GUI - *multithreading*

- Quando si scrive una GUI si assiste a un cambio di paradigma
  - Programma con flusso centralizzato -> programma reattivo basato su eventi

- Per questo motivo si adotta un'architettura multi-thread.
- Ciò permette di ottenere interfacce sempre reattive e che non si "bloccano"
  - Esempio: se ho un client di posta elettronica e qualcuno mi spedisce un messaggiodi 10M voglio comunque potere: vedere l'intestazione della mail, vedere una barra che mi indica quanto manca al download, scrivere una nuova mail nel frattempo, ecc.

3 tipi di thread - *thread iniziale*

- Il thread principale si occupa di inizializzare l'interfaccia grafica.


```
import javax.swing.*;

public class HelloWorldSwing {
    private static void createAndShowGUI() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        createAndShowGUI();
        //   posso andare avanti a fare altro
    }
}
```

3 tipi di thread - *Event Dispatch Thread (EDT)*

- Tutto il codice per la gestione dell'interfaccia grafica viene eseguito dentro l'EDT.

- Esiste una coda di eventi di sistema che raccoglie i click del mouse, la pressione sui tasti della tastiera, ecc.
- L'EDT controlla ciclicamente la coda. Raccoglie un evento e decide cosa farne. Se è un click su un bottone chiama il metodo per la gestione del click del mouse su quel bottone.

- La gestione mediante coda garantisce che venga rispettato un certo ordine nella gestione degli eventi

Immaginiamo cosa sarebbe successo se...

- ...non avessimo avuto l'EDT.

- Ipotizziamo di avere due tasti: A e B
  - Tasto A:
    - Legge il contenuto di un file e lo presenta all'utente all'interno di una textBox.
  - Tasto B:
    - Legge dallo stesso textBox e scrive i contenuti su un file
- Cosa succede se premo molto velocemente prima A e poi B?
  - Probabilmente avrei un comportamento anomalo!!!

- E' per questo che tutto ciò che lavora sull'interfaccia gira all'interno di un unico thread (l'EDT).

Swing e Thread

- Nelle Swing, la gestione degli eventi e il ridisegno delle finestre e dei componenti grafici è affidata ad un **unico** thread
  - **Dispatching Thread**

- Perché un unico thread? Perché le Swing non sono thread safe! L'accesso ad un oggetto da parte di più thread può causare problemi!

- Dispatching Thread:
  - Viene eseguito in background
  - Processa gli elementi presenti nella coda degli eventi grafici
  - Anche il ri-disegno dei componenti è un evento (paint)

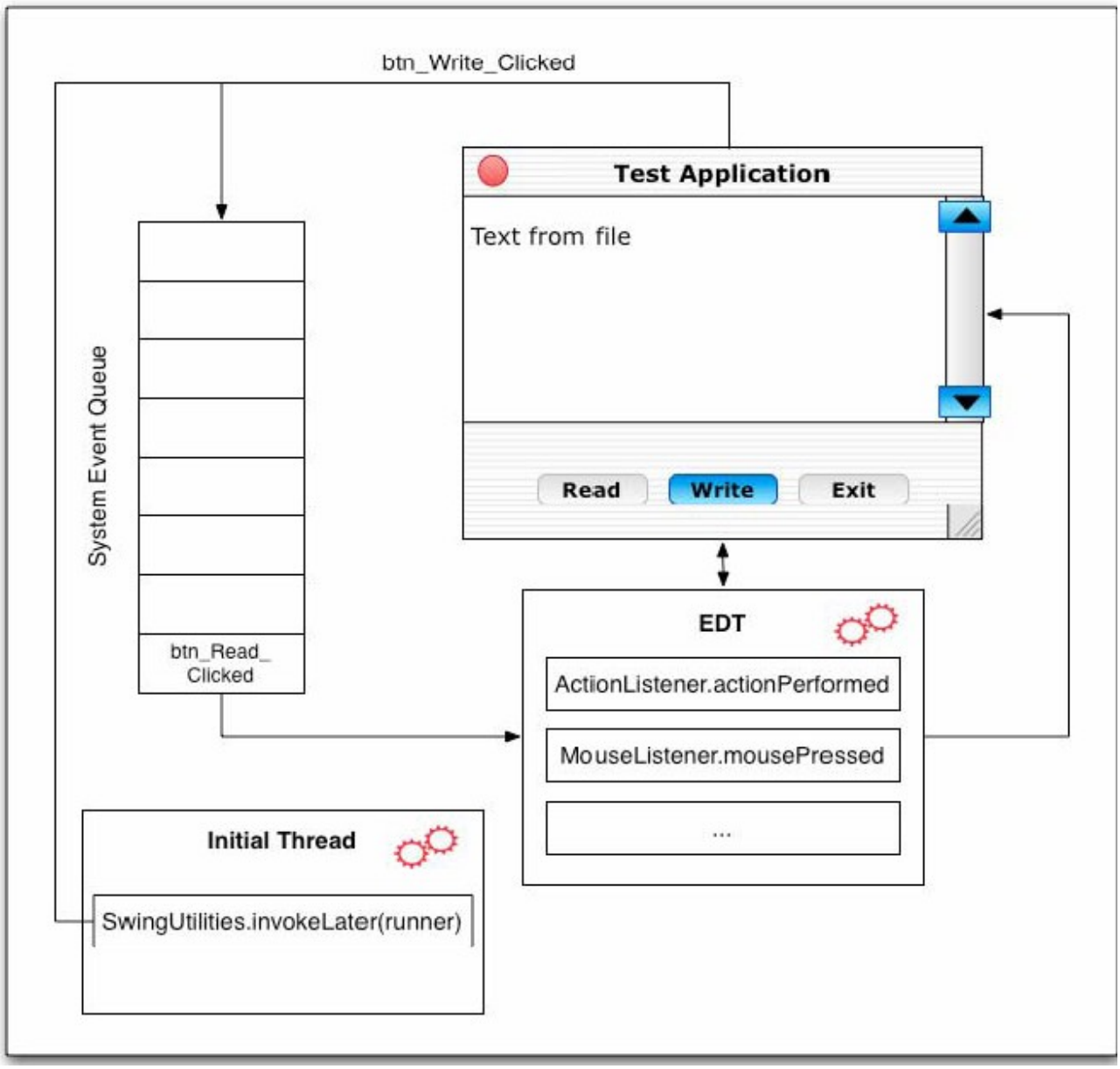
SwingUtilities

- Il modo giusto di inizializzare una GUI quindi non è quello che abbiamo visto prima, ma...

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        createAndShowGUI();
    }
} );
```

- In questo modo creo un oggetto Runnable (il quale si occupa di creare l'interfaccia) e chiedo che venga gestito dall'EDT.
- Esiste anche:
  - SwingUtilities.invokeAndWait(new Runnable() {...})

La coda degli eventi di sistema



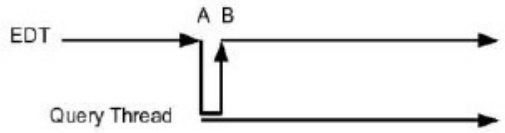
In figura si esemplifica il caso di un'applicazione con gestione di due eventi: quello di azione (implementazione del metodo actionPerformed) e quello di mouse (implementazione del metodo mousePressed)

Considerazioni per la progettazione

- Per avere una buona interfaccia grafica cbisogna essere reattivi
  - Bisogna dare l'impressione all'utente che sia sempre in controllo
- Il segreto è cercare di scrivere EventListener che siano veloci
- In questo modo eviteremo situazioni in cui un click può portare a una lunga attesa prima di vedere un risultato
- Ma se devo fare qualcosa che richiede tempo? Come faccio?
  - Dobbiamo introdurre il terzo tipo di thread

3 tipi di thread - *Worker Thread*

- Il Worker thread
  - viene creato dal programmatore
  - per l'esecuzione di compiti gravosi in background

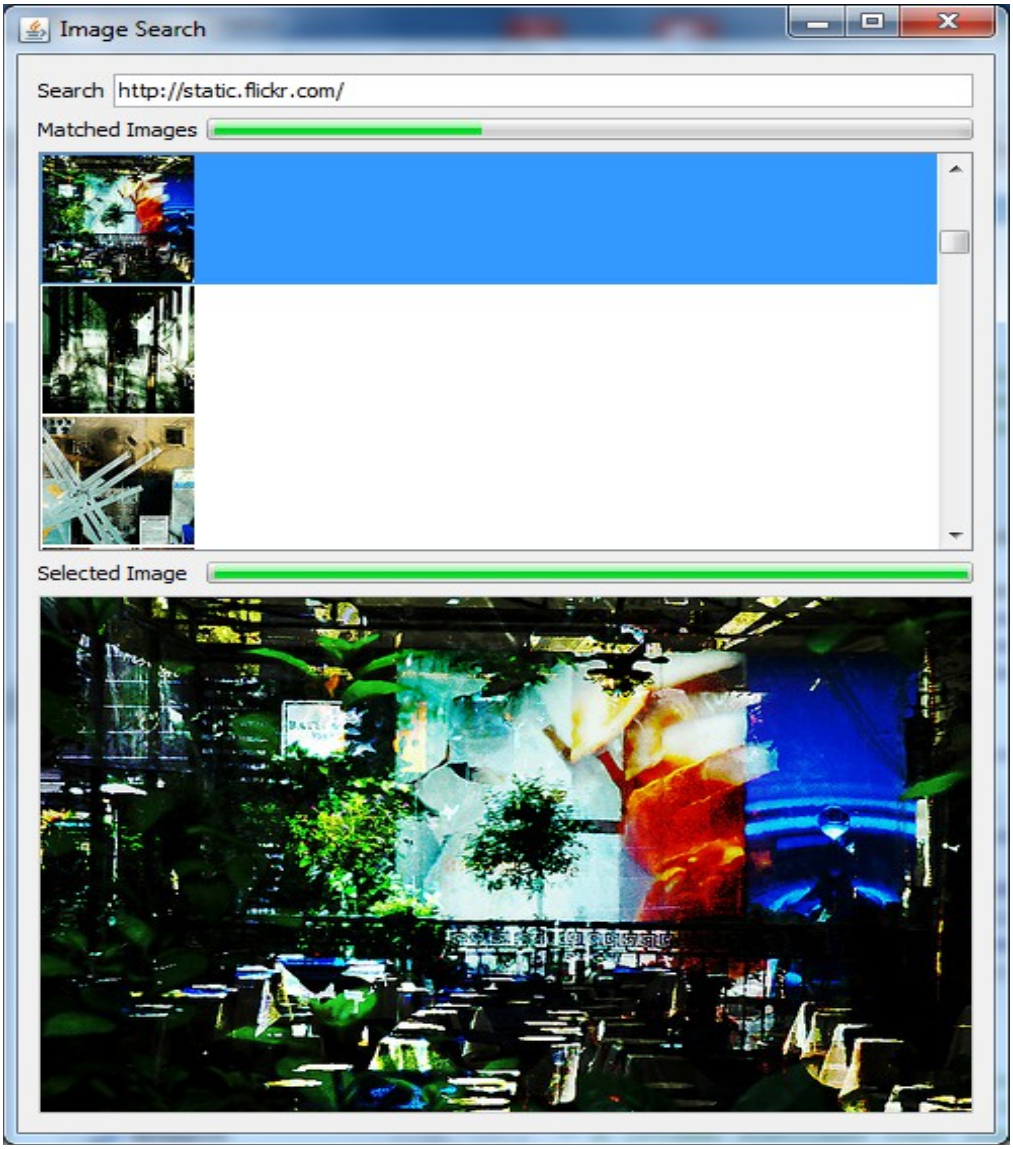


Un indizio...

- Per capire a fondo dobbiamo introdurre (per davvero) il multi-threading e la concorrenza in Java
- Per chi volesse approfondire per conto proprio...
  - Il modo giusto è quello di usare un oggetto SwingWorker
    - Definisce un metodo che si occupa del lavoro in background, e
    - Un secondo metodo che invece verrà eseguito nell'EDT e che fa da ponte tra il background thread e la GUI

Sitografia: http://home.dei.polimi.it/morzenti/IngSw/Eventi_Swing_2007.pdf
http://download.oracle.com/javase/tutorial/uiswing/concurrency/initial.html

http://www.programmazione.it/index.php?entity=eitem&idItem=35473
con link all'articolo http://java.sun.com/developer/technicalArticles/javase/swingworker/
da cui scaricare una Demo (tipico uso di **SwingWorker** per velocizzare il caricamento di immagini)



nb: in ambiente JCreator salvare nella cartella del **bytecode** il file che contiene la API_KEY

potendo creare nell'applicazione MainFrame un canale per leggerla senza impostare diverso *path*:

InputStream is = MainFrame.class.getResourceAsStream(**"FlickrKey.properties"**);

Per scegliere dalla galleria di immagini online, digitare nella casella di testo con etichetta Search l'indirizzo:

http://static.flickr.com/