**Arhitectura (Design Pattern) Model View Controller**

În arhitectura Model View Controller (MVC) aplicaţia propriu-zisă (obiectul { Model}) este proiectată independent de contextul în care va fi utilizată. Prin urmare, obiectul model poate fi asociat ulterior cu interfeţe utilzator (obiecte { View}) de cele mai diverse tipuri. Pentru a realize interactţiunea dintre elementele de interfaţa şi aplicaţie sunt utilizate obiecte de control ({ Controller}).

Exemplu In această foarte simplificată ilustrare a elementelor de arhitectură MVC, aplicaţia este alcătuită dintr-un obiect m din clasa Model, asupra caruia se poate acţiona pentru a mări cu o unitate atributul m.x (prin metoda increment()) sau pentru a inspecta valoarea acestui atribut (metoda get\_x()).

Interfaţa pe care o asociem modelului este de tip graphic şi este obiect v al clasei View. Elementul de interacţiune pe care îl avem în vedere în acest exemplu este obiectul buton v.b al clasei Button. La fiecare acţionare a sa este incrementată valoarea atributului m.x iar noua valoare este afişată în campul v.tf.

Interceptarea şi prelucrarea evenimentelor generate prin acţionarea butonului b sunt realizate prin intermediul unui obiect de control c al clasei Controller (această clasă implementează metoda void actionPerformed(ActionEvent) a interfaţei ActionListener).

Relaţia dintre clasele din acest exemplu este prezentată în Figura MVCClase. Colaborarea dintre obiectul model, obiectele de interfaţă şi obiectul de control este ilustrată în Figura MVCColaborare. Prin mesajul b.addActionListener(c) trimis către obiectul buton b se face operaţia de { instalare} a evenimentelor. După instalare, acţionarea butonului b este { notificată} prin mesajul actionPerformed(…) trimis obiectului de control. Ca răspuns, obiectul de control trimite mesaje către modelul m pentru modificarea şi afişarea atributului m.x.



Figura MVCClase. Relaţia dintre clase

2.1.3: setText(…)

2.1.2: get\_x()

2.1.1: actiune()

2.1: actionPerformed(actionEvent)

m: Model

b: Button

c: Controller

1: addActionListener(c)

2: click

user

Java Virtual Machine

tf: TextField

1. instalare

3. notificare

2. actiune

4. prelucrare

Figura MVCColaborare. Diagrama de colaborare între obiectele MVC

// Model View Controller

import java.awt.\*;

import java.awt.event.\*;

class Model{

private int x=0;

public Model(){};

public void increment(){x++;}

public int get\_x(){return x;}

}

public class View extends Frame{

private Button b;

protected Model m;

private Controller c;

protected TextField tf;

public static void main(String args[]){

Frame v= new View();

}

public View(){

setTitle("Exemplu Model-View-Controller");

b= new Button("Actiune");

add("North",b);

m=new Model();

c=new Controller(this);

b.addActionListener(c);

tf=new TextField(10);

add("Center",tf);

setSize(100,250);

setVisible(true);

}

}

class Controller implements ActionListener{

private View vw;

public Controller(View v){

vw=v;

}

public void actionPerformed(ActionEvent e){

vw.m.increment();

vw.tf.setText(String.valueOf(vw.m.get\_x()));

}

}