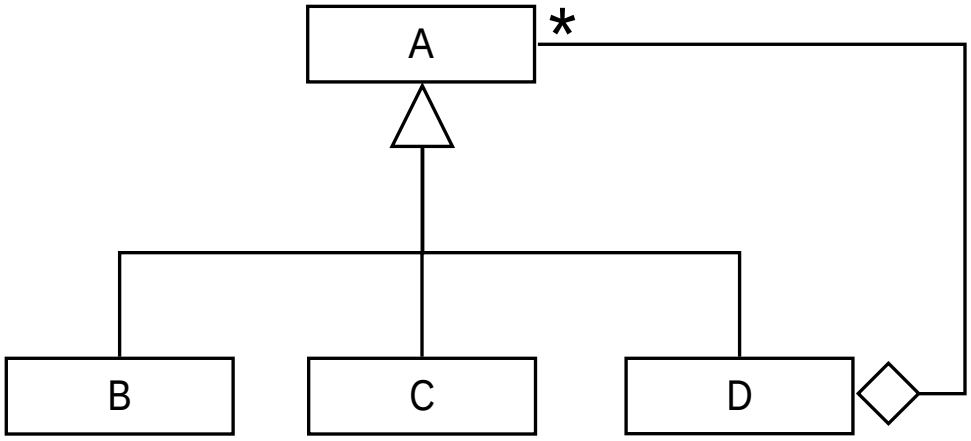
Lösningsförslag till tentamen

Kursnamn	**Objektorienterade applikationer**
Tentamensdatum	**2018-08-27**
Program	**DAI 2**
Läsår	**2017/2018, lp 3**
Examinator	**Uno Holmer**

Uppgift 1 (6 p)

Detta diagram är ett exempel på designmönstret Composite.



Uppgift 2 (8 p)

```
public class DieButton extends JButton implements ActionListener {
    private DieIcons icons;
    private static Random random = new Random();

    public DieButton() {
        icons = DieIcons.getInstance();
        addActionListener(this);
        roll();
    }
    private void roll() {
        setIcon(icons.getIcon(random.nextInt(icons.size())));
    }
    public void actionPerformed(ActionEvent e) { roll(); }
}
```

Uppgift 3 (12+10 p)

a)

```
public class Server {
    public static void main(String[] arg) {
        try {
            ServerSocket servSock =
                new ServerSocket(Integer.parseInt(arg[1]));
            while ( true )
                new ClientHandler(servSock.accept(),arg[0]);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public class ClientHandler {
    private DataInputStream inStream;
    private DataOutputStream outStream;
    private Socket socket;

    public ClientHandler(Socket socket,String fileName) {
        try {
            this.socket = socket;
            inStream =
                new DataInputStream(new FileInputStream(fileName));
            outStream =
                new DataOutputStream(socket.getOutputStream());
        }
        catch (IOException e) { e.printStackTrace(); }
        sendData();
    }

    private void sendData() {
        try {
            while ( true ) {
                try {
                    outStream.writeDouble(inStream.readDouble());
                }
                catch (EOFException e) {
                    inStream.close();
                    socket.close();
                    return;
                }
            }
        }
        catch (IOException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }
}
```

b)

```
public class Client {
    public static void main(String[] arg) {
        new Receiver(arg[0],Integer.parseInt(arg[1]),arg[2]);
    }
}

public class Receiver {
    private Socket socket;
    private DataInputStream inStream;
    private PrintWriter outFile;

    public Receiver(String address,int port,String fileName) {
        try {
            socket = new Socket(InetAddress.getByName(address),port);
            inStream = new DataInputStream(socket.getInputStream());
            outFile = new PrintWriter(new FileWriter(fileName));
        }
        catch (IOException e) { e.printStackTrace(); }
        getValues();
    }

    private void getValues() {
        while ( ! socket.isClosed() ) {
            try {
                outFile.println("" + inStream.readDouble());
            }
            catch (IOException e) {
                outFile.close();
                System.exit(0);
            }
        }
        outFile.close();
    }
}
```

Uppgift 4 (14 p)

```
public class Gui extends JFrame implements Observer {
    private JTextField placeField;
    private JTextField outputField;
    private Database database;

    public Gui(Database database) {
        this.database = database;
        database.addObserver(this);
        makeFrame();
    }

    private void makeFrame() {
        setTitle("Weather service");
        setLayout(new GridLayout(2,1));

        // Buttons
        JPanel buttonPanel = new JPanel();

        JButton tempButton = new JButton("temperature");
        tempButton.addActionListener(
            new ActionListener(){
                public void actionPerformed(ActionEvent e) {
                    computeTemperature();
                }
        });
        buttonPanel.add(tempButton);

        //...

        // Fields
        JPanel fieldPanel = new JPanel();
        placeField = new JTextField(8);
        fieldPanel.add(placeField);
        outputField = new JTextField(7);
        outputField.setEditable(false);
        fieldPanel.add(outputField);
        add(fieldPanel);

        pack();
        setVisible(true);
    }

    private void computeTemperature() {
        try {
            database.computeTemperature(placeField.getText());
        }
        catch (IllegalArgumentException e ) {
            errorMsg(e.getMessage());
        }
    }

    private void errorMsg(String msg) {
        JOptionPane.showMessageDialog(null,
                                      msg,
                                      "Wheather station",
                                      JOptionPane.ERROR_MESSAGE
        );
    }
```

```
    public void update(Observable o,Object arg) {
        if ( o instanceof Database && arg instanceof String ) {
            outputField.setText((String)arg);
        }
    }
}
```

Uppgift 5 (10 p)

```
public class Animation implements Drawable {
    private Drawable obj;

    Animation(Drawable obj) {
        this.obj = obj;
    }

    // Delegation
    public void draw(int rotationAngle) {
        obj.draw(rotationAngle);
    }

    public void erase() {
        obj.erase();
    }

    public void horizontalSpin(int revolutions,int rpm) {
        double rpms = rpm/60000.0;
        double timePerRotation = 1/rpms;
        double timePerStep = timePerRotation/60;
        for ( int r = 0; r < revolutions; r++ )
            for ( int angle = 0; angle < 360; angle += 6 ) {
                obj.draw(angle);
                try { Thread.sleep((int)timePerStep); }
                catch (Exception e) { return; }
            }
    }
}
```