Lösningar till tentamen i EDAF25

21 aug 2017

Lösning 1

Javaklasser (många varianter finns naturligtvis):

```
class Client {
    private Invoker invoker;

    public void newCommand(String cmdText) {
        Command cmd;
        if (cmdText.equals"("X))
            cmd = new XCommand();
        else if (cmdText.equals"("Y))
            cmd = new YCommand();
        else
            cmd = new ZCommand();
        invoker.store(cmd);
        cmd.execute();
    }
}

public interface Command {
    public abstract void execute();
}

class XCommand implements Command {
    public void execute() { ... }
}

class YCommand implements Command {
    public void execute() { ... }
}

class ZCommand implements Command {
    public void execute() { ... }
}

class Invoker {
    private ArrayList<Command> myCommand = new ArrayList<Command>();

    public void store(Command cmd) {
        myCommand.add(cmd);
    }

    public void doCommand(int nbr) {
        myCommand.get(myCommand.size()-nbr).execute();
    }
}
```

Lösning 2

(a) Javakod för en MVC modellering mha observer-ramverket

```
public class Model extends Observable {
    private boolean on = false;

    public void toggle() {
        on = !on;
        setChanged();
        notifyObservers();
    }

    public boolean isOn() {
        return on;
    }
}

public class Control implements ActionListener {
    private Model switch_;

    public Control(Model switch_, View view) {
        this.switch_ = switch_;
        view.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        switch_.toggle();
    }
}

public class View extends JButton implements Observer {
    private Model switch_;

    public View(Model switch_) {
        super("OFF");
        this.switch_ = switch_;
        switch_.addObserver(this);
    }

    public void update(Observable o, Object arg) {
        setLabel(switch_.isOn() ? "ON" : "OFF");
    }
}
```

(b) Klassdiagram i figur 1

Lösning 3

```
class NullSoundCard extends SoundCard {
    String getVersion() {
        return "Sound card not available!";
    }
    void playFile(String fileName) {
        // Do nothing
    }
}
...
SoundCard card = Hardware.getSoundCard();
if (card == null) {
    card = new NullSoundCard();
}
...
```

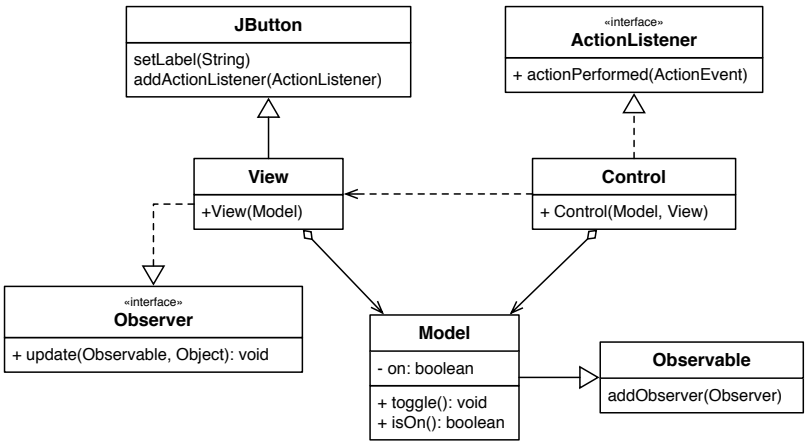


Figure 1: MVC

```
System.out.println(card.getVersion());
...
card.playFile("mysoundfile.snd");
```

Lösning 4

(a) Det maximala flödet i nätverket är 11

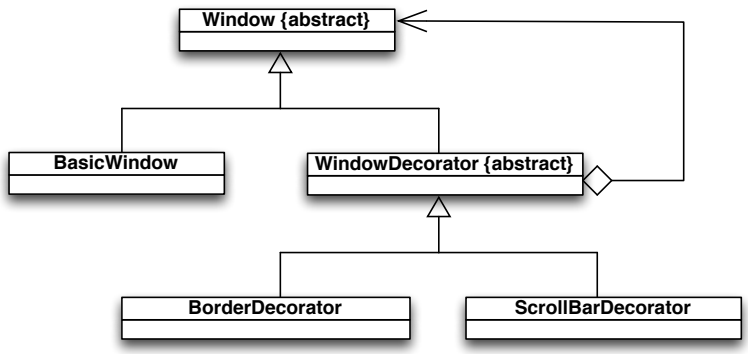
(b) och kan beräknas med hjälp av Ford-Fulkersons algoritm: 1) sök en utökande väg 2) öka flödet längs den vägen och beräkna residualgrafen 3) upprepa tills det inte längre finns någon utökande väg.

(c) Matrisrepresentation av grafen

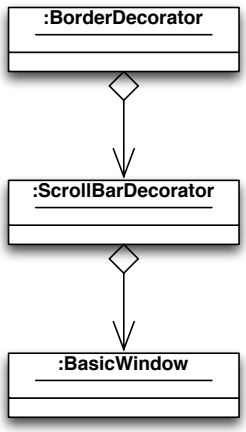
	Start	A	B	C	D	Slut
Start	0	7	0	4	0	0
A	0	0	9	0	6	0
B	0	0	0	0	3	4
C	0	3	2	0	0	0
D	0	0	0	0	0	8
Slut	0	0	0	0	0	0

Lösning 5

(a) Klassdiagram:



(b) Objektdiagram:



Lösning 6

T1 D, Användaren bestämmer själv när en ny utgåva av ett paket ska integreras men användaren ska inte behöva titta på implementeringen eller göra ändringar i paketet. Om användaren behöver ta ansvar för detta betraktas inte paketet som återanvändningsbart.

T2 E, Att tillämpa Mallmetod-mönstret är tvärtom ett sätt att eliminera duplicerad kod genom att det som är gemensamt för subklasserna läggs i mallmetoder i basklassen och subklasserna endast implementerar det som skiljer dem åt.

T3 C, Fabriksmetod-mönstret innebär att man använder arv för skapandet av objekt (påminner om mallmetod-mönstret) och hjälper i det här fallet eftersom användaren då kan implementera sin egen editor-variant med en egen implementering av fabriksmetoden create som känner till och kan instansiera de olika Shape-objekten som användaren väljer. En fabriksmetod kan men behöver inte finnas i en abstrakt fabriksklass. I det här fallet skulle det bara flytta problemet eftersom ramverket fortfarande behöver en instans av fabriken (och därmed känna till vilka typer av objekt som ska skapas) för att kunna delegera till denna.

T4 D, Eftersom Figure är en lista av Object är det möjligt att lägga till vilken typ av komponenter som helst till en figur. Dock kommer inte metoden draw att rita ut något annat än cirklar och kvadrater om den inte uppdateras.

T5 B eller D, SRP innebär att en klass bara får ha en ”anledning att förändras” vilket i sin tur beror på kontexten (hur och var klassen används). En nyckel för att kunna avgöra om en klass följer SRP är att dess ansvar beskrivs (genom dess namn och dokumentation). Namnet i exemplet antyder att klassens ansvar är att hålla reda på en figurs uppbyggnad och en anledning till förändring kan t.ex. vara att man vill ändra konceptet figur (t.ex. precisera vilka typer av komponenter som får lov att ingå i en figur). Metoden draw skulle dock kunna behöva ändras av andra orsaker (som att man vill ändra teknik för själva bildskapandet). SRP kan alltså innebära att ansvaret för att rita dessa komponenter bör delegeras till någon annan klass. Vi har dock för lite information för att kunna avgöra det. Object är en superklass med flera ansvar som t.ex. att leverera en strängrepresentation av sig själv eller en hashkod. Figure skulle kunna ha ett tydligt ansvar (och följa SRP) även om dess komponenter inte har det.

T6 A

T7 B Macro visar exempel på både kommando- och kompositmönstret. Kommandomönstret handlar om att kapsla in ett anrop av ett kommando medan kompositmönstret i det här fallet handlar om att strukturera dessa anrop hierarkiskt.

T8 A

T9 D, LSP säger att metoder som använder referenser till basklasser måste kunna använda instanser av dess sub-klasser utan att känna till det. Således kan arv/generalisering om man inte är försiktig leda till brott mot LSP. Samtidigt är arv/generalisering ett verktyg för att undvika brott mot andra viktiga principer som t.ex. OCP och DRY. LSP säger således snarare något om *hur* arv ska tillämpas än att det ska undvikas.

T10 A, Tillståndsmönstret används precis som strategimönstret för att kapsla in varianter av beteenden för ett objekt och växla mellan dessa via metodanrop specificerade i ett interface och implenterade i olika subklasser. I tillståndsmönstret definieras dessa varianter utifrån objektets interna tillstånd.