

---

## Lösningsförslag till tentamen

**Kursnamn**  
**Tentamensdatum**

**Objektorienterade applikationer**  
**2017-08-21**

**Program**  
**Läsår**  
**Examinator**

**DAI2**  
**2016/2017, lp 3**  
**Uno Holmer**

---

### Uppgift 1

a) (3 p)

```
public class A {  
    private B theB;  
    public A(B b) { theB = b; }  
}
```

```
public class B {  
    private C theC = new C();  
}
```

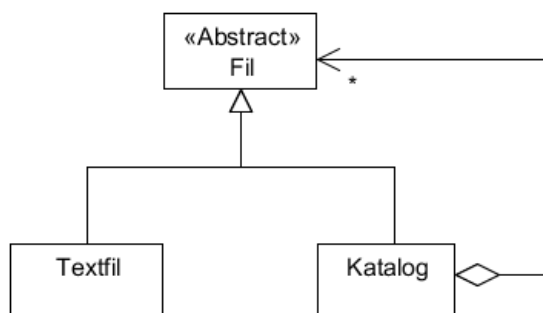
```
public class C {}
```

...

```
B b = new B();  
A a1 = new A(b);  
A a2 = new A(b);
```

b) (3 p)

Designmönstret kallas Composite.



## Uppgift 2 (8 p)

```
public class CounterIterator<E> implements Iterator<E> {
    private Iterator<E> it;
    private long count = 0;

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

    // Delegate
    @Override
    public boolean hasNext() { return it.hasNext(); }
    @Override
    public void remove() { it.remove(); }

    // Override and delegate
    @Override
    public E next() {
        count++;
        return it.next();
    }

    // Returns the number of elements returned by next() so far.
    public long getCount() { return count; }
}
```

## Uppgift 3

a) (5 p)

```
public class GameController {
    private GameLogic gameLogic;
    private Player player;
    private boolean gameOver;

    public GameController(GameLogic gameLogic, Player player) {
        this.gameLogic = gameLogic;
        this.player = player;
        gameOver = false;
    }
    public void nextClue() {
        if ( gameOver )
            return;
        if ( gameLogic.hasNextClue() )
            gameLogic.nextClue();
        else if ( gameLogic.hasNextQuestion() )
            gameLogic.nextQuestion();
        else
            gameOver = true;
    }
    public void userGuess(String guess) {
        if ( gameOver )
            return;
        player.addMarks(gameLogic.scoreUserGuess(guess));
        if ( gameLogic.hasNextQuestion() )
            gameLogic.nextQuestion();
        else
            gameOver = true;
    }
}
```

b) (5 p)

```
nextButton.addActionListener(  
    new ActionListener(){  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            controller.nextClue();  
        }  
    });  
  
guessField.addActionListener(  
    new ActionListener(){  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            controller.userGuess(guessField.getText());  
        }  
    });  
  
public void update(Observable o, Object arg) {  
    if ( o instanceof GameLogic && arg instanceof String ) {  
        String[] args = ((String)arg).split(":");  
        questionLabel.setText(args[0]);  
        clueArea.setText(args[1]);  
        levelLabel.setText(args[2]+ " poäng");  
        guessField.setText(null);  
    } else if ( o instanceof Player && arg instanceof Integer )  
        marksLabel.setText("Du har " + (int)arg + " poäng");  
    }  
}
```

c) (4 p)

```
public static void main(String[] arg) {  
    GameLogic logic = new GameLogic();  
    Player player = new Player();  
    GameController contr = new GameController(logic,player);  
    Gui gui = new Gui(contr);  
    logic.addObserver(gui);  
    player.addObserver(gui);  
    logic.reset();  
}
```

#### Uppgift 4

a) (4 p)

```
public class Server {  
    public Server(String port, String fileName) {  
        try {  
            ServerSocket servSock = new ServerSocket(Integer.parseInt(port));  
            while ( true ) {  
                Socket clientSock = servSock.accept();  
                System.out.println("Server: client connectcd");  
                new ClientHandler(clientSock, fileName);  
            }  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

b) (6 p)

```
public class ClientHandler extends Thread {
    private ObjectInputStream in;
    private ObjectOutputStream out;
    private String fileName;

    public ClientHandler(Socket sock,String fileName) {
        try {
            in = new ObjectInputStream(sock.getInputStream());
            out = new ObjectOutputStream(sock.getOutputStream());
            this.fileName = fileName;
            start();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        try {
            while ( ! interrupted() ) {
                Object o = in.readObject();
                if ( o instanceof String && ((String)o).equals("GETPOS") ) {
                    GPSCoordinate coord =
                        new GPSCoordinate(readGPSCoordinate());
                    out.writeObject(coord);
                }
            }
        }
        catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
            return;
        }
    }

    // Given metod
    private String readGPSCoordinate() { ... }
}
```

c) (7 p)

```
public class Client extends Thread {
    private ObjectInputStream in;
    private ObjectOutputStream out;

    public Client(String hostIP,String port) {
        try {
            InetAddress iAddr = InetAddress.getByName(hostIP);
            Socket sock = new Socket(iAddr,Integer.parseInt(port));
            out = new ObjectOutputStream(sock.getOutputStream());
            in = new ObjectInputStream(sock.getInputStream());
            start();
        }
        catch (IOException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }

    forts. ->
```

```
public void run() {
    while ( ! interrupted() ) {
        try {
            sleep(1000);
            out.writeObject("GETPOS");
            Object o = in.readObject();
            if ( o instanceof GPSCoordinate )
                System.out.println("Client: " + (GPSCoordinate)o);
        }
        catch (InterruptedException | IOException |
              ClassNotFoundException e)
        {
            return;
        }
    }
}
```

### Uppgift 5

a) (5 p)

```
public class FilterFactory {
    private HashMap<String,Filter> filterMap = new HashMap<>();

    public void addFilter(Filter f) {
        filterMap.put(f.getClass().getName(),f);
    }

    public Filter getFilter(String name) {
        return filterMap.get(name);
    }

    public Filter loadFilter(String name) {
        try {
            Filter f = (Filter)(Class.forName(name)).newInstance();
            addFilter(f);
            return f;
        } catch (Exception e) {
            return null;
        }
    }

    public Set<String> getKeys() {
        return filterMap.keySet();
    }
}
```

b) (10 p)

```
private JMenu makeFilterMenu(String menuName) {
    filterMenu = new JMenu(menuName);
    JMenuItem i = new JMenuItem("Other");
    i.addActionListener(
        new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e) {
                addOtherFilter(askForName());
            }
        });
    filterMenu.add(i);
    for ( String filterName : filterFactory.getKeys() ) {
        final String name = filterName;
        i = new JMenuItem(filterName);
        i.addActionListener(
            new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                    (filterFactory.getFilter(name)).apply(image);
                }
            });
        filterMenu.add(i);
    }
    return filterMenu;
}

private void addOtherFilter(String name) {
    Filter filter = filterFactory.getFilter(name);
    if ( filter == null ) {
        filter = filterFactory.loadFilter(name);
        if ( filter != null ) {
            final Filter f = filter;
            JMenuItem i = new JMenuItem(name);
            i.addActionListener(
                new ActionListener(){
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        f.apply(image);
                    }
                });
            filterMenu.add(i);
        } else
            JOptionPane.showMessageDialog(null,"No such filter","",
                JOptionPane.WARNING_MESSAGE);
    }
}

private String askForName() {
    return JOptionPane.showInputDialog(null,null,"Input filter name",
        JOptionPane.QUESTION_MESSAGE);
}
```