Design patterns

- By design patterns, we mean some arrangements of classes and objects and the relationships and collaborations between them.
- useful for solving recurring software design problems.
- **Example 1** *The ADT list (studied earlier) can be recognised as a design pattern. It includes not only the data structure but also a set of standard operations such as* insert, delete, search, sort *etc.. For each operation, we define* precondition, postcondition

java.util.Iterator *interface:*

```
public interface Iterator {
  public boolean hasNext();
  // postcondition: retruns true if the iterator
  // has more elements; returns false otherwise.

  public Object next()
            throws NoCuchElementException;
  // postcondition: if the iterator has no
  // more elements throws NoSuchElementException;
  // returns the next element otherwise

  public void remove()
         throws UnsupportedOperationException,
                IllegalStateException;
  // postcondition: If remove is not supported
  // throws UnsupportedOperationException;
  // else if next has not yet been called or
  // remove has been called since the last call
  // to next throws IllegalStateException;
  // else removes the last element returned by
  // this iterator.
} end Iterator
```


so the details of the implementation can be hidden.

Example 2 *The* iterator *can be seen as a design pattern in which a structured group of elements (vector, list, dictionary etc.) without exposing the internal structure.*

Template method design pattern

- A template method is the method responsible for an algorithm which contains the algorithm steps. Those steps required different implementation details are implemented by subclasses.
- The template method pattern is used to define algorithms.

Composite design pattern

- This is to group a set of *primitive* objects together to form a *composite*
- Composites can also be grouped with primitives, both can be viewed as components
- It is good to build hierarchies that extend to arbitrary depth

- **Example 3** java.awt *employs the composite design pattern to allow the components such as buttons, lists, labels, text fields to be constructed in a hierarchical ways*

Application framework

- This is a reusable software system in a particular application domain
- Design patterns help the software *reuse* and the application frameworks support the reuse at an even larger scale
- Framework determines the applications' overall design by providing most or all of its software components. e.g. AWT's frames, buttons etc.

- Frameworks support design reuse by prescribing many features of application's overall design
- Frameworks support code reuse by providing a library of useful components and by allowing *extending* exist classes

Advantages of using frameworks

1. A good framework captures expertise both in application and programming. By software reusing, the developpers can make use of the expertise which they do not have themselves
2. It is generally possible to build complex applications much more efficiently compared with the work from scratch

3. Applications derived from frameworks are generally reliable benefit from testing and effort went to frameworks
4. Applications based on a framework are relatively uniform

Disadvantages of using frameworks

1. It is difficult and time consuming to build a good framework
2. Evolving frameworks over time is a difficult and on-going task. Framework evolution is usually indispensable due to changes in application requirements and advances in technology
3. Applications based on a framework have to change in order to remain compatible with any new versions of

the framework

4. It may be impossible to customise a framework to exactly meet the application requirements.
5. Learning a framework can require a considerable time and effort
6. Developpers have to give up some controls over the design of application when using a framework.

Event-driven Programming

In javax.swing and java.awt

Recall the windows which we have created so far cannot react when we press a button. Why?
Recall package swing and awt

graphics

components

layout managers

[event handlers] (java.awt.event)
handles external events

Events

- Anything that takes place is an event.
- In Java an **event** is an *action* taken by the user.

Example

Common events

- Button event
- Scrollbar event
- Key pressed event
- Item chosen event

Handling events

The user

- presses a key or clicks the mouse
- clicks on a button, a scrollbar, a menu option, etc.

Computers cycles round in an event *loop*, waiting for an event.
Consider the following series of actions - 'serve in turn':

```
if  key_is_pressed
then Handle_key_action ...
else if  Mouse_dragged
   then Handle_mDrag_action ...
   else if  Mouse_clicked
      then Handle_mClick_action ..
...
```

What if we click the mouse out of a window?

- Only those "interesting" events are handled.
- Some methods should automatically invoke when an event happens.
- The methods are so-called *event handlers*.

Setting up the handling system

- In Java, events are *classified* and
- are *tied to* the types of components (e.g. Buttons, Text fields, Scrollbars, etc.).
- In order to start handlers automatically when an event happens, components need to be registered with a *listener* which is an interface.

Approaches

- Write a method for every class of events interested
- Register as a listener with components

Interaction with text fields

```
TextComponent
  TextField
  TextArea
```

Menus: choosing from a list

```
Choice
CheckBox
List
Menu
Popup
```


Applets in Action

Applets

- Short for "little applications"
- An applet is a Java program that operates within a browser
- Java is *hot* because it is integrated into World Wide Web
- Supporting applets
  - interpretation
  - JVM - Java Virtual Machine in all browsers

Converting an application to an applet

1. Check that all input/output relevant statements
2. Remove stopping statements
3. Overriding flow layout
4. Import applet package and extend Applet instead of Frame
5. Replace the class' constructor
6. Remove the main method
7. Refer to the applets' class file in a HTML file
8. Run the HTML file through an applet viewer or through a Web browser, e.g. HotJava, Netscape, Mosaic or Explorer.

Motivation

- Less traffic - The work is done on the user's machine
- Run faster - The user's machine can be dedicated to the applet
- Full facilities available - e.g. swing, awt.

Hypertext Markup Language - HTML

- A file has to be created with normally a FILENAME.**html**
- It has to include an important line which is so-called 'HTML tags' for an applet

```
<APPLET code="name" width=n height=m>
</APPLET>
```

Or

```
<APPLET code="NAME" codebase="DIRECTORY-NAME"
                              width=N height=M>
</APPLET>
```

Making an applet work

1. Prepare an applet (x.java)
   (a) Either
   - Write an applet *or*
   - Convert an application to an applet

   ```
   pico x.java
   ```

   (b) (Edit and) compile

   ```
   javac x.java
   ```

2. Prepare a HTML file
   (a) Write a HTML file (x.html *or* x.HTML)

   ```
   pico x.html
   ```

   (b) (Edit and) view

   ```
   appletviewer x.html
   ```

3. Put up on the Web page
   Suppose the Web page file is on a machine M
   (a) Decide which page to link
   (b) Move **x.class** and **x.html** file to M
   (the same directory as well)

3. Decide to put on CIS212/CS218 home page which is stored in a file *cis212.html*

```
<h3>
Announcement Board

</h3>

<h2>
CS217 Object-Oriented Programming/
CIS212a Computer Programming Paradigms
</h2>
<h3>
Pre-requirement: CIS107
</h3>
<h3>
Semester One
</h3>
<p>
Lecturer: Dr Ida Pu  (i.pu@gold.ac.uk) </p>
Surgery Hours: Tuesdays 1300-1400, 1500-1600
</p>

<p>
<ul>
Lectures
<li> <a href="slect1.ps">Week 1</a></li> (available)
...

</ul>
<p>
<ul>
<li> <a href="coursew1.ps">Coursework 1 </a></li> (available)
<li> <a href="coursew2.ps">Coursework 2 </a></li> (available)
</ul>
</p>

<p>
Please contact i.pu@gold.ac.uk if you have any problems.
</p>
<p>
```


Here are TO-DOs step-by-step:

Example 4 *1.*

```
import java.awt.*;
import java.applet.*;

public class warningApplet extends Applet {

  static private final int line = 15;
  static private final int letter = 5;

  public boolean handleEvent (Event e) {
    if (e.id == Event.WINDOW_DESTROY)
      System.exit (0);
    return super.handleEvent (e);
  }

  public void paint (Graphics g) {
    g.drawRect (2*letter, 2*line, 33*letter, 6*line);
    g.drawString("W A R N I N G",           9*letter, 4*line);
    g.drawString("Possible virus detected", 4*letter, 5*line);
    g.drawString("Reboot and run virus",    5*letter, 6*line);
    g.drawString("remover software",        7*letter, 7*line);
  }

}
```

2.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <TITLE></TITLE>
   <META NAME="GENERATOR" CONTENT="Mozilla/3.0Gold (Win95; I) [Netscape]">
</HEAD>
<BODY>

<P><APPLET code="warningApplet.class" width=300 height=200></APPLET></P>

</BODY>
</HTML>
```

```
Laboratories or Workshops: Tuesday 1400-1500, in WB300 and WB316
</p>
<ul>
<li> <a href="lab0.ps">Exercise 0</a></li> (available)
<li> <a href="lab1.ps">Exercise 1</a></li> (available)
...
</ul>
<p>
Test: running an applet
</p>
<ul>
<li> <a href="warningApplet.html"> virus </a> </li>
</ul>
```

4. Move files

cis212.html is currently on a machine "homepages".

```
[mas01ip@ma328 mas01ip]$ ftp homepages
                         ------------
Connected to homepages.gold.ac.uk.
220 Goldsmiths FTP Server ready.
Name (homepages:mas01ip): mas01ip
                          -------
331 Password required for mas01ip.
Password:
          ----------
230-
  __
 |  |--.-----.--------.-----.-----.---.-.-----.-----.-----.
 |     |  _  |        |  -__|  _  |  _  |  _  |  -__|__ --|
 |__|__|_____|__|__|__|_____|   __|___._|___  |_____|_____|
                                |__|        |_____|

 You mas01ip, have used 5259K out of 10000K (52 %) of your quota.

 This was last calcualted at Wed Dec  8 20:15:00 1999

230 User mas01ip logged in.
```

```
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd cis212
     ---------
250 CWD command successful.
ftp> cis212.html
     ^^^^^^^^^^^      ???
local: cis212.html remote: cis212.html
200 PORT command successful.
150 Opening BINARY mode data connection for cis212.html.
226 Transfer complete.
1821 bytes sent in 0.000166 secs (1.1e+04 Kbytes/sec)
ftp> put warningApplet.class
         -------------------
local: warningApplet.class remote: warningApplet.class
200 PORT command successful.
150 Opening BINARY mode data connection for warningApplet.class.
226 Transfer complete.
871 bytes sent in 8.9e-05 secs (9.6e+03 Kbytes/sec)
ftp> put warningApplet.html
         ------------------
local: warningApplet.html remote: warningApplet.html
200 PORT command successful.
150 Opening BINARY mode data connection for warningApplet.html.
226 Transfer complete.
276 bytes sent in 7e-05 secs (3.9e+03 Kbytes/sec)
ftp> bye
221 Goodbye.
[mas01ip@ma328 cis212]$
```

Applet security

Java guards

- Java checks the validity on the byte code arrived
- JVM will not perform any harmful operations

How applets work

1. Start Java runtime system
2. Call one of the following:
   - init()
   - destroy()
   - start()
   - stop()