Chapter 9

The MVC Design Pattern

```
DEFINITION PboxStackObj;
    CONST
        capacity = 8;
    TYPE
        Stack = RECORD
            (VAR s: Stack) Clear, NEW;
            (IN s: Stack) NumItems (): INTEGER, NEW;
            (VAR s: Stack) Pop (OUT val: REAL), NEW;
            (VAR s: Stack) Push (val: REAL), NEW
        END;
END PboxStackObj.

DEFINITION PboxStackADT;
    CONST
        capacity = 8;
    TYPE
        Stack = RECORD  END;
    PROCEDURE Clear (VAR s: Stack);
    PROCEDURE NumItems (IN s: Stack): INTEGER;
    PROCEDURE Pop (VAR s: Stack; OUT val: REAL);
    PROCEDURE Push (VAR s: Stack; val: REAL);
END PboxStackADT.
```

Figure 9.1
The interface for PboxStackObj and PboxStackADT for comparison.

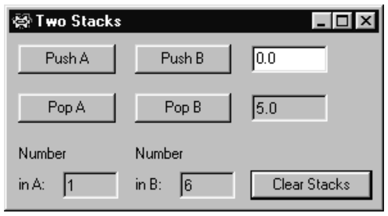


Figure 9.2
The dialog box for manipulating two stacks. It is implemented with PboxStackObj.

```
MODULE Hw99Pr0980;
    IMPORT Dialog, PboxStackObj;

    VAR
        d*: RECORD
            valuePushed*, valuePopped-: REAL;
            numItemsA-, numItemsB-: INTEGER;
        END;
        stackA, stackB: PboxStackObj.Stack;
```

Figure 9.3
A program that uses a stack class to implement the dialog box of Figure 9.2.

```
PROCEDURE PushA*;
BEGIN
    stackA.Push(d.valuePushed);
    d.numItemsA := stackA.NumItems();
    Dialog.Update(d)
END PushA;

PROCEDURE PushB*;
BEGIN
    stackB.Push(d.valuePushed);
    d.numItemsB := stackB.NumItems();
    Dialog.Update(d)
END PushB;
```

```
PROCEDURE **PopA***;
BEGIN
    stackA.Pop(d.valuePopped);
    d.numItemsA := stackA.NumItems();
    Dialog.Update(d)
END PopA;

PROCEDURE **PopB***;
BEGIN
    stackB.Pop(d.valuePopped);
    d.numItemsB := stackB.NumItems();
    Dialog.Update(d)
END PopB;
```

```
    PROCEDURE ClearStacks*;
    BEGIN
        stackA.Clear;
        stackB.Clear;
        d.valuePushed := 0.0; d.valuePopped := 0.0;
        d.numItemsA := 0; d.numItemsB := 0;
        Dialog.Update(d)
    END ClearStacks;

BEGIN
    ClearStacks
END Hw99Pr0980.
```

Procedure-oriented	**Object-oriented**
type	class
procedure	method
variable	object

Figure 9.4
Object-oriented terminology.

The MVC Design Pattern

- M model
- V view
- C controller

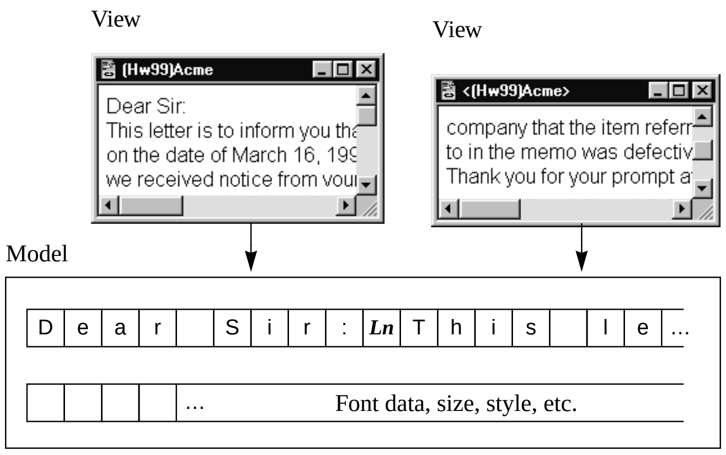

View
(Hw99)Acme
Dear Sir:
This letter is to inform you tha
on the date of March 16, 199
we received notice from your
View
<(Hw99)Acme>
company that the item referr
to in the memo was defectiv
Thank you for your prompt a
Model
D e a r S i r : *Ln* T h i s l e …
…
Font data, size, style, etc.

Figure 9.5
The relationship between a view and its model.

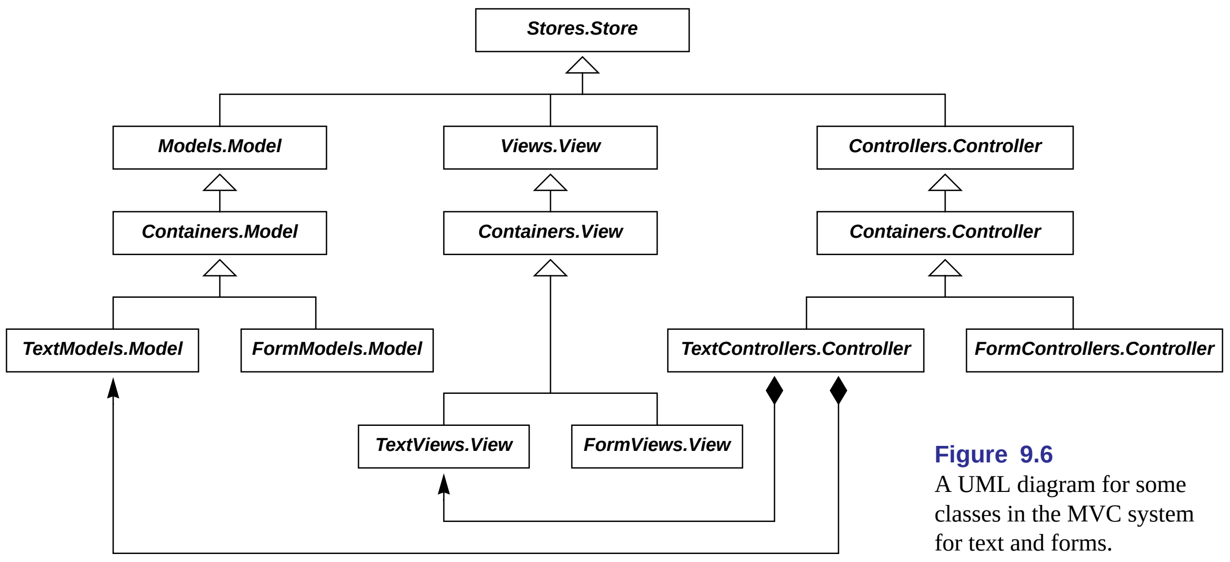


Figure 9.6
A UML diagram for some classes in the MVC system for text and forms.

```
DEFINITION PboxMappers;

    IMPORT TextModels;

    TYPE
        Formatter = EXTENSIBLE RECORD
            (VAR f: Formatter) ConnectTo (text: TextModels.Model), NEW;
            (VAR f: Formatter) WriteInt (n, minWidth: INTEGER), NEW;
            (VAR f: Formatter) WriteReal (x: REAL; minWidth, dec: INTEGER), NEW;
            (VAR f: Formatter) WriteChar (ch: CHAR), NEW;
            (VAR f: Formatter) WriteString (str: ARRAY OF CHAR), NEW
            (VAR f: Formatter) WriteLn, NEW;
            (VAR f: Formatter) WriteIntVector (IN v: ARRAY OF INTEGER; numItm, minWidth: INTEGER), NEW;
            (VAR f: Formatter) WriteRealVector (IN v: ARRAY OF REAL; numItm, minWidth, dec: INTEGER), NEW;
            (VAR f: Formatter) WriteIntMatrix (IN mat: ARRAY OF ARRAY OF INTEGER;
                numR, numC, minWidth: INTEGER), NEW;
            (VAR f: Formatter) WriteRealMatrix (IN mat: ARRAY OF ARRAY OF REAL;
                numR, numC, minWidth, dec: INTEGER), NEW;
        END;
```

Figure 9.7
The interface for PboxMappers.

```
    Scanner = EXTENSIBLE RECORD
        eot-: BOOLEAN;
        (VAR s: Scanner) ConnectTo (text: TextModels.Model), NEW;
        (VAR s: Scanner) Pos (): INTEGER, NEW;
        (VAR s: Scanner) ScanInt (OUT n: INTEGER), NEW;
        (VAR s: Scanner) ScanReal (OUT x: REAL), NEW;
        (VAR s: Scanner) ScanChar (OUT ch: CHAR), NEW;
        (VAR s: Scanner) ScanPrevChar (OUT ch: CHAR), NEW;
        (VAR s: Scanner) ScanString (OUT str: ARRAY OF CHAR), NEW
        (VAR s: Scanner) ScanIntVector (OUT v: ARRAY OF INTEGER; OUT numItm: INTEGER), NEW;
        (VAR s: Scanner) ScanRealVector (OUT v: ARRAY OF REAL; OUT numItm: INTEGER), NEW;
        (VAR s: Scanner) ScanIntMatrix (OUT mat: ARRAY OF ARRAY OF INTEGER;
            OUT numR, numC: INTEGER), NEW;
        (VAR s: Scanner) ScanRealMatrix (OUT mat: ARRAY OF ARRAY OF REAL;
            OUT numR, numC: INTEGER), NEW;
    END;

END PboxMappers.
```

```
DEFINITION TextModels;
    TYPE
        Directory = POINTER TO ABSTRACT RECORD
            (d: Directory) New (): Model, NEW, ABSTRACT;
        END;
        Model = POINTER TO ABSTRACT RECORD (Containers.Model);
    VAR
        dir-: Directory;
END TextModels.
```

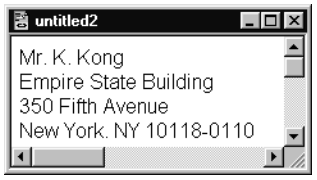
Figure 9.8
The interface for TextModels. Many items from the interface are omitted from this listing.

```
DEFINITION TextViews;
    TYPE
        Directory = POINTER TO ABSTRACT RECORD
            (d: Directory) New (text: TextModels.Model): View, NEW, ABSTRACT;
        END;
        View = POINTER TO ABSTRACT RECORD (Containers.View)
    VAR
        dir-: Directory;
END TextViews.
```

Figure 9.9
The interface for TextViews. Many items from the interface are omitted from this listing.

```
DEFINITION Views;
    TYPE
        View = POINTER TO ABSTRACT RECORD (Stores.Store)
        END;
    PROCEDURE OpenView (view: View);
END Views.
```

Figure 9.10
The interface for Views. Many items from the interface are omitted from this listing.



```
Mr. K. Kong
Empire State Building
350 Fifth Avenue
New York. NY 10118-0110
```

Figure 9.11
The output for the program in Figure 9.12

```
MODULE Hw99Pr0981;
    IMPORT TextModels, TextViews, Views, PboxMappers;

    PROCEDURE PrintAddress*;
        VAR
            md: TextModels.Model;
            vw: TextViews.View;
            fm: PboxMappers.Formatter;
    BEGIN
        md := TextModels.dir.New();
        fm.ConnectTo(md);
        fm.WriteString("Mr. K. Kong"); fm.WriteLn;
        fm.WriteString("Empire State Building"); fm.WriteLn;
        fm.WriteString("350 Fifth Avenue"); fm.WriteLn;
        fm.WriteString("New York, NY 10118-0110"); fm.WriteLn;
        vw := TextViews.dir.New(md);
        Views.OpenView(vw)
    END PrintAddress;

END Hw99Pr0981.
```

Figure 9.12
A program that creates a text model and displays it in a text view.

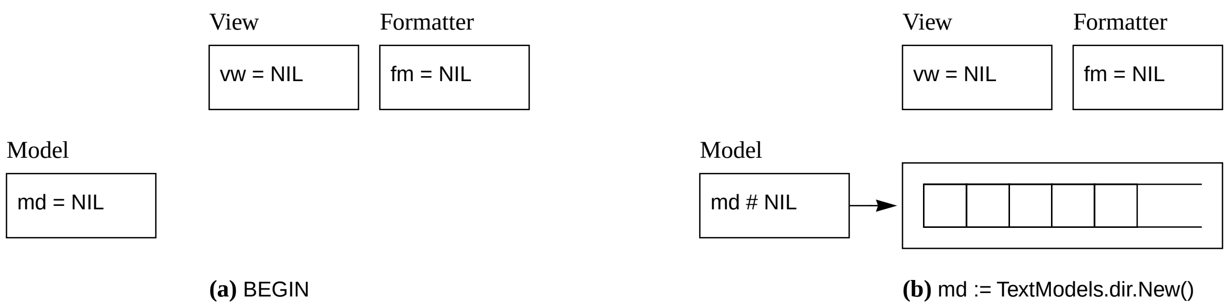
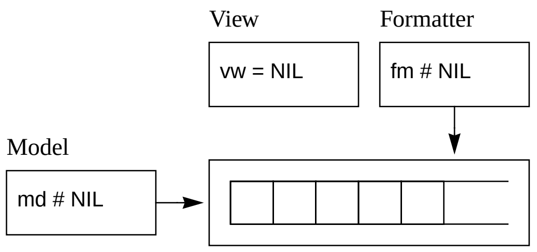
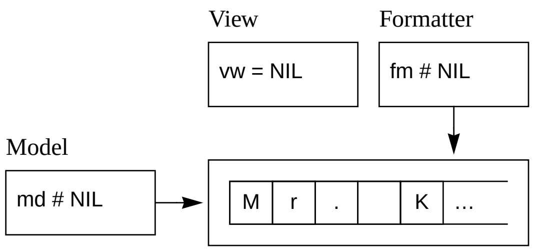


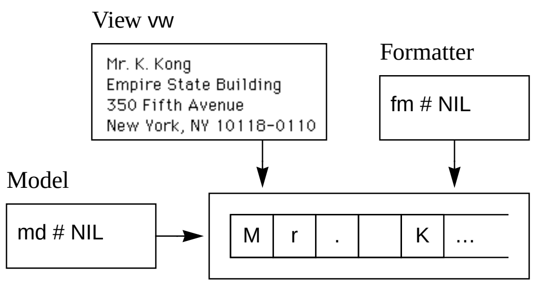
Figure 9.13
The effect of the MVC statements in Figure 9.12



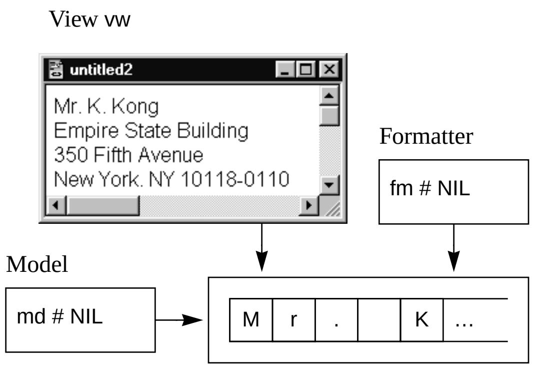
(c) fm.ConnectTo(md)



(d) fm.WriteString("Mr. K. Kong")



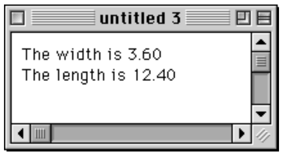
(e) vw := TextViews.dir.New(md)



(f) Views.OpenView(vw)

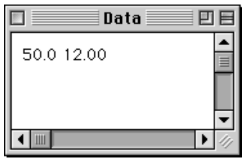
```
MODULE Hw99Pr0982;
    IMPORT TextModels, TextViews, Views, PboxMappers;

    PROCEDURE **Rectangle***;
        VAR
            md: TextModels.Model;
            vw: TextViews.View;
            fm: PboxMappers.Formatter;
            width: REAL;
            length: REAL;
    BEGIN
        md := TextModels.dir.New();
        fm.ConnectTo(md);
        width := 3.6;
        length := 12.4;
        fm.WriteString("The width is "); fm.WriteReal(width, 1, 2); fm.WriteLn;
        fm.WriteString("The length is "); fm.WriteReal(length, 1, 2); fm.WriteLn;
        vw := TextViews.dir.New(md);
        Views.OpenView(vw)
    END Rectangle;

END Hw99Pr0982.
```

Figure 9.14
A program that inserts real values into a text model.



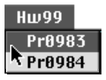
```
The width is 3.60
The length is 12.40
```

Figure 9.15
The output for the program in Listing 9.14

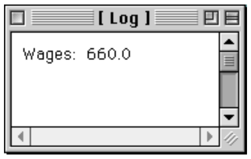


```
50.0 12.00
```

(a) The input window.



(b) The menu selection



```
Wages: 660.0
```

(c) The output.

Figure 9.16
The input and output of the program in Figure 9.18.

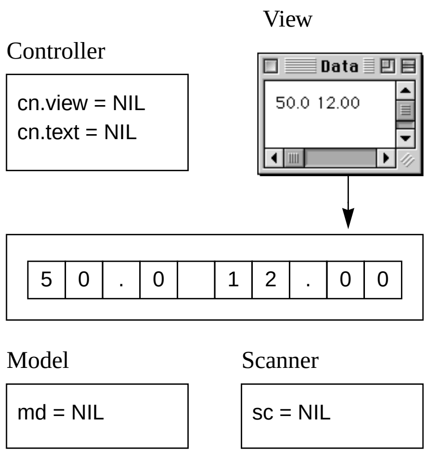
```
DEFINITION TextControllers;
    TYPE
        Controller = POINTER TO ABSTRACT RECORD (Containers.Controller)
            view-: TextViews.View;
            text-: TextModels.Model;
        END;
    PROCEDURE Focus (): Controller;
END TextControllers.
```

Figure 9.17
The interface for TextControllers. Many items from the interface are omitted from this listing.

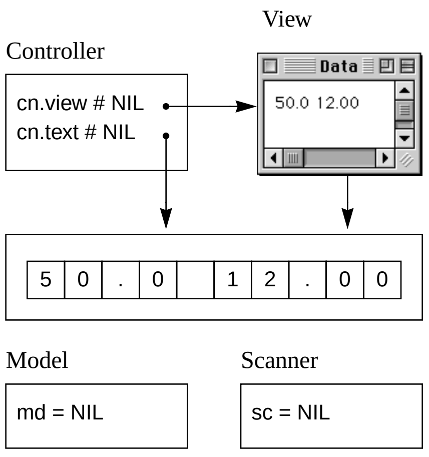
```
MODULE Hw99Pr0983;
    IMPORT TextModels, TextControllers, PboxMappers, StdLog;

    PROCEDURE ComputeWages*;
        VAR
            md: TextModels.Model;
            cn: TextControllers.Controller;
            sc: PboxMappers.Scanner;
            hours, rate: REAL;
            wages: REAL;
```

Figure 9.18
A program that gets its input from the focus window.

```
    BEGIN
        cn := TextControllers.Focus();
        IF cn # NIL THEN
            md := cn.text;
            sc.ConnectTo(md);
            sc.ScanReal(hours);
            sc.ScanReal(rate);
            IF hours <= 40.0 THEN
                wages := hours * rate
            ELSE
                wages := 40.0 * rate + (hours - 40.0) * 1.5 * rate
            END;
            StdLog.String("Wages:  "); StdLog.Real(wages); StdLog.Ln
        END
    END ComputeWages;

END Hw99Pr0983.
```

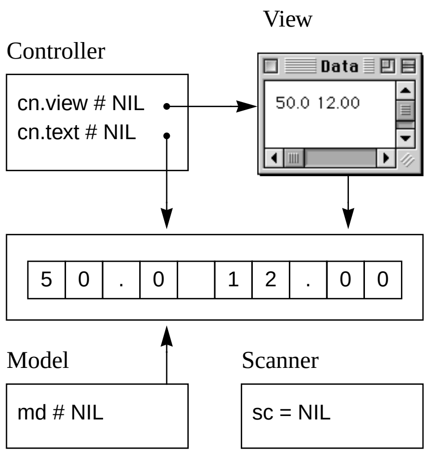


(a) BEGIN

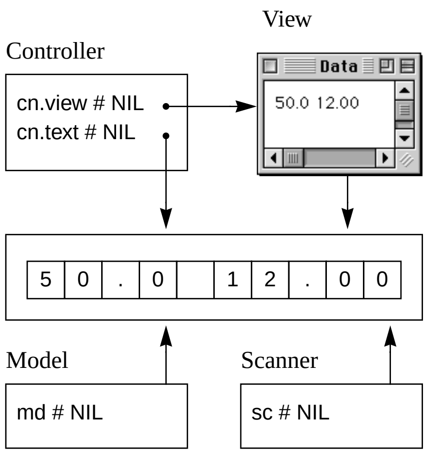


(b) cn := TextControllers.Focus()

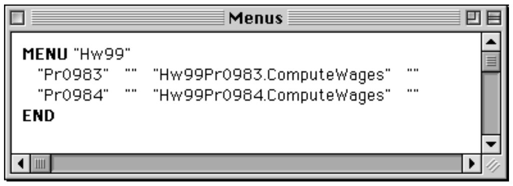
Figure 9.19
The effect of the MVC statements in Figure 9.18



(c) md := cn.text



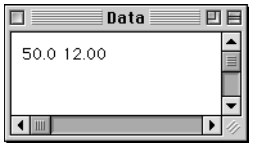
(d) sc.ConnectTo(md)



```
MENU "Hw99"
  "Pr0983"  ""  "Hw99Pr0983.ComputeWages"  ""
  "Pr0984"  ""  "Hw99Pr0984.ComputeWages"  ""
END
```

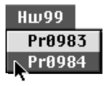
Figure 9.20
The menu document that produced the menu in Figure 9.16(b).

```
MENU "Hw99"
    "Pr0983"    ""    "Hw99Pr0983.ComputeWages"    ""
    "Pr0984"    ""    "Hw99Pr0984.ComputeWages"    ""
END
```

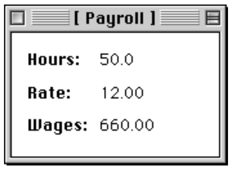


```
50.0 12.00
```

(a) The input window.



(b) The menu selection



```
Hours:   50.0
Rate:    12.00
Wages:   660.00
```

(c) The output.

Figure 9.21
The input and output of the program in Figure 9.22.

```
MODULE Hw99Pr0984;
    IMPORT TextModels, TextControllers, PboxMappers,
        PboxStrings, Dialog, StdCmds;
    VAR
        d*: RECORD
            hours-, rate-: ARRAY 16 OF CHAR;
            wages-: ARRAY 16 OF CHAR
        END;

    PROCEDURE ComputeWages*;
        VAR
            md: TextModels.Model;
            cn: TextControllers.Controller;
            sc: PboxMappers.Scanner;
            hours, rate: REAL;
            wages: REAL;
```

Figure 9.22
A program that gets its input from the focus window and puts its output in a dialog box.

```
    BEGIN
        cn := TextControllers.Focus();
        IF cn # NIL THEN
            md := cn.text;
            sc.ConnectTo(md);
            sc.ScanReal(hours);
            sc.ScanReal(rate);
            IF hours <= 40.0 THEN
                wages := hours * rate
            ELSE
                wages := 40.0 * rate + (hours - 40.0) * 1.5 * rate
            END;
            PboxStrings.RealToString(hours, 1, 1, d.hours);
            PboxStrings.RealToString(rate, 1, 2, d.rate);
            PboxStrings.RealToString(wages, 1, 2, d.wages);
            StdCmds.OpenAuxDialog('Hw99/Rsrc/Dlg0984', 'Payroll');
            Dialog.Update(d)
        END
    END ComputeWages;

END Hw99Pr0984.
```