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

Kurs	**Objektorienterad programutveckling, fk**
Tentamensdatum	**2018-08-22**
Program	**DAI2**
Läsår	**2017/2018, lp 2**
Examinator	**Uno Holmer**

Uppgift 1 (7 p)

Abstrahera ut aktiviteten i ett gränssnitt `Trainee` och utnyttja polymorfism i klassen `PersonalTrainer`.

```
public interface Trainee {
    void train();
}
public class Walker implements Trainee {
    @Override
    public void train() {
        System.out.println("Walking");
    }
}
public class Runner implements Trainee {
    @Override
    public void train() {
        System.out.println("Running");
    }
public class Swimmer implements Trainee {
    @Override
    public void train() {
        System.out.println("Swimming");
    }
}
public class PersonalTrainer {
    private Trainee trainee;
    public PersonalTrainer(Trainee trainee) {
        this.trainee = trainee;
    }
    public void activate() {
        trainee.train();
    }
}
```

Uppgift 2 (8 p)

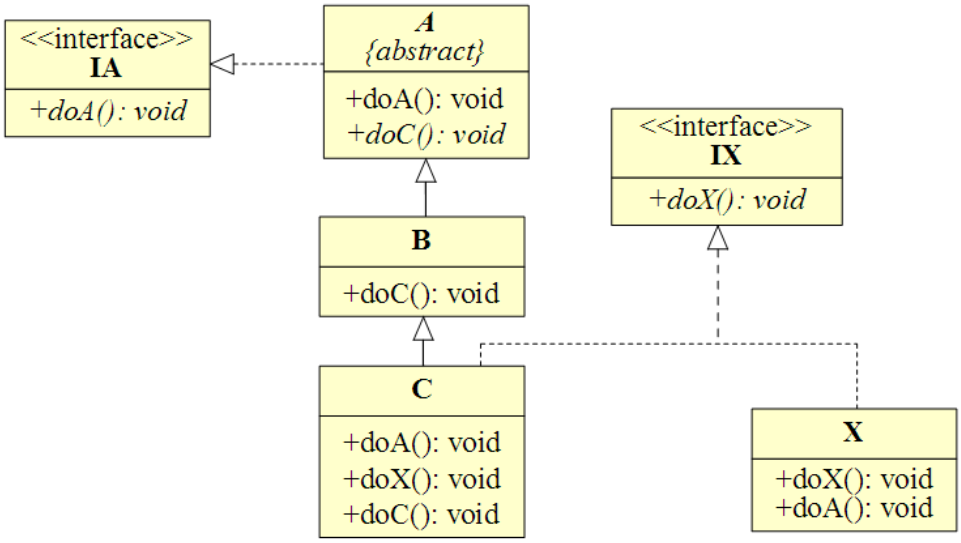
```
public abstract class AbstractAnimationDecorator implements Drawable {
    protected Drawable obj;

    public AbstractAnimationDecorator(Drawable obj) {
        this.obj = obj;
    }
    // Delegate
    public void move(int deltaX, int deltaY) {
        obj.move(deltaX,deltaY);
    }
    public void moveTo(int x, int y) {
        obj.moveTo(x,y);
    }
    public void draw() {
        obj.draw();
    }
    public void erase() {
        obj.erase();
    }
}

public class Animation extends AbstractAnimationDecorator {
    public Animation(Drawable d) {
        super(d);
    }
    // Decorate
    public void animate(int dx,int dy,int steps,int speed) {
        draw();
        for ( int i = 0; i < steps; i++ ) {
            try {
                Thread.sleep(1000/speed);
            }
            catch (Exception e) {}
            erase();
            move(dx,dy);
            draw();
        }
    }
}
```

Uppgift 3 (1+8 p)

a)



b)

i. Skriver ut: A doA

ii. Satsen IA a = **new** X(); ger ett kompileringsfel, eftersom X inte är en subtyp till IA.

iii. Satsen C c = **new** B(); ger ett kompileringsfel, eftersom B inte är en subtyp till C

iv. Satsen b.doX(); ger ett kompileringsfel, eftersom den statiska typen B inte har metoden doX().

v. Ger ett exekveringsfel, eftersom C och X är inkompatibla typer

vi. Skriver ut: C doC

vii. Skriver ut: A doA

viii. Satsen IA a = **new** A(); ger ett kompileringsfel, eftersom klassen A är abstrakt.

Uppgift 4 (1+6 p)

a) Se uppgift b.

b)

```
public static Map<String,List<Double>> loadCSVData(String csvFile)
    throws IOException,NumberFormatException
{
    Scanner sc = new Scanner(new File(csvFile));
    Map<String,List<Double>> map = new TreeMap<>();
    while ( sc.hasNextLine() ) {
        String line = sc.nextLine();
        String[] fields = line.split(":");
        String key = fields[0];
        List<Double> values = map.get(key);
        if ( values == null ) {
            values = new ArrayList<>();
            map.put(key,values);
        }
        for ( int i = 1; i < fields.length; i++ )
            values.add(Double.parseDouble(fields[i]));
    }
    sc.close();
    return map;
}
```

Uppgift 5 (4 p)

a) Specifikation I är starkast, eftersom eftervillkoret är starkare.

b) Specifikation III är starkast, eftersom förvillkoret är svagare.

c) Specifikation III är starkast, eftersom förvillkoret är svagare och eftervillkoret är starkare.

d) Går inte att avgöra. Specifikation IV har både starkare förvillkor och eftervillkor.

Uppgift 6 (9 p)

Vi antar att kortleken lagras i instansvariabeln `cards` av typen `ArrayList<Card>`.

```
@Override
public CardHand clone() {
    CardHand result = null;
    try {
        result = (CardHand)super.clone();
        result.cards = (ArrayList<Card>)cards.clone();
        for ( int i = 0;i < cards.size();i++ )
            result.cards.set(i,cards.get(i).clone());
        return result;
    }
    catch (CloneNotSupportedException e) {
        throw new InternalError();
    }
}

@Override
@SuppressWarnings("unchecked")
public final boolean equals(Object o) {
    if ( this == o )
        return true;
    else if ( o instanceof CardHand ){
        CardHand other = (CardHand)o;
        ArrayList<Card> left = (ArrayList<Card>)cards.clone();
        ArrayList<Card> right = (ArrayList<Card>)other.cards.clone();
        Collections.sort(left);
        Collections.sort(right);
        return left.equals(right);
    }
    return false;
}

@Override
public int hashCode() {
    ArrayList<Card> copy = (ArrayList<Card>)cards.clone();
    Collections.sort(copy);
    return copy.hashCode();
}
```

Uppgift 7 (3+5 p)

a)
Designen bryter mot principerna:
Tell don't ask: `ClimateControl` manilulerar det interna tillståndet i `AirCondition`. *Law of Demeter - don't talk to strangers*: `ClimateControl` opererar på `Cooler` med `AirCondition` som ombud. *Information expert*: Uppgifter som borde ligga i `AirCondition` utförs i `ClimateControl`.

b)
```
public class AirCondition {
    public static final double COMFORT_TEMPERATURE = 22.0;
    private double temperature;
    private Cooler cooler = new Cooler();

    public double getTemperature() { return temperature; }

    public void adjustTemperature() {
        double delta = temperature - COMFORT_TEMPERATURE;
        if ( delta > 1.0 )
            decreaseTemperature();
    }
    private void decreaseTemperature() {
        cooler.on();
        while ( temperature > COMFORT_TEMPERATURE ) {
            //sleep(100); // ms
        }
        cooler.off();
    }
    // Other members omitted.
}

public class ClimateControl {
    private AirCondition airCondition;
    public ClimateControl(AirCondition airCondition) {
        this.airCondition = airCondition;
    }
    public void adjustTemperature() {
        airCondition.adjustTemperature();
    }
    // Other members omitted.
}
```

Uppgift 8 (3+5 p)

a)

```
public class BoxView extends JButton implements Observer {
    public BoxView(String text) {
        super(text);
    }
    public void update(Observable obj,Object o) {
        String argString = (String)o;
        String[] args = argString.split(";");
        String state = args[0];
        boolean isFull = Boolean.parseBoolean(args[1]);
        if ( state.equals("SELECTED") )
            setText("SELECTED");
        else if ( state.equals("OPEN") ) {
            if ( isFull )
                setText("WIN!");
            else
                setText("- EMPTY -");
        } else
             setText("");
    }
}
```

b)

```
public class Gui extends JFrame {
    private GameEngine engine;

    public Gui() {
        engine = new GameEngine();
        makeFrame();
    }
    private void makeFrame() {
        setTitle("Game");
        setLayout(new GridLayout(2,3));

        for ( int i = 0; i < 3; i++ ) {
            BoxView bv = new BoxView("                    ");
            final int i2 = i;
            bv.addActionListener(
                new ActionListener(){
                    public void actionPerformed(ActionEvent e) {
                        engine.select(i2);
                    }
            });
            add(bv);
            engine.addBoxObserver(i,bv);
        }
        add(new JLabel());  // Dummy
        JButton newGameButton = new JButton("New Game");
        newGameButton.addActionListener(
            new ActionListener(){
                public void actionPerformed(ActionEvent e) {
                    engine.newGame();
                }
        });
        add(newGameButton);
        add(new JLabel());  // Dummy
        pack();
        setVisible(true);
    }
}
```