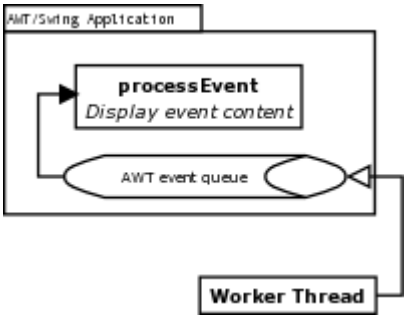
Java AWT/Swing event queue

All graphical components of the Java graphical user interface (GUI) AWT or Swing belong to the GUI main thread. Java AWT and Swing are not thread safe, which is why it is not permitted to access any GUI component from any thread other than the main GUI thread.
There are many Java GUI applications which do not work correctly, because they access a GUI component from a second thread. As many computer are single CPU computers with a single core CPU these applications seems to work correctly but they may crash without apparent cause or reason.
Running such applications on a multi core or multi CPU machine will crash.

The solution to this kind of problem is to create user defined AWT event and store them in the AWT event queue. The main AWT GUI thread is then able to dispatch these events.



The diagram above show an AWT/Swing application where a worker thread stores events in the AWT event queue. The AWT main thread takes the events from the queue and the execution of the event will be done from within the AWT main thread context.

User defined AWT events.

To create an AWT event, first create a class which is derived from the AWTEvent class.
In my example this class is called SimpleAWTEvent (see below).
Every AWT event needs an event ID, AWT internal events IDs are reserved.
To get your event ID use the static variable java.awt.AWTEvent.RESERVED_ID_MAX. If you work with different AWTEvent implementations you need to create unique event IDs.

```
public class SimpleAWTEvent extends AWTEvent
{
   public  static final int EVENT_ID = AWTEvent.RESERVED_ID_MAX + 1;
   private String          str;
   private int             percent;


   SimpleAWTEvent( Object target, String str, int percent)
   {
      super( target, EVENT_ID);
      this.str     = str;
      this.percent = percent;
   }
   public String getStr()
   {       return( str );    }
   public int getPercent()
   {       return( percent );    }
}
```

In this example the SimpleAWTEvent class passes a string (str) and a integer (percent) to the main AWT dispatcher.

The Worker

The SimpleWorker in this example inherits java.lang.Thread and implements the run() method. From the method java.awt.Toolkit.getDefaultToolkit().getSystemEventQueue() we get an instance of the event queue. In this queue we are able to queue SimpleAWTEvent events.

```
Object target = Instance of the main thread
EventQueue eventQueue = Toolkit.getDefaultToolkit().getSystemEventQueue();
eventQueue.postEvent( new SimpleAWTEvent( target, “Hello GUI”, 1 ));
```

Enable a new Event type

In your main application you need to enable the SimpleWorker event ID. The event ID is stored in the public static variable EVENT_ID in SimpleAWTEvent.

```
enableEvents( SimpleAWTEvent.EVENT_ID);
```

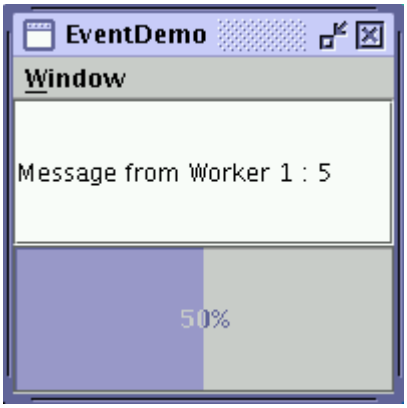
Dispatch the event in the main AWT/Swing GUI

To dispatch the event in the main AWT/Swing GUI thread you need to overload the "**processEvent(AWTEvent event)**" method which is defined in the java.awt.Window class.

```
protected void processEvent( AWTEvent event)
{
    if ( event instanceof SimpleAWTEvent )
    {
       SimpleAWTEvent ev = (SimpleAWTEvent) event;
       textField.setText( ev.getStr() );                  // access GUI component
       progressBar.setValue( ev.getPercent() );           // access GUI component
    }
    else     // other events go to the system default process event handler
    {
       super.processEvent( event );
    }
}
```

In the **processEvent** method you are running in the AWT main thread context. Now you are able to access all GUI components (in our case the textField and progressBar).

The complete example is stored in the jar file EventDemo.jar.
This pages in PDF format EventDemo.pdf



This is a screen dump of the example.

To run the demo use : "java -jar EventDemo.jar"
To view the source extract the files with "jar xvf EventDemo.jar"

If you have any questions or suggestion please let me know.

Mail to :
Stephan@Kauss.org