Lösningsförslag till tentamen*

Kursnamn	**Objektorienterade applikationer**
Tentamensdatum	**2017-06-08**
Program	**DAI 2**
Läsår	**2016/2017, lp 3**
Examinator	**Uno Holmer**

Uppgift 1 (8 p)

```
public class Clock extends Observable {
    private Time time;

    public void tick() {
        time++;
        setchanged();
        notifyObservers(time);
    }
}

public class Display implements Observer {
    private Clock clock;

    public Display( Clock clock ) {
        this.clock = clock;
        clock.addObserver( this );
    }

    private void show( Time time ) { ... }

    public void update( Observable obs, Object o ) {
        if ( obs == clock && o instanceof Integer )
            show( (Integer)o );
    }
}

public class Console implements Observer {
    private Clock clock;

    public Console( Clock clock ) {
        this.clock = clock;
        clock.addObserver( this );
    }

    private void print( int time ) {
        System.out.println(time);
    }

    public void update( Observable obs, Object o ) {
        if ( obs == clock && o instanceof Integer )
            print( (Integer)o );
    }
}
```

Uppgift 2 (4+4+8 p)

a) (4 p)

```
public class Konto {
    private long saldo = 0;

    public synchronized void transaktion(long belopp) {
        long nyttSaldo = saldo + belopp;
        saldo = nyttSaldo;
    }

    public long getSaldo() {
        return saldo;
    }
}
```

b) (4 p)

```
public class Bankomat extends Thread {
    private Konto konto;
    private ArrayList<Long> uttagssekvens;

    public Bankomat(Konto konto,ArrayList<Long> uttagssekvens) {
        this.konto = konto;
        this.uttagssekvens = uttagssekvens;
    }

    public void run() {
        for ( Long belopp : uttagssekvens )
            konto.transaktion(belopp);
    }
}
```

c) (8 p)

```
public class Simulator {
    private static int antalBankomater = 1000;
    private static int antalUttag = 10;
    private static long startSaldo = 0;

    private Konto konto;
    private Bankomat[] bankomater;

    public Simulator() {
        konto = new Konto();
        bankomater = new Bankomat[antalBankomater];
        for ( int i = 0; i < antalBankomater; i++ )
            bankomater[i] =
                new Bankomat(konto,slumpUttag(antalUttag));

        konto.transaktion(startSaldo);
    }
```

```
    public void simulera() {
        System.out.println("Startsaldo: " + konto.getSaldo());

        // starta bankomattrådarna
        for ( Bankomat b : bankomater )
            b.start();

        // invänta trådarna
        for ( Bankomat b : bankomater )
            try {
                b.join();
            }
            catch (Exception e) {}

        System.out.println("Saldo: " + konto.getSaldo());
    }

    private ArrayList<Long> slumpUttag(int antalUttag) {
        Random rand = new Random();
        ArrayList<Long> l = new ArrayList<Long>(antalUttag);
        for ( int i = 0; i < antalUttag; i++ ) {
            int belopp = 500 + rand.nextInt(500);
            startSaldo += belopp;
            l.add((long)-belopp);
        }
        return l;
    }
```

Uppgift 3 (10 p)

```
public class Application {
    public Application() {
        String cmd;
        while (true) {
            cmd = JOptionPane.showInputDialog(null,"Choose command");
            if (cmd == null)
                System.exit(0);
            processCommand(cmd);
        }
    }

    public void processCommand(String commandName) {;
        if (Constants.isCommand(commandName)) {
            // Capitalize the first letter
            commandName = Character.toUpperCase(commandName.charAt(0))
                          + commandName.substring(1).toLowerCase();
            Command command = getCommand(commandName);
            command.execute();
        } else
            JOptionPane.showMessageDialog(null,
                    commandName,
                    "Illegal command",
                    JOptionPane.ERROR_MESSAGE
            );
    }

    public static Command getCommand(String nameString) {
        Command command = null;
        try {
            // Load the class with the name given as a string
```

```
            Class cmdClass = Class.forName(nameString);
            // Create an "normal" object from the class object
            Object cmdObject = cmdClass.newInstance();
            // Cast it to the Command interface
            command = (Command)cmdObject;
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return command;
    }

    public static void main(String[] arg) {
        new Application();
    }
}
```

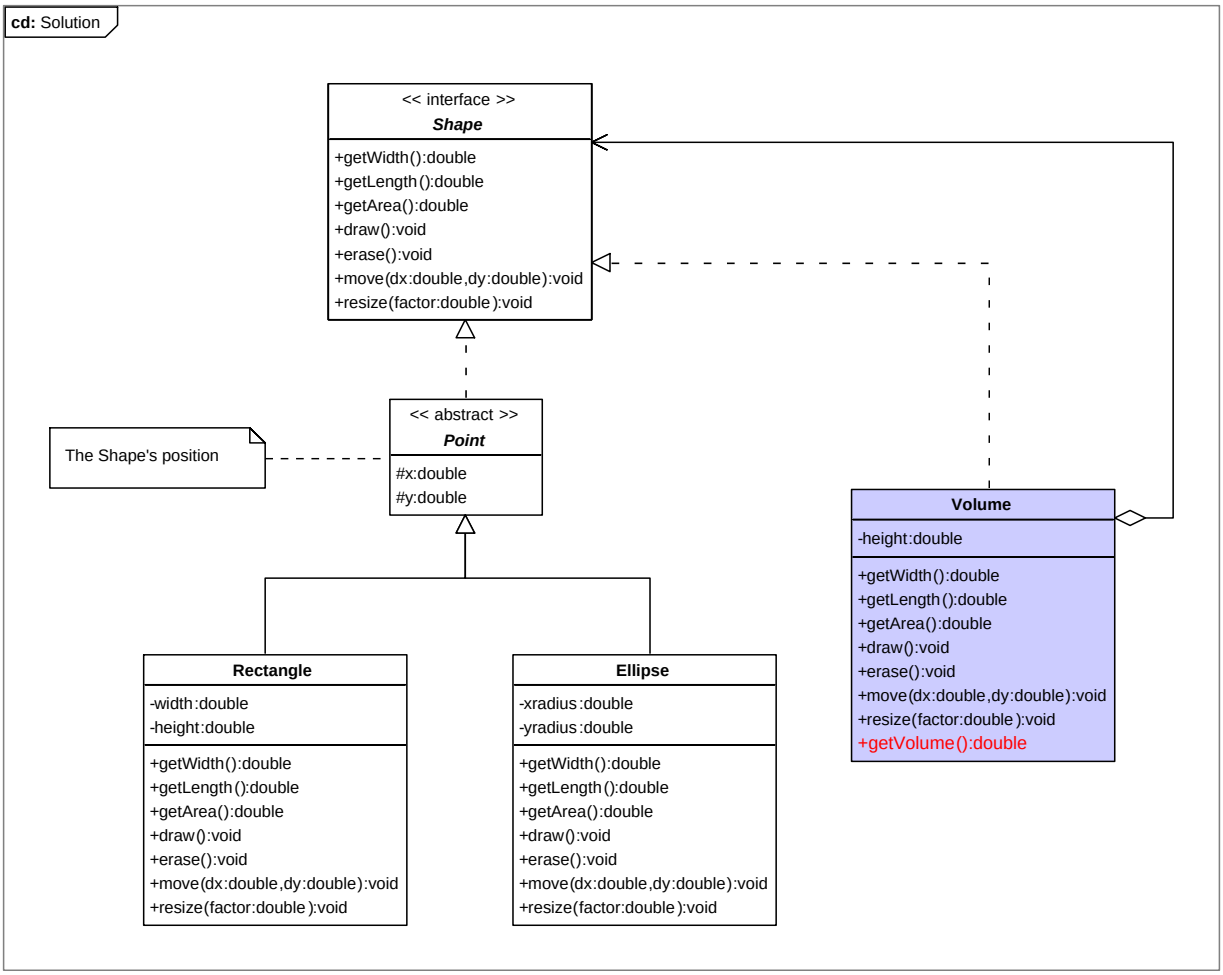
Uppgift 4 (16 p)

```
public class MessageSender {
    private String adress;
    private int port;
    private DatagramSocket socket;

    public MessageSender(String adress,int port)
           throws SocketException
    {
        this.adress = adress;
        this.port = port;
        socket = new DatagramSocket();
    }

    public void send(String text) {
        byte[] data = text.getBytes();
        InetAddress iaddr = null;
        try {
            iaddr = InetAddress.getByName( adress );
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }
        DatagramPacket packet =
            new DatagramPacket( data, data.length, iaddr, port );
        try {
            socket.send(packet);
        }
        catch (IOException e) {}
    }
}

public class MessageReceiver {
    private int port;
    private DatagramSocket socket;
    private JTextArea textDisplay = null;

    public MessageReceiver(int port) {
        this.port = port;
    }

    public void receiveMessages() throws SocketException {
        DatagramSocket socket = new DatagramSocket( port );
        byte[] data = new byte[ 1024 ];
        DatagramPacket packet =
               new DatagramPacket( data, data.length );
        while ( true ) {
            try {
                socket.receive( packet );
                String message =
                       new String( packet.getData(), 0,
                                   packet.getLength() );
                textDisplay.append( message + '\n' );
            }
            catch ( IOException ie ) {
                break;
            }
        }
    }
```

```
public void setTextDisplay(JTextArea textDisplay) {
        this.textDisplay = textDisplay;
    }
}
```

Uppgift 5 (3+6+1 p)

a) Observera att Volume bör implementera interfacet Shape, inte ärva från Point.



b)

```
public class Volume implements Shape
{
    private Shape obj; // The wrapped object
    private double height;

    public Volume(Shape obj,double height)
    {
        this.obj = obj;
        this.height = height;
    }

    // Delegate to the wrapped object
    public double getWidth() { return obj.getWidth(); }
    public double getLength() { return obj.getLength(); }
    public double getArea() { return obj.getArea(); }
    public void draw() { obj.draw(); }
    public void erase() { obj.draw(); }
    public void move(double dx,double dy) { obj.move(dx,dy); }
    public void resize(double factor) {
        height *= factor;
        obj.resize(factor);
    }

    // This is the actual decoration code
    public double getVolume() {
        return obj.getArea()*height;
    }
}
```

c) A cube with side length 10, with a corner positioned in origo:

```
Volume cube = new Volume(new Rectangle(0,0,10,10),10);
System.out.println(cube.getVolume());
```