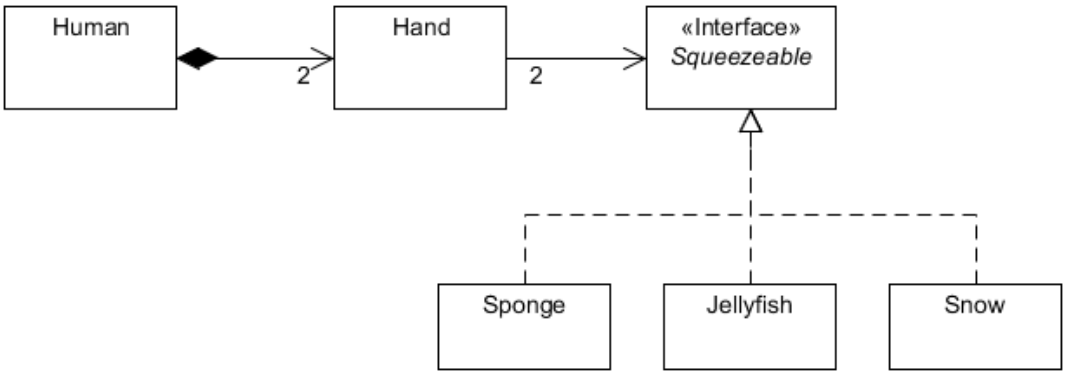
Lösningsförslag till tentamen *P r e l i m i n ä r*

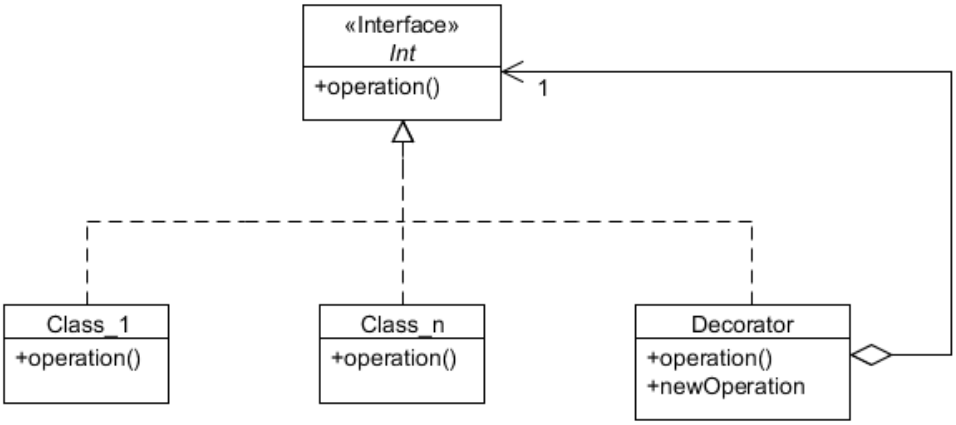
Kursnamn	**Objektorienterade applikationer**
Tentamensdatum	**2013-03-14**
Program	**DAI2**
Läsår	**2012/2013, lp 3**
Examinator	**Uno Holmer**

Uppgift 1

a) (5 p)



b) (5 p)



Uppgift 2

a) (3 p)

```
public class CellView extends JButton implements Observer {
    public void update(Observable obj,Object o) {
        if ( obj instanceof Cell && o instanceof String )
            setText((String)o);
    }
}
```

b) (10 p)

```
public class Gui extends JFrame {
    public Gui(GameLogic model,GameController controller) {
        setTitle("Tic-Tac-Toe");
        final int size = model.getSize();
        setLayout(new GridLayout(size,size));
        final GameController contr = controller;
        for ( int row = 0; row < size; row++ )
            for ( int col = 0; col < size; col++ ) {
                final int r = row,c = col;
                CellView cw = new CellView();
                model.addCellObserver(r,c,cw);
                cw.addActionListener(
                    new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            contr.select(r,c);
                        }
                    });
                add(cw);
            }
        setLocationRelativeTo(null);
        setPreferredSize(new Dimension(200,200));
        pack();
        setVisible(true);
    }
}
```

c) (3 p)

```
public static void main(String[] arg) {
    GameLogic model = new GameLogic(3);
    GameController controller = new GameController(model);
    new Gui(model,controller);
}
```

Uppgift 3

a) (4 p)

```
public class Database {
    private File file;

    public Database(String fileName) throws IOException {
        file = new File(fileName);
    }

    public String lookup(String uid,String pwd) throws IOException {
        Scanner scn = new Scanner(file);
        while ( scn.hasNextLine() ) {
            String[] buf = (scn.nextLine()).split(":");
            if ( buf.length == 3 && buf[0].equals(uid) && buf[1].equals(pwd))
                return buf[2]; // License key found
        }
        return null; //FAIL
    }
}
```

b) (10 p)

```
public class Server {
    public static final String FAIL_MSG = "Authorization failed";
    private static final int BUF_SIZE = 1024;
    private Database database;
    private int port;

    public Server(int port,Database database) {
        this.port = port;
        this.database = database;
    }

    public void start() {
        try {
            DatagramSocket receiveSocket = new DatagramSocket(port);
            byte[] buf = new byte[BUF_SIZE];
            DatagramPacket receivePacket =
                new DatagramPacket(buf,buf.length);
            DatagramSocket replySocket = new DatagramSocket();
            while ( true ) {
                receiveSocket.receive(receivePacket);
                String uid = receivePacket.getAddress().getHostAddress();
                String pwd =
                   new String(receivePacket.getData(),
                              0,receivePacket.getLength());
                String result = database.lookup(uid,pwd);
                if ( result == null )
                    result = FAIL_MSG;

                replySocket.send(
                    new DatagramPacket(result.getBytes(),
                                       result.length(),
                                       receivePacket.getAddress(),
                                       receivePacket.getPort()));
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Uppgift 4

a) (4 p)

```
public class Fork {
    private boolean free = true;

    public synchronized void drop() {
        free = true;
        notify();
    }

    public synchronized void grab() {
        while ( ! free ) {
            try {
                wait();
            }
            catch ( InterruptedException e ) {
                return;
            }
        }
        free = false;
    }
}
```

b) (4 p)

```
public class Programmer extends Thread {
    private String name;
    private Fork left,right;
    private int noOfChunks;
    private static Random random = new Random();

    public Programmer(String name,Fork left,Fork right) {
        this.name = name;
        this.left = left;
        this.right = right;
        noOfChunks = 0;
    }

    public void run() {
        while ( ! interrupted() ) {
            try {
                sleep(10+random.nextInt(100));
            }
            catch ( InterruptedException e ) {
                break;
            }
            left.grab();
            right.grab();
            System.out.println(name + " eats (" + noOfChunks++ + ")");
            left.drop();
            right.drop();
        }
    }
}
```

c) (4 p)

```
public class Simulator {
    private static final int SIZE = 5;
    public Simulator() {
        Fork firstFork = new Fork();
        Fork left = firstFork;
        for ( int i = 0; i < SIZE; i++ )
            if ( i < SIZE - 1 ) {
                Fork right = new Fork();
                new Programmer("Programmer "+i,left,right).start();
                left = right;
            } else // Close the circle
                new Programmer("Programmer "+i,left,firstFork).start();
    }
}
```

Uppgift 5 (8 p)

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

    public ReadOnlyIterator(Iterator<E> it) {
        if ( it == null )
            throw new IllegalArgumentException(
                        "ReadOnlyIterator applied to null");
        this.it = it;
    }

    public boolean hasNext() { return it.hasNext(); }

    public E next() { return it.next(); }

    public void remove() {
        throw new UnsupportedOperationException(
                        "ReadOnlyIterator: remove not supported.");
    }
}
```