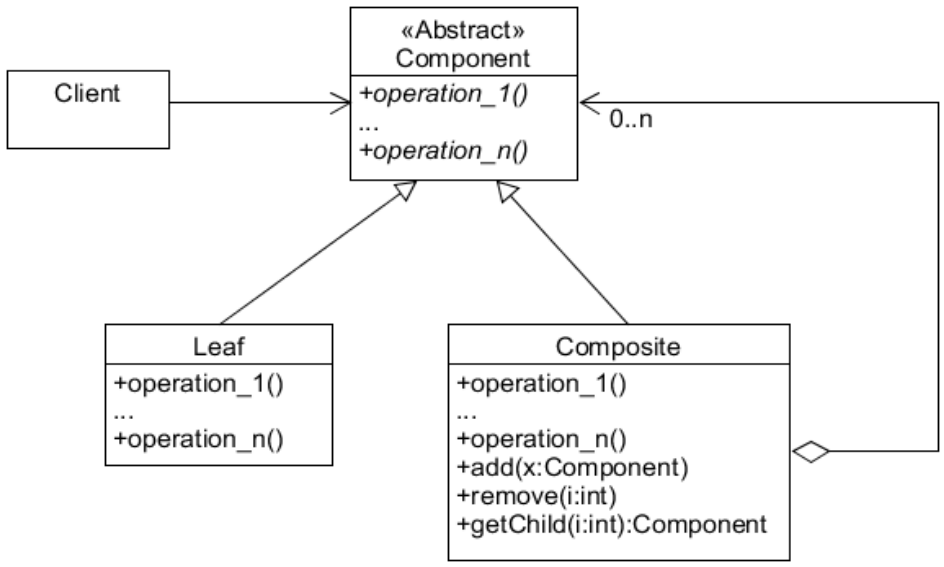
Lösningsförslag till tentamen *P r e l i m i n ä r*

Kursnamn	**Objektorienterade applikationer**
Tentamensdatum	**2017-03-16**
Program	**DAI2**
Läsår	**2016/2017, lp 3**
Examinator	**Uno Holmer**

Uppgift 1

a) (2 p)



b) (3 p)

```
public final class SingletonRandom extends Random {
    private static SingletonRandom instance = null;

    private SingletonRandom() {}

    public synchronized static SingletonRandom getInstance() {
        if ( instance == null )
            instance = new SingletonRandom();
        return instance;
    }
}
```

Uppgift 2 (6 p)

```
public class Die extends Observable {
    private int value;
    private static Random random = new Random();

    public void roll() {
        value = 1 + random.nextInt(6);
        setChanged();
        notifyObservers(value);
    }
}

public class Gui extends JFrame implements Observer {
    private JTextField textField;

    public Gui(Die die) {
        die.addObserver(this);
        setLayout(new FlowLayout());
        JButton button = new JButton("Roll the die");
        add(button);
        button.addActionListener(e -> die.roll());
        textField = new JTextField(1);
        textField.setEditable(false);
        add(textField);
        pack();
        setVisible(true);
    }

    public void update(Observable obs,Object o) {
        textField.setText("" + (Integer)o);
    }
}

...
public static void main(String[] arg) {
    Die die = new Die();
    Gui gui = new Gui(die);
    die.roll();
}
```

Uppgift 3 (10 p)

```
public class SkipIterator<E> implements Iterator<E> {
    private Iterator<E> it;

    public SkipIterator(Iterator<E> it) {
        this.it =  it;
    }

    @Override
    public boolean hasNext() { return it.hasNext(); }
    @Override
    public E next() { return it.next(); }
    @Override
    public void remove() { it.remove(); }
```

```
    public E skip(int n) {
        int skipped = 0;
        E result = it.next();
        while ( it.hasNext() && skipped < n ) {
            result = it.next();
            skipped++;
        }
        return result;
    }
}
```

Uppgift 4 (12 p)

```
public class Search {
    public static void main(String[] arg) throws IOException {
        if ( arg.length != 2 ) {
            System.out.println("Usage: java Search <directory> <string>");
            System.exit(0);
        }
        File f = new File(arg[0]);
        if ( f.isDirectory() )
            for ( File file : f.listFiles() )
                if ( isJavaSourceFile(file) )
                    search(file,arg[1]);
    }

    private static boolean isJavaSourceFile(File f) {
        return f.getName().endsWith(".java");
    }

    private static void search(File file,String pattern) throws IOException {
        boolean isFileNameWritten = false;
        Scanner sc = new Scanner(file);
        while ( sc.hasNextLine() ) {
            String line = sc.nextLine();
            if ( line.contains(pattern) ) {
                if ( ! isFileNameWritten ) {
                    System.out.println(file.getName() + ":");
                    isFileNameWritten = true;
                }
                System.out.println(line);
            }
        }
    }
}
```

Uppgift 5 (16 p)

```
public class Server {
    public static void main(String[] arg) {
        if ( arg.length != 1 ) {
            System.out.println("Usage: java Server <port number>");
            System.exit(0);
        }
        try {
            ServerSocket servSock =
                new ServerSocket(Integer.parseInt(arg[0]));
            while ( true )
                new ClientHandler(servSock.accept());
        }
        catch ( IOException e) {
            e.printStackTrace();
        }
    }
}

public class ClientHandler extends Thread {
    private Socket clientSocket;
    private DataInputStream inStream;
    private DataOutputStream outStream;
    private File dataStore;

    public ClientHandler(Socket clientSocket) {
        this.clientSocket = clientSocket;
        try {
            getStreams();
            openFile();
        }
        catch ( IOException e) {
            System.out.println("IO error on server startup.");
            throw new IllegalStateException();
        }
        start();
    }

    private void getStreams() throws IOException {
        inStream = new DataInputStream(clientSocket.getInputStream());
        outStream = new DataOutputStream(clientSocket.getOutputStream());
    }

    private void openFile() throws IOException {
        String clientIP = clientSocket.getInetAddress().getHostAddress();
        dataStore = new File(clientIP + ".txt");
        dataStore.createNewFile();
    }

    @Override
    public void run() {
        try {
            while ( ! interrupted() ) {
                int clientValue = inStream.readInt();
                int newMax = getMaxValue(clientValue);
                ...
```

```
                updateMaxValue(newMax);
                outStream.writeInt(newMax);
            }
        }
        catch ( IOException e) {
            System.out.println("IO error in server. Terminating.");
            return;
        }
    }

    private int getMaxValue(int clientValue) throws IOException {
        int newMax = clientValue;
        Scanner sc = new Scanner(dataStore);
        if ( sc.hasNextInt() ) {
            int oldMax = sc.nextInt();
            newMax = Math.max(clientValue,oldMax);
        }
        sc.close();
        return newMax;
    }

    private void updateMaxValue(int newMax) throws IOException {
        PrintWriter pw = new PrintWriter(dataStore);
        pw.print("" + newMax);
        pw.close();
    }
}
```

Uppgift 6 (11 p)

```
public static JMenu createMenu(String configFile) throws IOException {
    Scanner sc = new Scanner(new File(configFile));
    JMenu menu = null;
    if ( sc.hasNextLine() )
        menu = new JMenu(sc.nextLine());
    while ( sc.hasNextLine() ) {
        String[] buf = sc.nextLine().split(":");
        String itemName = buf[0];
        String className = buf[1];
        menu.add(createMenuItem(itemName,className));
    }
    return menu;
}

private static JMenuItem createMenuItem(String itemName,String className) {
    JMenuItem item = null;
    try {
        Class c = Class.forName(className);
        MenuAction action = (MenuAction)(c.newInstance());
        item = new JMenuItem(itemName);
        item.addActionListener(e -> action.fire());
    }
    catch ( ReflectiveOperationException e ) {
        e.printStackTrace();
        System.exit(0);
    }
    return item;
}
```