UI software architectures & Modeling interaction

(part of this content is based on previous classes from A. Bezerianos, S. Huot, M. Beaudouin-Lafon, N.Roussel, O.Chapuis)
Assignment 1

Design and implement an interactive tool for creating the layout of comic strips

Software architecture - MVC
structure of an interactive system

What we see
- output

What we act with
- input

What happens
- treatment
- computation
- communication
- data (storage and access)

visible part « front end »

invisible part « back end »
example 1

- data model (albums, artists, categories, etc.)
- communication with iTunes server
- manage queries
- manage sales
- security

back end

front end
example 2

- geometric models
- calculations (transformations, rendering, etc.)
- store and access designs

front end

back end
example 3

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Email</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>6825</td>
<td>JUANITA LAMBERT</td>
<td><a href="mailto:mb4fasley@everyma11.biz">mb4fasley@everyma11.biz</a></td>
<td>719 MANNING HWY, CLYDE 75504</td>
</tr>
<tr>
<td>5740</td>
<td>GREG CABRERA</td>
<td>chandler@btmail biz</td>
<td>739 GENESEE BLVD, CORDELE, 17433</td>
</tr>
<tr>
<td>8590</td>
<td>ALBASA WISE</td>
<td><a href="mailto:hulweb@veryode.net">hulweb@veryode.net</a></td>
<td>205 ALICE RD, CAMILLA 14855</td>
</tr>
<tr>
<td>9282</td>
<td>SHARON WINTERS</td>
<td><a href="mailto:lorgan@hotmail1.com">lorgan@hotmail1.com</a></td>
<td>955 COHEN PIKE, TYRONE 511</td>
</tr>
<tr>
<td>2150</td>
<td>KRISTY FRANKS</td>
<td><a href="mailto:jgates3@somemall11.com">jgates3@somemall11.com</a></td>
<td>1471 ALEXIS PKWY, BALDWIN 85</td>
</tr>
<tr>
<td>10027</td>
<td>JEFF RICE</td>
<td><a href="mailto:diebold@btmail.org">diebold@btmail.org</a></td>
<td>104 DUNDEE PKWY, HOGBNsville 3741</td>
</tr>
<tr>
<td>7972</td>
<td>TAMARA BRYANT</td>
<td><a href="mailto:holwem@everyma11.us">holwem@everyma11.us</a></td>
<td>1382 WOGAN BLVD, CITY OF CALHOUN, 43790</td>
</tr>
<tr>
<td>5824</td>
<td>ALISHA YANG</td>
<td><a href="mailto:foundwong@hotmail1.net">foundwong@hotmail1.net</a></td>
<td>716 HOGANS DR, HARDING 58332</td>
</tr>
<tr>
<td>3402</td>
<td>JASON NGUYEN</td>
<td>haxedô<a href="mailto:hirs@ma12u.com">hirs@ma12u.com</a></td>
<td>527 MICHAEL CRES, FORT STEWART, 14654</td>
</tr>
<tr>
<td>3620</td>
<td>LINDSEY CABRERA</td>
<td><a href="mailto:askedreams@btmail.com">askedreams@btmail.com</a></td>
<td>1420 LAZELERIE HTS, FORT STEWART, 21650</td>
</tr>
<tr>
<td>3911</td>
<td>ANTHONY MATHES</td>
<td><a href="mailto:cfsnreki@everyma11.co.uk">cfsnreki@everyma11.co.uk</a></td>
<td>1325 OKEY LN, THUNDERBOL, 36366</td>
</tr>
<tr>
<td>572</td>
<td>JARED FORD</td>
<td><a href="mailto:orlying@btmail.co.uk">orlying@btmail.co.uk</a></td>
<td>723 EUCLID RD, FORSYTH, 42014</td>
</tr>
<tr>
<td>9720</td>
<td>AUTUMN WILLIAMS</td>
<td><a href="mailto:roomwhere@hotmail1.net">roomwhere@hotmail1.net</a></td>
<td>200 HILL PARK, NORCROSS, 58355</td>
</tr>
<tr>
<td>3447</td>
<td>MARION BROWN</td>
<td><a href="mailto:hwanegen@btmail.co.uk">hwanegen@btmail.co.uk</a></td>
<td>739 MIDDLE PATH, MACON 9944</td>
</tr>
<tr>
<td>9356</td>
<td>HOPE HAYNES</td>
<td><a href="mailto:sayofil@veryode.com">sayofil@veryode.com</a></td>
<td>1023 COOPER RIDERS CRES, STAVENVILLE, 342</td>
</tr>
<tr>
<td>2259</td>
<td>JEANNIE RANDOLPH</td>
<td><a href="mailto:womto@veryode.net">womto@veryode.net</a></td>
<td>972 EDISON PATH, CENTREVILLE, 45980</td>
</tr>
<tr>
<td>8264</td>
<td>RACHAEL CONLEY</td>
<td><a href="mailto:smolewhite@everyode.net">smolewhite@everyode.net</a></td>
<td>1223 STEVENS CT, CARROLLTON, 40004</td>
</tr>
<tr>
<td>4224</td>
<td>TRACIE DAVENPORT</td>
<td><a href="mailto:swhitaker52@veryode.net">swhitaker52@veryode.net</a></td>
<td>549 TALLEY RUN, BRISTOL, 53113</td>
</tr>
<tr>
<td>1836</td>
<td>NANCY GOFF</td>
<td><a href="mailto:holdfire@urma.us">holdfire@urma.us</a></td>
<td>147 FLEMING HWY MORA, 37823</td>
</tr>
<tr>
<td>6294</td>
<td>KATHY MORENO</td>
<td><a href="mailto:nadcode@everyma11.co.uk">nadcode@everyma11.co.uk</a></td>
<td>1341 BYRNE DR, SUMMER, 80490</td>
</tr>
</tbody>
</table>

- tabular structure
- storage and data access

back end

front end
link between the two parts
... programming using an organization model

organize, structure an interactive application by separating:

- Data and their treatment: **the Model**
- Data representation: **the View**
- Application behavior to input: **the Controller**
MVC is:

- A *design pattern* (standardized design solution independent of programming language)
- A *software architecture* (a way to structure an application or a set of software packages)

Introduced in 1979 by Trygve Reenskaug

Strongly linked to OO programming (Smalltalk)
MVC: *ideal* interactions between components

- **Model**: application functionality, data access and management
- **View**: presentation of data and functionality to the user
- **Controller**: manage user input, update application behavior

User flow:
1. User interacts with View
2. View updates Controller
3. Controller updates Model
4. Model updates View
5. View updates user
MVC: interactions between components

Model
- application functionality
- data access and management

View
- presentation of data and functionality to the user

Controller
- manage user input
- update application behavior
**MVC: interactions between components**

- **Model**
  - application functionality
  - data access and management

- **View**
  - presentation of data and functionality to the user

- **Controller**
  - manage user input
  - update application behavior
MVC: interactions between components

Model
- application functionality
- data access and management

View
- presentation of data and functionality to the user

Controller
- manage user input
- update application behavior

user input
notification of input
**MVC: interactions between components**

- **Model**
  - application functionality
  - data access and management

- **View**
  - presentation of data and functionality to the user

- **Controller**
  - manage user input
  - update application behavior

User input → notification of input → notification of state change
**MVC**: interactions between components

- **Model**
  - application functionality
  - data access and management

- **View**
  - presentation of data and functionality to the user

- **Controller**
  - manage user input
  - update application behavior

Internal operations

Notification of state change

Notification of input

User input
MVC: interactions between components

Model
- application functionality
- data access and management

View
- presentation of data and functionality to the user

Controller
- manage user input
- update application behavior

select a View

internal operations

notification of state change

user input

notification of input
**MVC: interactions between components**

- **Model**
  - application functionality
  - data access and management

- **View**
  - presentation of data and functionality to the user

- **Controller**
  - manage user input
  - update application behavior

**Interactions**
- User input → Model
- Model → Controller
- Controller → View
- View → Model
- Model → Controller
- Controller → Model

**Notifications**
- Notification of state change
- Notification of input
- Internal operations
MVC: interactions between components

- Model
  - application functionality
  - data access and management

- View
  - presentation of data and functionality to the user

- Controller
  - manage user input
  - update application behavior

- Controller -> View: select a View
- View -> Controller: notification of state change
- Model -> Controller: notification of input
- Model -> View: request state

- Model: internal operations
- Controller: refresh
- View: user input
MVC: interactions between components

Alternative architecture

Model
- application functionality
- data access and management

Controller
- manage user input
- update application behavior

View
- presentation of data and functionality to the user

Internal operations
Notification of state change
Notification of input
Update a View
User input
Refresh
MVC: referencing between components
MVC: the model

The model:

- Represents data
- Gives access to data
- Gives access to data management functionality
- Exposes the application functionality

Functional layer of the application
MVC: the view

The view:

- Shows the (or one) representation of the data in the model
- Ensures consistency between data representation and their state in the model (application)

Output of the application
MVC: the controller

The controller:

- Represents the application behavior w.r.t. user actions
- Translates user actions to actions on the model
- Calls the appropriate view w.r.t. the user actions and the model updates

Effect and treatment of input
advantages of MVC

Clean application structure

Adapted to concepts of O-O programming

Independence of
data – representation – behavior

Modular and reusable
disadvantages of MVC

Implementation complex for large applications

Too many calls between components
- « Spaghetti » code

Controller and View are often tightly linked to Model (and often to each other)

need to adapt implementation
MVC and Java Swing Widgets

Model-View-Controller separation not strict

Model categories:
- Visual status of GUI controls, e.g., pressed or armed button
- Application-data model, e.g., text in a text area
- Swing uses a model by default for each widget

View & Controller (often part of the same UI object)
- Look & Feel + Listener
- Examples: JButton, JLabel, JPanel, etc.
example

```
<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Sport</th>
<th># of Years</th>
<th>Vegetarian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kathy</td>
<td>Smith</td>
<td>Snowboarding</td>
<td>5</td>
<td>false</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>Rowing</td>
<td>3</td>
<td>true</td>
</tr>
<tr>
<td>Sue</td>
<td>Black</td>
<td>Knitting</td>
<td>2</td>
<td>false</td>
</tr>
<tr>
<td>Jane</td>
<td>White</td>
<td>Speed reading</td>
<td>20</td>
<td>true</td>
</tr>
<tr>
<td>Joe</td>
<td>Brown</td>
<td>Pool</td>
<td>10</td>
<td>false</td>
</tr>
</tbody>
</table>
```

Table Object

```
javax.swing.JTable
javax.swing.table.TableModel
```

Table Model Object

Table Data

```
javax.swing.JTable
javax.swing.table.TableModel
```
example

The data

Object[][] data = {
   {"Kathy", "Smith", "Snowboarding", new Integer(5), new Boolean(false)},
   {"John", "Doe", "Rowing", new Integer(3), new Boolean(true)},
   {"Sue", "Black", "Knitting", new Integer(2), new Boolean(false)},
   {"Jane", "White", "Speed reading", new Integer(20), new Boolean(true)},
   {"Joe", "Brown", "Pool", new Integer(10), new Boolean(false)}
};
example

The model

class MyTableModel extends AbstractTableModel {
    private String[] columnNames = ...
    private Object[][] data = ...

    public int getColumnCount() {
        return columnNames.length;
    }

    public int getRowCount() {
        return data.length;
    }

    public String getColumnName(int col) {
        return columnNames[col];
    }

    public Object getValueAt(int row, int col) {
        return data[row][col];
    }
    ...
}
The view

TableModel dataModel = new MyTableModel();

JTable table = new JTable(dataModel);
JScrollPane scrollpane = new JScrollPane(table);
public class MySelectionListener implements ListSelectionListener {
    private JTable table;

    public MySelectionListener(JTable table){
        this.table = table;
        table.setCellSelectionEnabled(true);
        ListSelectionModel cellSelectionModel = table.getSelectionModel();
        cellSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        cellSelectionModel.addListSelectionListener(this);
    }

    public void valueChanged(){
        ...
    }
}
Modeling Interaction
WIMP interfaces

WIMP: Window, Icons, Menus and Pointing

Presentation
- Windows, icons and other graphical objects

Interaction
- Menus, dialog boxes, text input fields, etc

Input
- pointing, selection, ink/path

Perception-action loop
- feedback
**direct manipulation**  
Ben Shneiderman (1983)

1. Persistent representation of objects of interest
2. Use of physical actions instead of complex syntax
3. Operations are quick, incremental, reversible, and their effect on objects is immediately visible *(feedback)*
4. Incremental learning, to permit use of the interface with little prior knowledge
direct manipulation: examples

text editors (e.g., Word, OpenOffice)
bitmap/vector graphics (e.g., Photoshop, Illustrator).
Counter-example: Latex ...

Icon interaction:
- Generic interface
- Use of metaphors
- drag-and-drop
direct manipulation?
direct manipulation problems

Identifying objects of interest

- example: styles in Word

Immediate feedback difficult when there is a delay between action and result

Direct or indirect manipulation?

- menus, dialog boxes, scroll-bars, etc.
describing interactions: state machines

Finite Automata
State = interaction state
Transition = input events

State Machine
- boolean expressions of events associated to transitions (guard)
- actions associated to transitions (not always present)

Example:
example: dragging windows

- Click inside the window
- Click on the bar of the window / Pick the window
- Click outside the window
- Drag / Move the window
- Release / Release the window
example: dragging windows

```
mouseDown(e) & !window.getBar().contains(e.getPoint())

mouseDown(e)
    & window.getBar().contains(e.getPoint())
    / action1(e)

S1

S2

mouseMove(e) / action2(e)

action1(e) {
    x0 = e.getX();
    y0 = e.getY();
}

action2(e) {
    translateWindow(e.getX() - x0, e.getY() - y0);
    action1(e);
}
```
state machines & MVC

View

- presentation of data and functionality to the user

Controller

- manage user input
- update application behavior

update a View

notification of input

user input

refresh
representing states

Common approach: use of global variables within a controller

```java
public enum State {S1, S2, S3, S4}
private State state = State.S1;
```

**or (use of multiple variables)**

```java
private boolean buttonPressed = false, mouseMoved = false;
```

In the following lecture, we’ll introduce SwingStates, a Java library for modeling interaction through states, state transitions, and state machines.
**common problems**

Getting trapped to states with no transitions (deadlocks)

Maintaining the code to capture new or unforeseen states is usually hard

An interaction can involve several UI components. Not always clear how to divide interaction between multiple controllers and state machines.
Which UI objects are involved?  
Which controller handles this interaction?
interaction modes

Mode: distinct state of the UI where the same user input has a different interpretation
- text vs. drawing mode in an editing tool
- typing capital or small characters

Mode switching
- e.g., Caps lock key, specialized button

Quasimode: mode being active through some constant action from the user
- e.g., use of modifier keys such as Shift, Alt, Control while typing or pointing
interaction modes: problems

« modes are a significant source of errors, confusion, unnecessary restrictions, and complexity in interfaces »

Jef Ruskin

Ruskin advocated for modeless interfaces. He also recommended the use of quasimodes instead of explicit modes.

Other points of view (Jacob Nielsen)

« users cannot cope with everything at once »
«...need the interface to narrow their attention »
« Real life is highly moded »
making modes visible
eliminating modes

Special mode for changing time  No modes, direct editing

What are the trade-offs in these designs?

(credits to Niall Murphy)