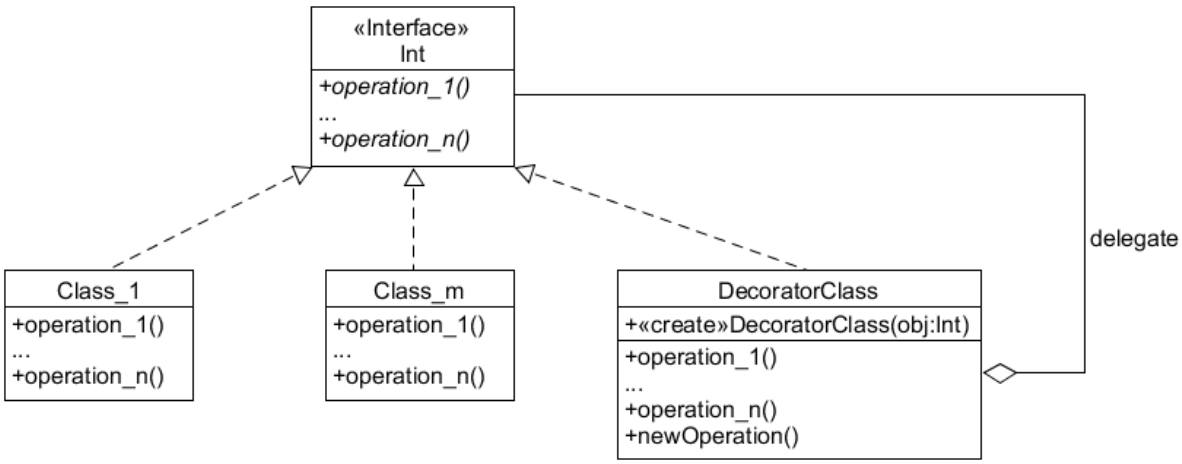
Lösningsförslag till dugga *P r e l i m i n ä r*

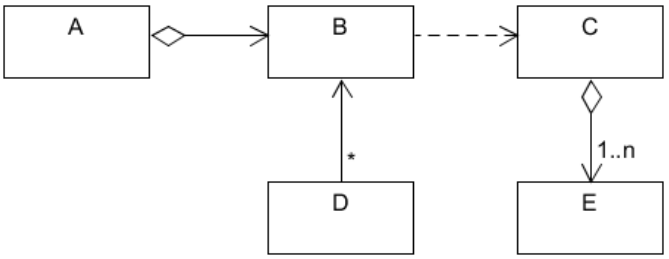
Kursnamn	**Objektorienterade applikationer**
Provdatum	**2017-02-09**
Program	**DAI 2**
Läsår	**2016/2017, lp 3**
Examinator	**Uno Holmer**

Uppgift 1 (1+1+1 p)

a)



b)



c)

```
public interface A;

public interface B extends A;

public interface C;

public class D implements B,C;

public interface E;

public class F extends D implements E;
```

Uppgift 2 (4 p)

```
public class WheatherDataLoader
extends Observable
implements DataLoader
{
    private String fileContents = null;

    @Override
    public void loadData(String station) throws IOException {
        StringBuilder buf = new StringBuilder();
        Scanner in = new Scanner(new FileReader(station + ".txt"));
        while ( in.hasNextLine() ) {
            buf.append(in.nextLine());
            buf.append("\n");
        }
        fileContents = buf.toString();
        setChanged();
        notifyObservers(fileContents);
    }

    @Override
    public String getData() {
        return fileContents;
    }
}
```

Uppgift 3 (4+1 p)

a)

```
public class WheatherGui extends JFrame implements Observer {
    private JTextArea textArea;
    private DataLoader dataLoader;

    public WheatherGui(DataLoader dataLoader) {
        this.dataLoader = dataLoader;
        makeFrame();
    }
    private void makeFrame() {
        setTitle("Wheather observations");
        makeMenubar();
        textArea = new JTextArea(10,30);
        textArea.setEditable(false);
        add(textArea);
        pack();
        setVisible(true);
    }
    private void makeMenubar() {
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);
        JMenu menu = new JMenu("Stations");
        menuBar.add(menu);
        menu.add(makeMenuItem("Gothenburg"));
        menu.add(makeMenuItem("Marstrand"));
        menu.add(makeMenuItem("Lysekil"));
    }
    private JMenuItem makeMenuItem(String station) {
        JMenuItem item = new JMenuItem(station);
        item.addActionListener((e) -> { loadData(station); });
        return item;
    }
    private void loadData(String station) {
        try {
           dataLoader.loadData(station);
        }
        catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                "Cannot find data for " + station);
        }
    }
    @Override
    public void update(Observable src,Object arg) {
        if ( src instanceof DataLoader && arg instanceof String ) {
            textArea.setText((String)arg);
        }
    }
// Alternativt:
    @Override
    public void update(Observable src,Object arg) {
        if ( src instanceof DataLoader )
            textArea.setText(((DataLoader)dataLoader).getData());
    }}
```

b)

```
public class Main {
    public static void main(String[] arg) {
        WheatherDataLoader dataLoader = new WheatherDataLoader();
        WheatherGui gui = new WheatherGui(dataLoader);
        dataLoader.addObserver(gui);
    }
}
```

Uppgift 4 (4+3+2+1 p)

a)

```
public class DataZipper {
    private String message = null;
    private boolean isEmpty = true;

    public synchronized void put(String message) {
        while ( ! isEmpty ) {
            try {
                wait();
            }
            catch ( InterruptedException e) {
                e.printStackTrace();
                System.exit(0);
            }
        }
        this.message = message;
        isEmpty = false;
        notifyAll();
    }

    public synchronized String take() {
        while ( isEmpty ) {
            try {
                wait();
            }
            catch ( InterruptedException e) {
                e.printStackTrace();
                System.exit(0);
            }
        }
        isEmpty = true;
        notifyAll();
        return message;
    }
}
```

b)

```
public class Writer extends Thread {
    private final int id;
    private int seqNo;
    private DataZipper dataZipper;

    public Writer(int id,DataZipper dataZipper) {
        this.id = id;
        this.dataZipper = dataZipper;
        seqNo = 0;
    }

    public void run() {
        while ( ! interrupted() ) {
            try {
                sleep(10);
            }
            catch ( InterruptedException e ) {
                e.printStackTrace();
                break;
            }
            dataZipper.put("Id: " + id + ", message: " + seqNo++);
        }
    }
}
```

c)

```
public class Reader extends Thread {
    private DataZipper dataZipper;

    public Reader(DataZipper dataZipper) {
        this.dataZipper = dataZipper;
    }

    public void run() {
        while ( ! interrupted() ) {
            System.out.println(dataZipper.take());
        }
        System.out.println("Reader interrupted");
    }
}
```

d)

```
public static void main(String[] arg) {
    int n = Integer.parseInt(arg[0]);
    DataZipper zip = new DataZipper();
    for ( int i = 1; i <= n; i++ )
        (new Writer(i,zip)).start();

    (new Reader(zip)).start();
}
```

Uppgift 5 (4 p)

```
public class DatagramSender {
     public static void main( String[] arg ) {
         try {
          Scanner kbd = new Scanner(System.in);
          InetAddress toAddr = InetAddress.getByName(arg[0]);
          int toPort = Integer.parseInt(arg[1]);
          DatagramSocket socket = new DatagramSocket();

          while ( kbd.hasNextLine() ) {
               String message = kbd.nextLine();
               if (message == null)
                    break;
               System.out.println(message);
               byte[] data = message.getBytes();
               socket.send(
                   new DatagramPacket(data, data.length,
                                      toAddr, toPort));
          }
        }
        catch (UnknownHostException uhe) {
            System.out.println("Unknown host: " + arg[0]);
        }
        catch (SocketException se) {
            System.out.println("Cannot create socket");
        }
        catch (IOException ie) {
            System.out.println("Datagram send failure");
        }
     }
}
```

Uppgift 6 (4 p)

```
public class AverageComputer {
    public static void main(String[] arg) {
        try {
            DataInputStream in =
                new DataInputStream(new FileInputStream(arg[0]));
            try {
                int count = 0;
                double sum = 0.0;
                while ( in.available() > 0 ) {
                    float x = in.readFloat();
                    if ( x > 0 ) {
                        sum += x;
                        count++;
                    }
                }
                if ( count > 0 )
                    System.out.println(sum/count);
                else
                    System.out.println("No numbers found");
            }
            finally { in.close(); }
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```