Programación conducida por eventos y Applets como ejemplo de “Frameworks”

Agustín J. González

ELO-326. Seminario II

2do. Sem. 2001

Frameworks

- Un ***framework*** (“marco de trabajo”) es un conjunto de clases que implementan todos los servicios comunes de un cierto tipo de aplicación.
- Para construir una aplicación, el programador deriva de alguna de las clases del framework y agrega las operaciones específicas de la aplicación.
- Por ejemplo: considere la clase Windows para describir una ventana gráfica de una interfaz usuario. Se espera que otras clases sean derivadas de esta para proveer implementaciones de tareas como dibujar su contenido. Cada ventana debe saber como re-dibujarse cuando es restaurada desde el icono, o expuesta luego de ser cubierta por otra ventana.

Frameworks (cont)

- La idea no es que el usuario deba conocer todos los detalles como se maneja la interfaz usuario, sino sólo las operaciones claves que le son de interés.
- Ejemplo: cuando un área que tolera scrolling es movida, la operación que maneja esta labor invoca a la función paint para desplegar la porción de la imagen deseada. La clase base no tiene idea sobre qué se está pintando, pero sabe cuando el pintado es necesario.
- La clase base impone el orden de ejecución de los métodos provistos por la clase derivada.
- El programador tiene poca influencia en el orden en que las operaciones son llamadas.
- La programación sobre un framework maneja eventos aislados como pintado, comandos de menu, clicks del mouse, etc.
- Cuando el código termina, se retorna el control al framework.
- Este comportamiento es el conocido como programación conducida por eventos (event-driven programming)

Applet como un framework simple

- Un applet es un programa Java especial que está contenido en una página web.
- El applet es cargado y ejecutado por el navegador Web para agregar interactividad y dinamismo a una página Web.
- Un applet se ejecuta bajo el control de un navegador o el utilitario ***appletviewer***.
- La programación del applet se hace bajo un framework que implementa gran parte de la Graphical User Interface (GUI)
- Las applet poseen limitaciones por razones de seguridad; por ejemplo no se está permitido manipular archivos locales. Ver Tester.java y Tester.html

Ciclo de vida de un Applet

- Los pasos que tiene lugar cuando un applet es invocada son los siguientes:
- Cuando el navegador encuentra el rótulo <applet>, el navegador busca el código especificado en el parámetro y lo transfiere por la red hasta el browser (navegador).

```
<HTML>
<title>Applet Test Page</title>
<H1>Testing Applet</H1>
Esta es la Applet
<APPLET CODE=”codigo_de_la applet.class" WIDTH=200 HEIGHT=50>
</APPLET>
que estamos probando.
</HTML>
```

- Una vez cargado un objeto codigo_de_la applet es instanciado (creado) e inicializado (llamando a init()).
- Luego el método start() del applet es llamado y la apariencia del applet es desplegada (llamando a paint()).
- Luego el browser monitorea y controla los eventos asociados al applet (mouse, teclado, otros eventos (timers etc) y envía mensajes a los objetos manejadores de estos eventos que previamente deben ser registrados (event listeners)
- El navegador crea un hilo separado para que cada applet ejecute concurrentemente.

Ciclo de vida de un Applet (cont)

- Ejemplo Applet simple Hello.java y su prueba hello.html
  otro ejemplo: HelloApp.java helloApp.html
- El browser maneja la applet invocando métodos heredados de la clase Applets, los cuales pueden ser sobremontados. La clase define métodos vacíos por defecto.
- init(): Llamada al iniciar una nueva applet, luego que el código es cargado o recargado. Oprecaiones muy largas (carga de archivos desde la red) deberían hacerse en un hilo separado.
- start(): Llamando para poner el applet a correr. Este método es llamado cuando el applet ha sido creada e inicializada. El método paint() es llamado automáticamente después de start().
- stop(): Llamado para detener el applet. Normalmente significa detener cualquier thread (hilo) creado en start().
- destroy(): Llamada para efectuar limpieza final.
- El applet puede transitar varias veces por los estado partida y detenida.
- Ejemplo applet con captura de eventos del mouse Click.java Click.html

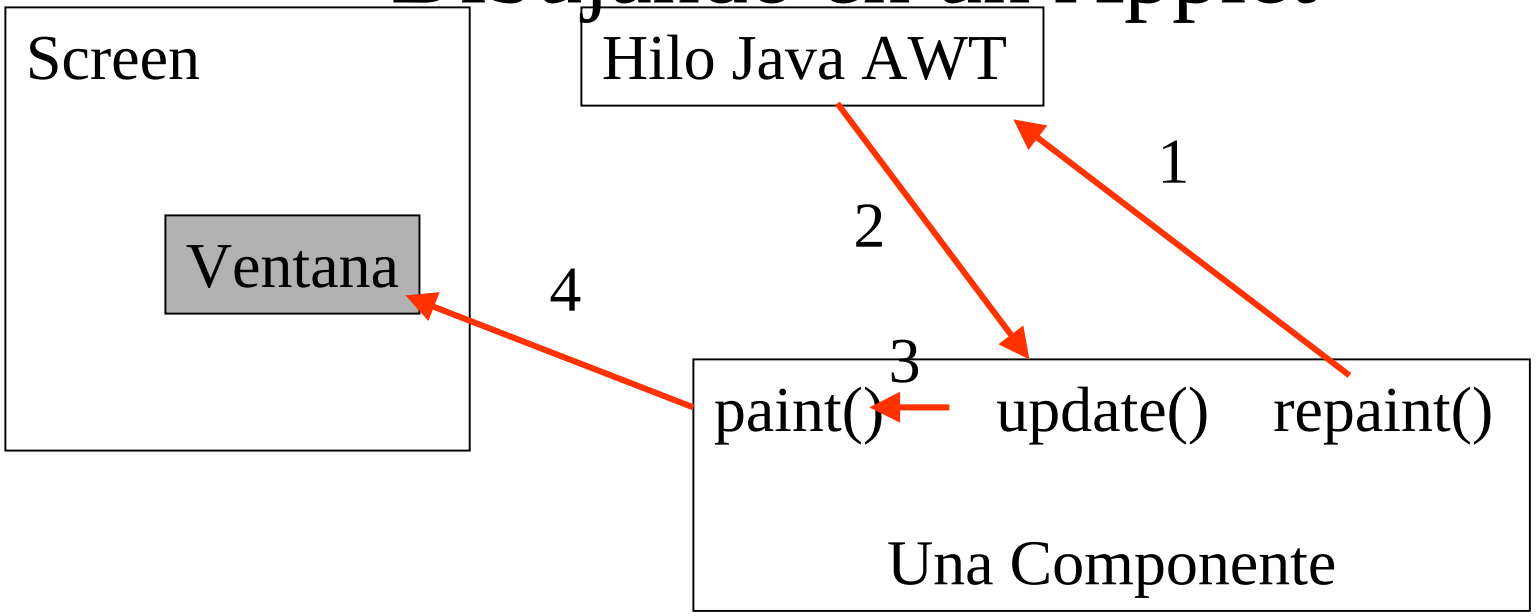
Dibujando en un Applet

- La applet tiene un área de trabajo en la página del browser. El largo y alto son especificados en el rótulo <APPLET>
- Usualmente hacen uso de componentes GUI (Graphical User Interface) tales como botones, menús, etc.
- Una componente GUI es un objeto sofisticado (parte del framework gráfico) que ya tiene considerado comportamientos esperados. Además es capaz de dibujarse a si mismo, cambiar de tamaño, color, y procesa algunos eventos (cierre ventana, iconiza, etc)
- La clase Applet hereda de las clases Panel, Conteiner, y Component las cuales porveen varias operaciones.
- Para dibujar en un applet usamos objetos de la AWT (Abstract Windowing Toolkit).
- Tres métodos son invocados para desplegar un dibujo:
- paint(): Este método dibuja la componente completa. Componentes estándares como buttons y menus tienen ya definida estos correctamente métodos. Componentes usuarios deben sobremontar el método

  public void paint(Graphics g)

  para dibujar en forma personalizada.
- repaint(): Se llama a esta función para registrar un requerimiento de actualización de la apariencia de la componente.
- update(): la AWT llama al método update() de la componente en respuesta a un repaint(). Este método limpia el área a redibujar y luego llama a paint().

Dibujando en un Applet



- Hay varias formas para estos métodos:
  repaint(), repaint(long tm), repaint(int x, int y, int w, int h), repaint(int x, int y, int w, int h, long tm)
- La componentes pueden obtener sus dimensiones con getSize().width, getSize().height
- el método paint debe ser invocado vía llamados a repaint.
- Cuando paint es llamado se pasa el contexto gráfico que le corresponde. Ver clase Graphics.
- En JDK 1.2 se expande la AWT para permitir manejo en 2D. Aun cuando el objeto especificado es tipo Graphics, el pasado es Graphics2D. Así podemos hacer:
  Graphics2D g2 = (Graphics2D) g;
  Y así tener acceso a todos los métodos de esta clase. Ver clase Graphics2D.

Ejemplo Tic Tac Toe

- Este ejemplo posee varias clases creadas por el programador para dar vida a esta aplicación.
- Las clases son:
- TicTacToe.java : la applet Controla globalmente el juego.
- TicBoard.java : implementa el tablero de juego, dibuja el tablero y las marcas X u O
- TicGame.java : Maneja la dinámica del juego, chequea movidas, registra posiciones, genera movidas, y determina el estatus del juego.
- ClickHandler.java : encargado de recibir las movidas del usuario.
- Aplicación resultante: TicTacToe.html
- Destacar:
  Obtención del URL de origen en init()
  Carga de imágenes remotas en setBoard()
  ClickHandler
  Drawline en drawBoard
  drawImage en drawPiece
  Estrategia de jugadas 1.- encontrar movida ganadora, bloquear movida que haga ganar, hacer movida válida.
- Otro ejemplo: Applet para interpolación lineal
- Mañana: ¿Cómo agregamos sonido al juego?

Efectos de Sonidos en Applets

- Java applets pueden manejar diversos formatos de sonido. Entre ellos: au, wav, AIFF, MIDI.
- La clase applet ofrecen los métodos play(URL url) y play(URL dir, String filename) para reproducir un archivo de audio.
- Por razones de seguridad el URL debe corresponder al computador origen del Applet.
- También se disponen de los métodos getAudioClip(URL url) y getAudioClip(URL dir, String filename). Éste crea un objeto del tipo AudioClip.
- La clase AudioClip ofrece métodos play(), loop(), y stop().
- Ver nueva clase TicSound: Código

```
public class TicSound extends TicTacToe
{   public void init()
    {  super.init();
       returnClip=  getAudioClip(codeBase, "audio/return.au");
       joyClip=   getAudioClip(codeBase, "audio/joy.au");
       beepClip=  getAudioClip(codeBase, "audio/beep.au");
    }
// otros métodos .....
   protected AudioClip returnClip;
    protected AudioClip joyClip;
    protected AudioClip beepClip;
}
```

Efectos de Sonidos en Applets

- Java applets pueden manejar diversos formatos de sonido. Entre ellos: au, wav, AIFF, MIDI.
- La clase applet ofrecen los métodos play(URL url) y play(URL dir, String filename) para reproducir un archivo de audio.
- Por razones de seguridad el URL debe corresponder al computador origen del Applet.
- También se disponen de los métodos getAudioClip(URL url) y getAudioClip(URL dir, String filename). Éste crea un objeto del tipo AudioClip.
- La clase AudioClip ofrece métodos play(), loop(), y stop().
- Ver nueva clase TicSound: Código

```
public class TicSound extends TicTacToe
{   public void init()
    {  super.init();
       returnClip=  getAudioClip(codeBase, "audio/return.au");
       joyClip=   getAudioClip(codeBase, "audio/joy.au");
       beepClip=  getAudioClip(codeBase, "audio/beep.au");
    }
// otros métodos .....
   protected AudioClip returnClip;
    protected AudioClip joyClip;
    protected AudioClip beepClip;
}
```

Manejo de Eventos

- Ya hemos visto algo en los ejemplos Click y TicTacToe.
- Las acciones efectuadas por la AWT (abstract Windowing Toolkit) ante la llegada de un evento son:
  - Se determina qué componente genera el evento (botton, etc)
  - Se crea un objeto que representa el evento, son instancias de subclases de java.util.EventObject
  - Se reporta el evento a la componente invocando al método correspondiente al evento. Esto siempre que se haya registrado un objeto para procesar los eventos.
  - Se invoca el método correspondiente para cada objeto registrado con ese evento.
- Event Listeners: El manejo de evento se hace entonces estableciendo objetos escuchadores y registrandolos con las fuentes de eventos. Una fuente de evento entrega el evento al llamar métodos bien establecidos y fijos en los objetos registrados.
- El proceso de registro y "desregistro" se hace con addXyzListener y removeXyzListener
- Clase "event listener": Una forma de crear objetos capaces de procesar eventos es crear una clase que implemente la interfaz del evento de interés, Ej. KeyListener, MouseListener, ActionListener, etc.

Ejemplo

```
public class ButtonApplet extends Applet
{
   private TextField t;
   private Button b;

   public void init()
  {
      b = new Button("Cuenta");
      t = new TextField(5);
      b.addActionListener(new
     MyActionListener(t));
      add(b);
      add(t);
  }
}
```

Registro Action Listener

Clase que implementa el event Listener

```
class MyActionListener implements
     ActionListener
{
   TextField _t;
   int count;
   MyActionListener(TextField t)
   {
     _t = t;
     count=0;
   }
```

Método que se hace cargo del evento

```
   public void
     actionPerformed(ActionEvent e)
   {  _t.setText(" "+count++);  }
}
```

Escritura del event listener con Clases adaptadoras

- Java ya dispone de clases que implementan las interfaces más comunes.
- El programador puede derivar la clase y definir sólo aquellos métodos que le son de interés. La clase base tiene implementación por defecto para los otros métodos.
- **Manejador Anónimo**
- Es muy común que se deba crear sólo un objeto para manejar los eventos y para él debamos crear una nueva clase.
- Dada esta repetición, Java permite crear este objeto como una instancia de una clase anónima con una notación especial:
  new superClassName() { class body}