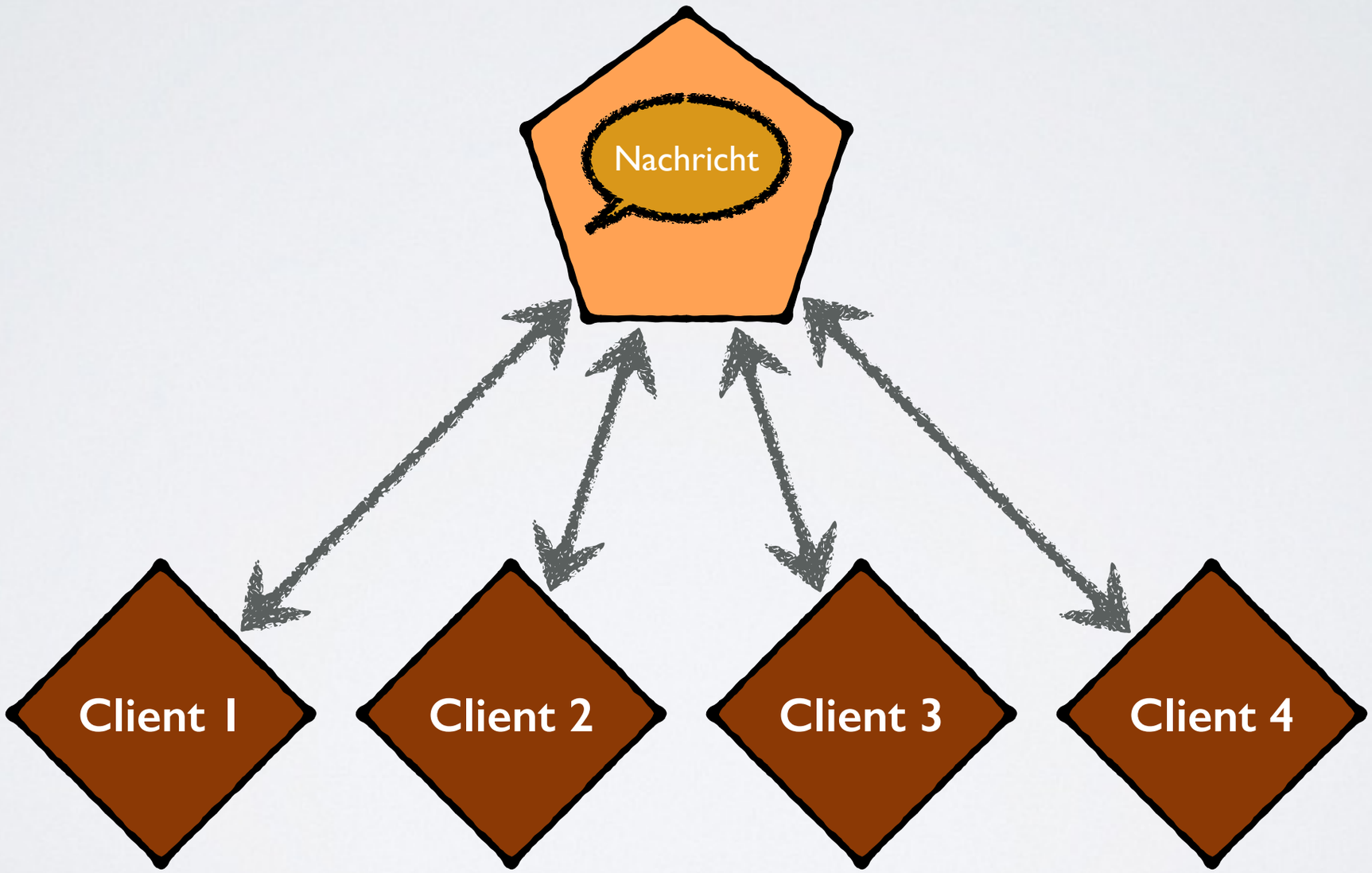


NETZWERK

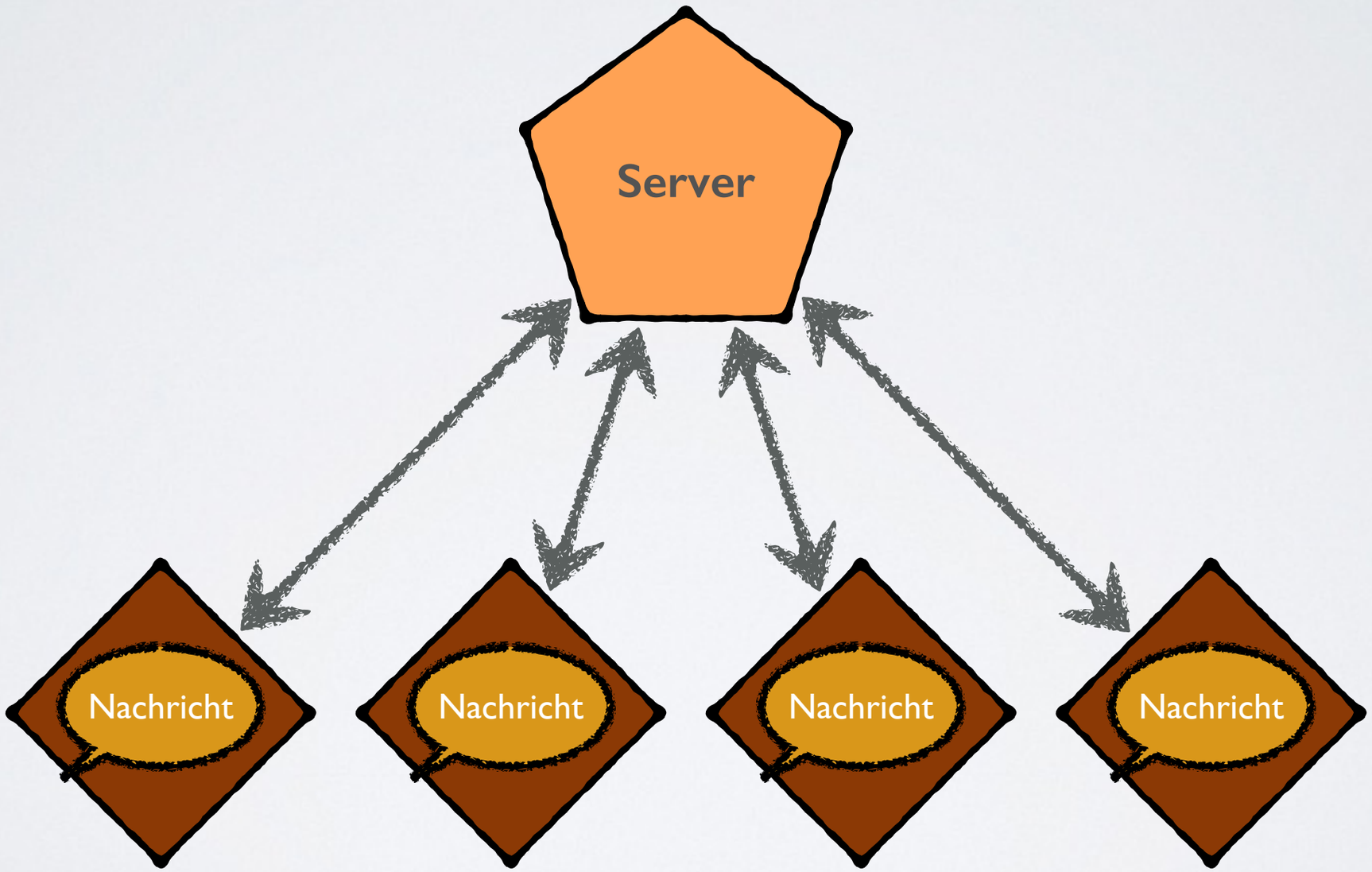
CLIENT-SERVER



CLIENT-SERVER



CLIENT-SERVER



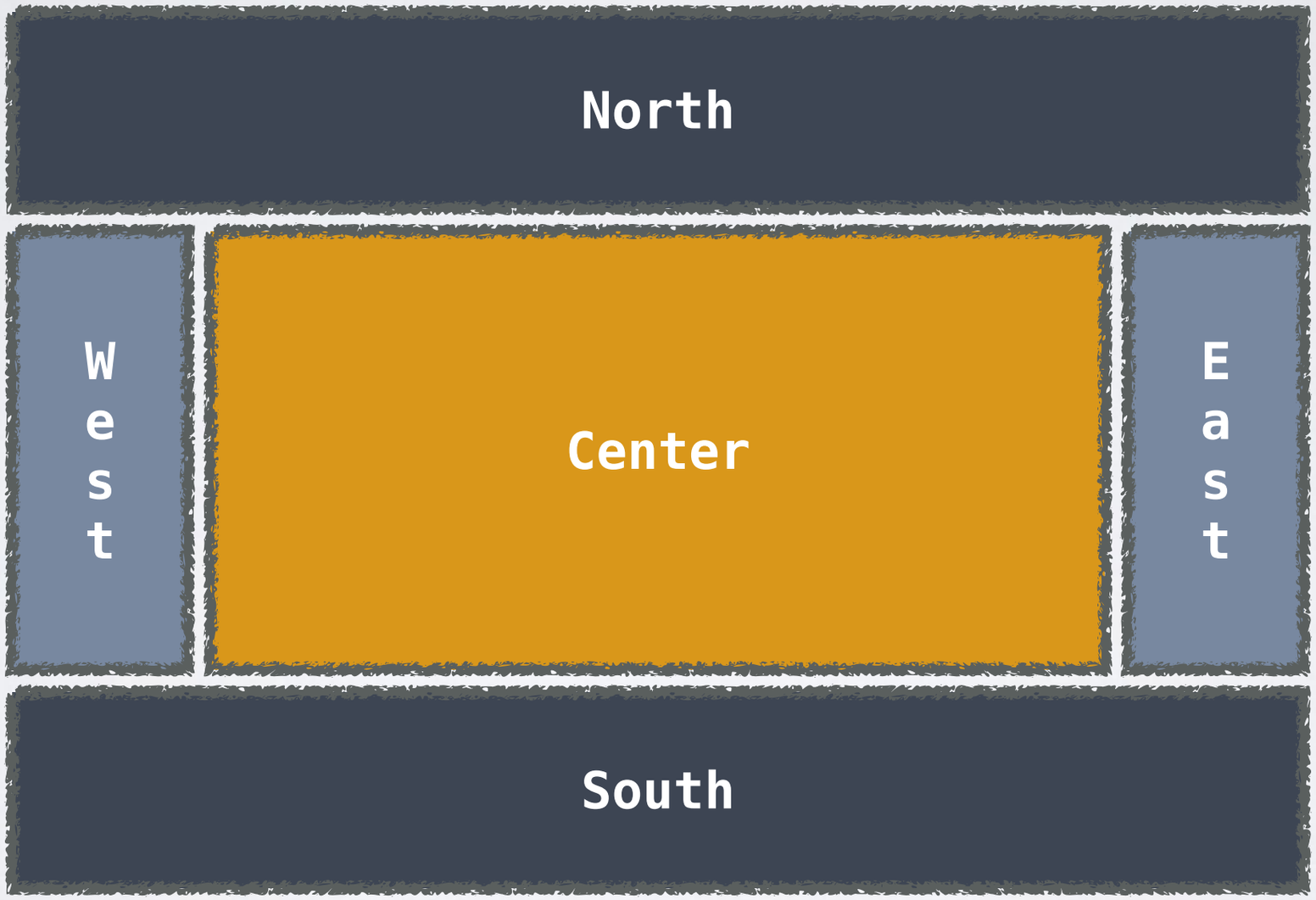
CLIENT

```
public class Client extends JFrame {
    BufferedReader in;
    PrintWriter out;
    JTextField textField = new JTextField(40);
    JTextArea messageArea = new JTextArea(8, 40);

    public Client() {
        messageArea.setEditable(false);
        textField.addActionListener((e) -> {
            out.println(textField.getText());
            textField.setText("");
        });

        getContentPane().add(textField, "North");
        getContentPane().add(new JScrollPane(messageArea), "Center");
```

BORDERLAYOUT



CLIENT

```
public class Client extends JFrame {
    BufferedReader in;
    PrintWriter out;
    JTextField textField = new JTextField(40);
    JTextArea messageArea = new JTextArea(8, 40);

    public Client() {
        messageArea.setEditable(false);
        textField.addActionListener((e) -> {
            out.println(textField.getText());
            textField.setText("");
        });

        getContentPane().add(textField, "North");
        getContentPane().add(new JScrollPane(messageArea), "Center");

        setTitle("Chat");
        pack();
    }
```

CLIENT

```
private String showInputDialog(String message, String title) {
    return JOptionPane.showInputDialog(
            this, message, title, JOptionPane.PLAIN_MESSAGE);
}

private String getServerAddress() {
    return showInputDialog("IP Adresse eingeben", "Willkommen");
}

private String getUserName() {
    return showInputDialog("Name eingeben", "Name");
}
```

CLIENT

```
private void run() throws IOException {
    String serverAddress = getServerAddress();
    Socket socket = new Socket(serverAddress, 9001);
    in = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
    out = new PrintWriter(socket.getOutputStream(), true);

    while (true) {
        String line = in.readLine();
        if (line.startsWith("SUBMITNAME")) {
            out.println(getUserName());
        } else if (line.startsWith("NAMEACCEPTED")) {
            messageArea.append("Verbunden mit " + serverAddress);
        } else if (line.startsWith("MESSAGE")) {
            messageArea.append(line.substring(8) + "\n");
        }
    }
}
```

CLIENT MAIN-METHODE

```
public static void main(String[] args) throws Exception {
    Client client = new Client();
    client.setVisible(true);
    client.run();
}
```

SERVER

```
public class Server {
    private static final int PORT = 9001;

    private static HashSet<String> names = new HashSet<String>();
    private static HashSet<PrintWriter> writers =
            new HashSet<PrintWriter>();

    public static void main(String[] args) throws Exception {
        System.out.println("The chat server is running.");
        try(ServerSocket listener = new ServerSocket(PORT)) {
            while (true) {
                new Thread(new Handler(listener.accept())).start();
            }
        }
    }
```

CONNECTION HANDLER

```
private static class Handler implements Runnable {
    private Socket socket;

    public Handler(Socket socket) {
        this.socket = socket;
    }

    public void run() {
        try(ClientConnection connection =
                new ClientConnection(socket)) {
            connection.connect();
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

CLIENT CONNECTION

```
private static class ClientConnection implements AutoCloseable {
    private final BufferedReader in;
    private final PrintWriter out;
    private final Socket socket;
    private String name;

    public ClientConnection(Socket socket) throws IOException {
        this.socket = socket;
        in = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(socket.getOutputStream(), true);
    }
```

CLIENT CONNECTION

```
public void connect() throws IOException {
    getName();
    out.println("NAMEACCEPTED");
    writers.add(out);
    broadcast();
}
```

CLIENT CONNECTION

```
private void getName() throws IOException {
    while (true) {
        out.println("SUBMITNAME");
        name = in.readLine();
        if (name == null) {
            return;
        }
        synchronized (names) {
            if (!names.contains(name)) {
                names.add(name);
                break;
            }
        }
    }
}
```

CLIENT CONNECTION

```
private void broadcast() throws IOException {
    while (true) {
        String input = in.readLine();
        if (input == null) {
            return;
        }
        synchronized (writers) {
            for (PrintWriter writer : writers) {
                writer.println("MESSAGE " + name + ": " + input);
            }
        }
    }
}
```

AUTOCLOSEABLE

```
@Override
public void close() throws Exception {
    if (name != null) {
        names.remove(name);
    }
    if (out != null) {
        writers.remove(out);
    }
    try {
        socket.close();
    } catch (IOException e) {
    }
}
```

BILDQUELLEN

Bild	Quelle
Swing	http://www.flickr.com/photos/18276635@N00/2538693500/in/photolist-4Skt8A-4ZgwYT-5c8sg3-5nR7uh-5xZDXi-5y4Zkd
Animation	http://2.bp.blogspot.com/-Tpd5sjM_DBE/UEB5jn5FG3I/AAAAAAAAAU_c/uXDWHCPn7zU/s1600/SNOPPYQUOP.png
Bewegung	http://www.querbilder.de/cartoons/165_schnecke_stolpern.html
Netzwerk	http://joyreactor.com/post/467765