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

Kursnamn	**Objektorienterade applikationer**
Provdatum	**2018-02-08**
Program	**DAI 2**
Läsår	**2017/2018, lp 3**
Examinator	**Uno Holmer**

Uppgift 1 (3+3 p)

a)

```
public class MySingletonClass {
    private Set<String> set;
    private static MySingletonClass instance = null;
    private MySingletonClass() {
        set = new HashSet<>();
    }
    public synchronized static MySingletonClass getInstance() {
        if ( instance == null )
            instance = new MySingletonClass();
        return instance;
    }
    public void add(String s) {
        set.add(s);
    }
    public boolean contains(String s) {
        return set.contains(s);
    }
}
```

b)

D1-K3, D2-K5, D3-K1

Uppgift 2 (5+1+1 p)

a)

```
public class Gui extends JFrame implements Observer {
    private JLabel stockLabel;
    private JLabel valueLabel;
    private Model model;

    public Gui(Model model) {
        this.model = model;
        makeFrame();
    }

    private void makeFrame() {
        setTitle("Stock Market");
        setLayout(new GridLayout(2,2));
        add(new JLabel("Stock"));
        stockLabel = new JLabel();
        add(stockLabel);
        add(new JLabel("Value"));
        valueLabel = new JLabel();
        add(valueLabel);
        makeMenubar();
        pack();
        setVisible(true);
    }

    private void makeMenubar() {
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);
        JMenu menu = new JMenu("Company");
        menuBar.add(menu);
        menu.add(makeMenuItem("Ericsson"));
        menu.add(makeMenuItem("Volvo"));
        menu.add(makeMenuItem("Chalmers AB"));
    }

    private JMenuItem makeMenuItem(String company) {
        JMenuItem item = new JMenuItem(company);
        item.addActionListener(
            e -> {  handleCompanySelection(company); });
        return item;
    }

    private void handleCompanySelection(String company) {
        String noOfStocks = askNoOfStocks(company);
        if ( noOfStocks == null )
            return;
        stockLabel.setText(noOfStocks + " " + company);
        try {
            model.computeStockValue(company,
                                    Integer.parseInt(noOfStocks));
        }
```

```
        catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                                          "IO Error",
                                          "Stock Market",
                                          JOptionPane.ERROR_MESSAGE);
        }
    }
    private String askNoOfStocks(String company) {
        return JOptionPane.showInputDialog(null,
                    "How many stocks?",
                    company,
                    JOptionPane.QUESTION_MESSAGE);
    }
    @Override
    public void update(Observable src,Object arg) {
        if ( src instanceof Model && arg instanceof Integer)
            valueLabel.setText(""+(Integer)arg);
    }
}
```

b)

```
public class Model extends Observable {
    private DataSource stockRates = new DatagramReceiver();

    public void computeStockValue(String company,int noOfStocks)
    throws IOException
    {
        setChanged();
        notifyObservers(stockRates.getValue(company)*noOfStocks);
    }
}
```

c)

```
public class Main {
    public static void main(String[] arg) {
        Model model = new Model();
        Gui gui = new Gui(model);
        model.addObserver(gui);
    }
}
```

Uppgift 3 (5 p)

```
public class DatagramReceiver implements DataSource {
    private static final String SERVER_ADDRESS = "127.0.0.1";
    private static final int SERVER_PORT = 1234;
    private int replyPort;

    @Override
    public int getValue(String key) throws IOException {
        sendRequest(key);
        return receiveResponse();
    }

    private void sendRequest(String key) throws IOException {
        try {
            InetAddress addr =
                InetAddress.getByName(SERVER_ADDRESS);
            DatagramSocket socket = new DatagramSocket();
            replyPort = socket.getLocalPort();
            String request = "REQUEST:" + key;
            byte[] data = request.getBytes();
            DatagramPacket pack =
                new DatagramPacket(data,data.length,addr,SERVER_PORT);
            socket.send(pack);
            socket.close();

        }
        catch (Exception e) {
            throw new IOException("Server communication failed");
        }
    }

    private int receiveResponse() throws IOException  {
        byte[] buf = new byte[1024];
        DatagramPacket pack = new DatagramPacket(buf,0,buf.length);
        DatagramSocket socket = new DatagramSocket(replyPort);
        socket.receive(pack);
        socket.close();
        String digits = new String(buf,0,pack.getLength());
        return Integer.parseInt(digits);
    }
}
```

Uppgift 4 (4+1+1 p)

a)

```
public class Register {
    private int value;
    private boolean isEmpty = true;

    public synchronized void load(int x) {
        while ( ! isEmpty ) {
            try {
                wait();
            }
            catch ( InterruptedException e) {
                e.printStackTrace();
                System.exit(0);
            }
        }
        value = x;
        isEmpty = false;
        notifyAll();
    }
    public synchronized int read() {
        while ( isEmpty ) {
            try {
                wait();
            }
            catch ( InterruptedException e) {
                e.printStackTrace();
                System.exit(0);
            }
        }
        isEmpty = true;
        notifyAll();
        return value;
    }
}
```

b)

```
public class Adder extends Thread {
    private Register leftOperand;
    private Register rightOperand;
    private Register output;

    public Adder(Register leftOperand,Register rightOperand,
                 Register output)
   {
        this.leftOperand = leftOperand;
        this.rightOperand = rightOperand;
        this.output = output;
    }

    public void run() {
        while ( ! interrupted() )
            output.load(leftOperand.read() + rightOperand.read());
    }
}
```

c)

```
public class Main {
    public static void main(String[] arg) {
        Register in1 = new Register();
        Register in2 = new Register();
        Register out = new Register();
        Adder adder = new Adder(in1,in2,out);
        Source s1 = new Source(in1);
        Source s2 = new Source(in2);
        adder.start();
        while ( true )
            System.out.println(out.read());
    }
}
```

Uppgift 5 (6 p)

```
public class ExtremeFinder {
    public static void main(String[] arg) throws IOException {
        try {
            findExtremeValues(arg[0],Integer.parseInt(arg[1]),
                                     Integer.parseInt(arg[2]));
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
    private static void findExtremeValues(String inputFile,
        int lo,int hi) throws IOException
    {
        DataInputStream in =
            new DataInputStream(new FileInputStream(inputFile));
        PrintWriter out =
            new PrintWriter(new FileWriter(inputFile+".txt"));
        List<Integer> highValues = new ArrayList<>();
        out.println("Temperatures below " + lo);
        while ( in.available() > 0 ) {
            int x = in.readInt();
            if ( x < lo )
                out.print(x + " ");
            if ( x > hi )
                highValues.add(x);
        }
        out.println();
        out.println("Temperatures above " + hi);
        for ( int x : highValues )
            out.print(x + " ");
        out.println();
        in.close();
        out.close();
    }
}
```