Lösningsförslag till övning 6.

Uppgift 1

```
 import java.io.*;
public class SearcherThread extends Thread {
    private FileSearcher fs;
    private String pattern;
    public SearcherThread(FileSearcher fs, String pattern) {
        this.fs = fs;
        this.pattern = pattern;
    }
    public void run() {
        try {
            fs.search(pattern);
            if(fs.found())
                System.out.println(fs.getFileName());
        }
        catch (Exception e) {}
    }//run
}// SearcherThread

import java.io.*;
public class FileSearch {
    public static void main(String[]arg) throws Exception {
        String pattern = arg[0];
        for(int i = 1; i < arg.length; i++) {
            FileSearcher fs = new FileSearcher(arg[i]);
            SearcherThread searcher = new SearcherThread(fs, pattern);
            searcher.start();
        }
    }//main
}// FileSearch
```

Uppgift 2

```
public class Color {
    private int red;
    private int green;
    private int blue;

    public Color(int red, int green, int blue) {
        if (!isRBGColor(red, green, blue)) {
            throw new IllegalArgumentException();
        }
        this.red = red;
        this.green = green;
        this.blue = blue;
    }//konstruktor

    public synchronized void setColor(int red, int green, int blue) {
        if (!isRBGColor(red, green, blue)) {
            throw new IllegalArgumentException();
        }
        this.red = red;
        this.green = green;
        this.blue = blue;
    }setColor

    public getColor int[] getColor() {
        int[] retVal = new int[3];
        retVal[0] = red;
        retVal[1] = green;
        retVal[2] = blue;
        return retVal;
    }//getColor

    public synchronized void invert() {
        red = 255 - red;
        green = 255 - green;
        blue = 255 - blue;
    }//invert

    private static boolean isRBGColor(int red, int green, int blue) {
        return (0 <= red && red <= 255) && ( 0 <= green && green <= 255 )
                &&  ( 0<= blue && blue <= 255);
   }// isRBGColor
}//Color
```

Alternativ lösning: Synkronisering av satser I stället för av metoder.

```
public class Color {
    private int red;
    private int green;
    private int blue;
    private final Object lock = new Object();

    public Color(int red, int green, int blue) {
        if (!isRBGColor(red, green, blue)) {
            throw new IllegalArgumentException();
        }
        this.red = red;
        this.green = green;
        this.blue = blue;
    }//konstruktor

    public void setColor(int red, int green, int blue) {
        if (!isRBGColor(red, green, blue)) {
            throw new IllegalArgumentException();
        }
        synchronized(lock) {
            this.red = red;
            this.green = green;
            this.blue = blue;
        }
    }//setColor

    public int[] getColor() {
        int[] retVal = new int[3];
        synchronized(lock) {
            retVal[0] = red;
            retVal[1] = green;
            retVal[2] = blue;
        }
        return retVal;
    }//getColor

    public void invert() {
        synchronized(lock) {
            red = 255 - red;
            green = 255 - green;
            blue = 255 - blue;
        }
    }//invert

    private static boolean isRBGColor(int red, int green, int blue) {
        return (0 <= red && red <= 255) && ( 0 <= green && green <= 255 )
                &&  ( 0<= blue && blue <= 255);
    }// isRBGColor
}//Color
```

Uppgift 3

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class RotatingSquare extends JPanel  implements Runnable {
    private int vinkel= 0;
    private Thread activity = new Thread(this);
    private boolean rotateRight = true;

    public RotatingSquare() {
        activity.start();
    }//konstruktor

    public void paintComponent(Graphics pen) {
        super.paintComponent(pen);
        int x0 = getWidth()/2;
        int y0 = getHeight()/2;
        int radie = Math.min(getWidth(), getHeight())/4;
        Polygon p = new Polygon();
        for (int i = 0; i <= 4; i = i + 1) {
            int x1 = x0 + (int) (radie*Math.cos(Math.toRadians(vinkel + i*90)));
            int y1 = y0 + (int) (radie*Math.sin(Math.toRadians(vinkel + i*90)));
            p.addPoint(x1,y1);
        }
        pen.setColor(Color.blue);
        pen.fillPolygon(p);
    }//paintComponent

    public void run() {
        while(!Thread.interrupted()) {
            try {
                Thread.sleep(100);
            }
            catch (InterruptedException e) {
                break;
            }
            if (rotateRight)
                vinkel = (vinkel + 1) % 360;
            else
                vinkel = (vinkel - 1) % 360;
            repaint();
        }
    }//run

    public void turn(){
         rotateRight = !rotateRight;
    }//turn
}// RotatingSquare
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class RotateWindow extends JFrame implements ActionListener{
    private RotatingSquare square = new RotatingSquare();
    private JButton button = new JButton("TURN");

    public RotateWindow() {
        setLayout(new BorderLayout());
        button.addActionListener(this);
        add("Center", square);
        add("South", button);
        setSize(300, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }//konstruktor

    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == button)
            square.turn();
    }//actionPerformed

    public static void main(String[] args){
        new RotateWindow();
    }//main
}// RotateWindow
```

Uppgift 4

```
import java.util.*;
public class DishWasher extends Thread {
    private DryingFrame frame;
    private static Random rand = new Random();

    public DishWasher( DryingFrame frame) {
        this.frame = frame;
    }

    public void run() {
        for(int unit =1; unit <= 100; unit++){
            System.out.println("Diskar enhet " + unit);
            try {
                Thread.sleep(500 + rand.nextInt(500));
            }
            catch(Exception e){
                System.err.println("Diskaren har gått hem!");
            }
            frame.put(unit);
            System.out.println("Har diskat enhet " + unit);
        }
        frame.finished();
    }//run
}// DishWasher
```

```
import java.util.*;
public class DishDrier extends Thread {
    private DryingFrame frame;
    private static Random rand = new Random();

    public DishDrier( DryingFrame frame) {
        this.frame = frame;
    }

    public void run() {
        while (true) {
            int unit = frame.get();
            if (unit == -1)
                break;
            System.out.println("Torkar enhet " + unit);
            try {
                Thread.sleep(2000 + rand.nextInt(4000));
            }
            catch(Exception e){
                System.err.println("Diskaren har gått hem!");
            }
            System.out.println("Har torkat enhet " + unit);
        }
    }//run
}// DishDrier
```

```
public class DryingFrame  {
    private int[] queue;
    private int number;
    private boolean finish = false;

    public DryingFrame (int size) {
        queue = new int[size];
        number = 0;
    } //  DryingFrame

    public synchronized int get() {
        while (number == 0 && !finish) ) {
            try{
                wait();
            }
            catch(Exception e){}
        }
        if (number != 0) {
            int unit = queue[0];
            for(int i = 1; i < number; i++)
                queue[i-1] = queue[i];
            number--;
            notify();
            return unit;
        }
          return -1;
    }//get

    public synchronized void put(int unit) {
        while (number == queue .length) {
            try{
                wait();
            }
            catch(Exception e){}
        }
        queue[number] = unit;
        number++;
        notify();
    }//put

    public void finished() {
          finish = true;
     }// finished
}// DryingFrame


public class DishSimulation throws InterruptedException {
    public static void main(String[] args) {
        DryingFrame frame = new DryingFrame(4) ;
        DishWasher d = new DishWasher(frame);
        d.start();
        new DishDrier(frame).start();
        new DishDrier(frame).start();
    }//main
}// DishSimulation
```