Code Glosser

Developed by: Louis Jenkins

Advised by: Drue Coles

What is Code-Glosser

- Academic Enrichment Tool
  - Makes it easy for instructors to provide feedback to students
  - Feedback in the form of a markup
    - Highlighted segment of code with a message
- Free of cost and Open-Sourced under the BSD 3-Clause License
- Created for Independent Study for Dr. Coles
  - Dr. Coles needed software to provide feedback to students
- Diverged from Dr. Coles' original plan into something much more
  - Going from a (not-so) simple NetBeans plugin to it's own stand-alone application

Features

- Portability
  - To Markup
    - Java Runtime Environment 8
  - To View
    - Web Browser (HTML + CSS support)
- Effortless Markups
  - Markup portions of code in moments
    - Leave feedback for student with a message
  - Templates
    - Save more time by applying templated markups
    - Comes with both Google's Java and C++ style guide templates
- Project-Scale Grading
  - Markup and exportation of entire projects
    - Projects are as simple as a directory containing code
    - Does not need to be a NetBeans or Eclipse project
      - Can even be individual files
- Syntax Highlighting
  - Uses the highlight.js library for syntax highlighting
    - Supports 169 languages
    - Does not require an internet connection to markup or to view

Why should you use Code-Glosser

Current Approach to Grading

- Print out project
  - Waste of paper
  - Bulky to carry around
  - Very bad for multi-file projects
- Leave feedback in pen
  - Your handwriting may not be intelligible to everyone
    - Mistakes cannot be erased
  - You are much less likely to leave feedback for small or less important things
    - Especially if a lot of people make the same mistakes
  - Feedback left most likely short
    - Need to fit on a single page, and also need to write them in bulk
  - Time Consuming
    - Too much time and effort to do for each and every student, every assignment for every class
- Hand back to student
  - They may not even read it anyway
    - May have been wasted time

The Code-Glosser Approach

- Open the project in Code-Glosser
  - Can open any file inside of the project
    - File markups are preserved when switching files
- Markup project
  - Can markup while you're reading it
    - No need to make multiple passes
  - Easy to make changes
    - Create, Modify, and Delete on demand
  - Easy to leave feedback on very common mistakes
    - Templates make it trivial
  - Leave feedback on any section of code
    - Can be a variable, a block of code, or even a function
- Export Project
  - Exports into a compressed archive that the student may view
    - No printing needed
    - All electronic

Demo

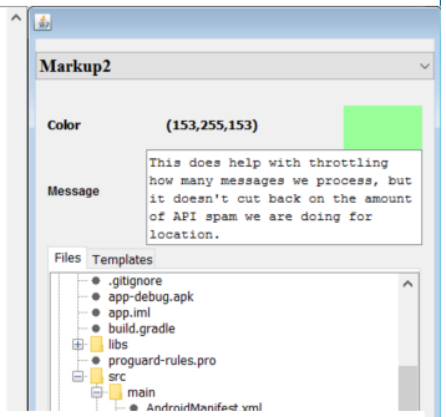
Youtube Video

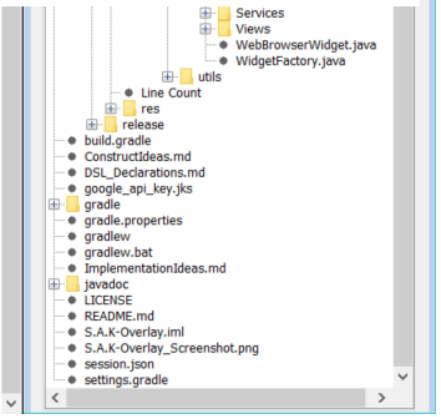
Included as a URL and not embedded due to it causing crashes on multiple machines tested on.

Java – S.A.K-Overlay (Android Windows Manager)

```
@Override
public void setup() {
    super.setup();
    TextView mAddress = (TextView) getContentView().findViewById(R.id.google_maps_address);
    ((MapFragment) getChildFragmentManager().findFragmentById(R.id.map)).getMapAsync(googleMap -> {
        mMap = googleMap;
        mMap.setMyLocationEnabled(true);
        mMap.getMyLocation();
        mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
        mMap.setTrafficEnabled(true);
        mMap.setBuildingsEnabled(true);
        mGeocoder = new Geocoder(getActivity(), Locale.getDefault());
        mMap.setOnMyLocationChangeListener(location -> {
            mOnNextAddress.onNext(location);
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(location.getLatitude(), location.getLongitude()), 18.0f));
        });
    });
    mOnNextAddress
            .observeOn(Schedulers.io()) // Do processing of next address change on a background IO-Bound thread, as geocoder can block.
            .throttleLast(5, TimeUnit.SECONDS) // As changes can occur rather frequently, we only want to handle it once per 5 seconds.
            .map(location -> {
                String address = null;
                try {
                    List<Address> addresses = mGeocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1);
                    if (addresses != null && !addresses.isEmpty()) {
                        Address a = addresses.get(0);
                        ArrayList<String> partialAddresses = new ArrayList<String>(a.getMaxAddressLineIndex());
                        for (int i = 0; i < a.getMaxAddressLineIndex(); i++) {
                            partialAddresses.add(a.getAddressLine(i));
                        }
                        address = TextUtils.join(System.getProperty("line.separator"), partialAddresses);
                    }
                } catch (IOException | IllegalArgumentException e) {
                    Log.w(getClass().getName(), e.getMessage());
                }
                return address;
            }
            )
            .filter(s -> s != null)
            .observeOn(AndroidSchedulers.mainThread()) // Swap back to the main thread for the easy bit.
            .subscribe(mAddress::setText);
}

}
```

This does help with throttling how many messages we process, but it doesn't cut back on the amount of API spam we are doing for location.





Go – Concurrent Map (Runtime)

```
// If hdr is nil, then no bucketData has been created yet. Do a wait-free creation of bucketData;
for hdr == nil {
        // Note that bucketData has the same first 3 fields as bucketHdr, and can be safely casted
        newHdr := (*bucketHdr)(newobject(t.bucketdata))
        // Since we're setting it, may as well attempt to acquire lock and fill out fields
        newHdr.lock = gptr
        newHdr.parent = arr
        newHdr.parentIdx = uint32(idx)
        // If we fail, then some other Goroutine has already placed theirs.
        if atomic.Casp1((*unsafe.Pointer)(unsafe.Pointer(&arr.buckets[idx])), nil, unsafe.Pointer(newHdr)) {
                // If we succeed, then we own this bucket and need to keep track of it
                g.releaseBucket = unsafe.Pointer(newHdr)
                // Also increment count of buckets
                atomic.Xadd(&arr.count, 1)
                setByUs = true
        }
        // Reload hdr
        hdr = (*bucketHdr)(atomic.Loadp(unsafe.Pointer(&arr.buckets[idx])))
}

// Attempt to acquire lock
for {
        // If we set the lock, we skip the locking part all together
        if setByUs {
                break
        }
        // Reset backoff variables
        spins = 0
        backoff = DEFAULT_BACKOFF

        lock := atomic.Loaduintptr(&hdr.lock)

        // If the state of the bucket is INVALID, then either it's been deleted
        if lock == INVALID {
                // Reload hdr, since what it was pointed to has chan
                hdr = (*bucketHdr)(atomic.Loadp(unsafe.Pointer(&arr
                // If the hdr was deleted, then attempt to create a nev
                for hdr == nil {
                        // Note that bucketData has the same
                        newHdr := (*bucketHdr)(newobject(t.bu
                        // Since we're setting it, may as well a
                        newHdr.lock = gptr
                        newHdr.parent = arr
                        newHdr.parentIdx = uint32(idx)
```

```
}

// Attempt to acquir
for {
        // If we set the lock, we skip the locking part all together
        if setByUs {
                break
        }
        // Reset backoff variables
        spins = 0
        backoff = DEFAULT_BACKOFF

        lock := atomic.Loaduintptr(&hdr.lock)
```

Instead of entering the for loop, either create a new label to jump to after the loop, or set it as the condition (for !setByUs)

Markup1

Color (255,51,51)

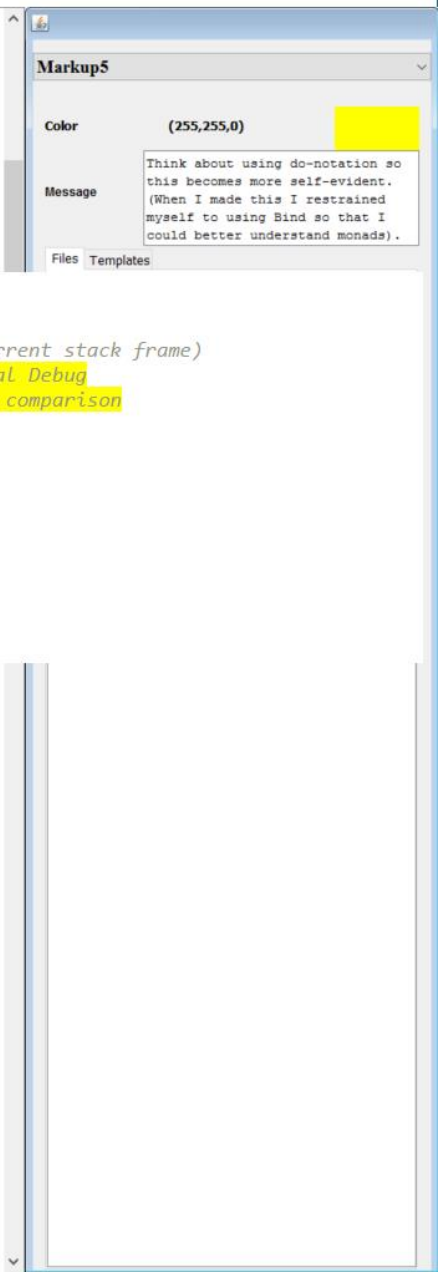
Message: Instead of entering the for loop, either create a new label to jump to after the loop, or set it as the condition (for !setByUs)

Files | Templates

- runtime
  - alg.go
  - append_test.go
  - asm.s
  - asm_386.s
  - asm_amd64.s
  - asm_amd64p32.s
  - asm_arm.s
  - asm_arm64.s
  - asm_mips64x.s
  - asm_ppc64x.h
  - asm_ppc64x.s
  - asm_s390x.s
  - atomic_arm64.s
  - atomic_mips64x.s
  - atomic_pointer.go
  - atomic_ppc64x.s
  - callers_test.go
  - cgo
  - cgo.go
  - cgocall.go
  - cgocallback.go
  - cgocheck.go
  - cgo_mips64x.go
  - cgo_mmap.go
  - cgo_ppc64x.go
  - chan.go
  - chanbarrier_test.go
  - chan_test.go
  - closure_test.go
  - compiler.go
  - complex.go
  - complex_test.go
  - concurrent_map.go
  - cpuprof.go
  - cputicks.go
  - crash_cgo_test.go
  - crash_nonunix_test.go
  - crash_test.go
  - crash_unix_test.go
  - debug

Haskell – JVM ByteCode Interpreter

```haskell
{- | Starting point of execution of ByteCode isntructions -}
execute :: Runtime_Environment -> IO ()
execute env = head <$> readIORef (stack env)  -- Take the head of the stack (current stack frame)
  >>= \frame -> when (debug_mode env) (debugFrame frame >>= putStrLn) -- Optional Debug
  >> getPC' frame >>= \pc -> maxPC frame >>= \max_pc -> -- Program Counters for comparison
  -- While valid program_counter, execute instruction
  unless (pc >= max_pc) (getNextBC frame >>= execute' frame >> execute env)
    where
      -- The main dispatcher logic
      execute' :: StackFrame -> ByteCode -> IO ()
      execute' frame bc
        -- NOP
        | bc == 0 = return ()
        -- Constants
        | bc >= 1 && bc <= 15 = constOp frame bc
        -- Push raw byte(s)
        | bc == 16 || bc == 17 =
          -- Special Case: 0x10 pushes a single byte, but 0x11 pushes a short
          (if bc == 16 then fromIntegral <$> getNextBC frame else getNextShort frame)
            >>= pushOp frame . fromIntegral
        -- Load from runtime constant pool
        | bc >= 18 && bc <= 20 =
          -- Special Case: 0x12 uses only one byte for index, while 0x13 and 0x14 use two
          (if bc == 18 then fromIntegral <$> getNextBC frame else getNextShort frame)
            >>= loadConstantPool env . fromIntegral >>= pushOp frame
        -- Loads
        | bc >= 21 && bc <= 53 = loadOp frame bc
        -- Stores
        | bc >= 54 && bc <= 86 = storeOp frame bc
        -- Special Case: 'dup' is used commonly but ignored, so we have to stub it
        | bc == 89 = return ()
        -- Math
        | bc >= 96 && bc <= 132 = mathOp frame bc
        -- Conditionals
        | bc >= 148 && bc <= 166 = cmpOp frame bc
        -- Goto: The address is the offset from the current, with the offset being
        -- the next two instructions. Since we advance the PC 2 (+1 from reading this
        -- instruction), we must decrement the count by 3 to correctly obtain the target.
        | bc == 167 = getNextShort frame >>= \jmp -> modifyPC frame (+ (jmp - 3))
        -- Return
        | bc == 177 = return ()
        -- Runtime Stubs
        | bc >= 178 || bc <= 195 = runtimeStub env frame bc
        | otherwise = error $ "Bad ByteCode Instruction: " ++ show bc
```



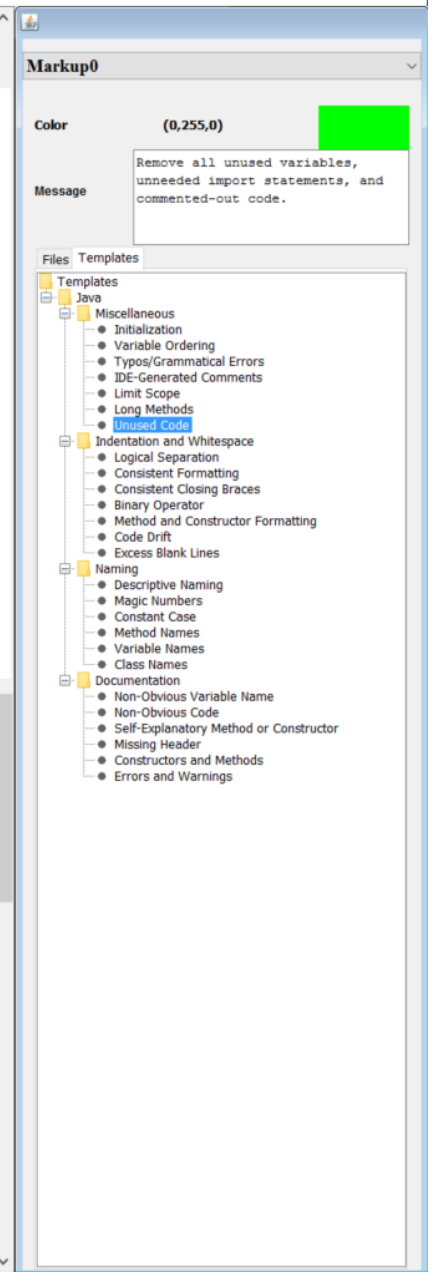
Think about using do-notation so this becomes more self-evident.
(When I made this I restrained myself to using Bind so that I could better understand monads).

C – MoltarOS (Multitasking)

```
static void task_switch(regs_t *UNUSED(regs)) {
    // Check if our timeslice expired
    if (current->ticks) {
        current->ticks--;
        return;
    }

    // KTRACE("Preparing to task switch");
    volatile task_t *curr = current;

    // Ensure we preserve our current state so
    // Information needing to be saved are the
    uint32_t esp, ebp, eip;

    // The stack pointer and base pointer can
    // but the instruction pointer is a tough one
    // MUST be of the instruction following 'rea
    asm volatile ("mov %%esp, %0" : "=r" (esp
    asm volatile ("mov %%ebp, %0" : "=r" (ebp
    eip = read_eip();

    // If the instruction pointer is DUMMY_EIP,
    // to a new task. Note that the task itself wo
    // which forces it to return to it's normal exe
    if (eip != DUMMY_EIP) {
        // KTRACE("Preserving Current Process Info: pid: %x, eip: %x, esp: %x, ebp: %x", current->id, eip, esp, ebp);
    } else {
        // KTRACE("Child returned, continuing task...");
        return;
    }

    // Select the next process
    current = LIST_NEXT(current, next_task);
    if (!current) {
        current = LIST_FIRST(&tasks);
    }
    current->ticks = TICKS_PER_SLICE;

    // Preserve the current task's progress.
    curr->eip = eip;
    curr->esp = esp;
    curr->ebp = ebp;

    // Jump to other task. The task's stack pointer and base pointer will accurately point to the
    // task's previous position on the stack.
```

```
    // KTRACE("Preparing to task switch");
    volatile task_t *curr = current;

    // Ensure we preserve our current state so we return here later.
    // Information needing to be saved are the registers esp, ebp, and eip
    uint32_t esp, ebp, eip;

    // The stack pointer and base pointer can readily be retrieved
    // but the instruction pointer is a tough one. The instruction pointer
    // MUST be of the instruction following 'read_eip'
    asm volatile ("mov %%esp, %0" : "=r" (esp));
    asm volatile ("mov %%ebp, %0" : "=r" (ebp));
    eip = read_eip();

    // If the instruction pointer is DUMMY_EIP, then we have just jumped here after switching
    // to a new task. Note that the task itself could also have been inside of an interrupt handler,
    // which forces it to return to it's normal execution.
    if (eip != DUMMY_EIP) {
        // KTRACE("Preserving Current Process Info: pid: %x, eip: %x, esp: %x, ebp: %x", current->id, eip, esp, ebp);
    } else {
        // KTRACE("Child returned, continuing task...");
        return;
    }
```

Remove all unused variables, unneeded import statements, and commented-out code.



X86 Assembly – MoltarOS (Bootstrap)

```
bootstrap_page_directory:
        ; Because we are remapping our kernel to VIRTUAL_ADDRESS_START our current
        ; instruction pointer will cause us to page fault (and then double fault and
        ; then triple fault) and trigger a hardware reset. Hence before we enable paging
        ; we must identity map each virtual address to it's respective physical address.
        dd PDE_DEFAULT
        ; All pages besides the kernel's are not present in memory
        times (KERNEL_INDEX - 1) dd 0
        ; Kernel Entry
        dd PDE_DEFAULT
        ; Kernel Stack (4MB)
        dd 0x00400083
        ; Pages after the kernel
        times (1024 - KERNEL_INDEX
```

```
section .text

; This is the entry point for the linker, which will start the ins
; valid physical memory location.
global start
        start equ (_start - VIRTUAL_ADDRESS_STA

global _start

        ; Setup paging -- Note that at this point, we a
        ; to ensure that we begin relative to the physical location 0x0.
        _start:
                ; Load the Page Directory we setup above
                mov ecx, (bootstrap_page_directory - VIRTUAL_ADDRESS_START)
                mov cr3, ecx

                ; Enable 4MB paging by setting the PSE bit in CR4
                mov ecx, cr4
                or ecx, 0x00000010
                mov cr4, ecx

                ; Enable paging by setting the PG bit in CR4
                mov ecx, cr0
                or ecx, 0x80000000
                mov cr0, ecx

                ; NOTE: At this point in time, paging has been enabled, and we are now using virtual memory.
                ; This includes the instruction pointer, and as such we need to perform a long jump. As well,
                ; the CPU needs to clear it's instruction cache it prefetched, which can be cleaned out by
                ; performing a branch instruction, so this does performs double duty.
```

```
; This is the Page Directory that we use to bootstrap.
bootstrap_page_directory:
        ; Because we are remapping our kernel to VIRTUAL_ADDRESS_START our current
        ; instruction pointer will cause us to page fault (and then double fault and
        ; then triple fault) and trigger a hardware reset. Hence before we enable paging
        ; we must identity map each virtual address to it's respective physical address.
        dd PDE_DEFAULT
        ; Al
        times
        ; Ker
        dd PDE_DEFAULT
        ; Kernel Stack (4MB)
        dd 0x00400083
        ; Pages after the kernel
        times (1024 - KERNEL_INDEX - 2) dd 0
```

A 4MB stack is excessively large, especially since each forked procesed will inherit this. Chunk it into 4KB zones and use the rest for the local heap.

```
section .text
```

Markup4

Color (255,255,0)

Message: A 4MB stack is excessively large, especially since each forked procesed will inherit this. Chunk it into 4KB zones and use the rest for the local heap.

- sched
- script.sh
- x86
  - boot.asm
  - boot.o
  - exceptions.c
  - exceptions.o
  - gdt.c
  - gdt.o
  - idt.c
  - idt.o
  - idt_handler.asm
  - idt_handler.o
  - io_port.c
  - io_port.o
  - linker.ld
- libc
- tmp

Templates

- No currently implemented way to create them in Code-Glosser
  - Unless significant demand, cannot do so
- Templates need to be done by hand
  - Tedious but saves a lot of time in the long run
  - No currently formatted style guide available
    - Does come with one for Java and C++
- Templates are required to have a special format
  - Contains Categories and Templates
    - Described in the next few slides

Templates – JSON Format

- Categories
  - Used to logically separate markups
  - Can contain other categories and markups
  - Must contain a key “category” set to *true*
  - Must contain a title which describes it
  - Must contain a body containing an array of other templates or categories

Templates – JSON Format

- Templates
  - The actual markup data in question
  - Must contain a title that describes it
  - Must contain a message that is displayed
  - Can optionally contain a color that is shown in Code-Glosser

Templates – Other Languages

JSON

```json
"category": true,
"title": "C++ (Google Style Guide)",
"body": [
  {
    "category": true,
    "title": "Header Files",
    "body": [
      {
        "title": "The '#define' Guard",
        "message": "All header files should have '#define' guards to prevent multiple inclusion. The format of
      },
      {
        "title": "Header File Dependencies",
        "message": "Don't use an #include when a forward declaration would suffice."
      },
      {
        "title": "Inline Functions",
        "message": "Define functions inline only when they are small, say, 10 lines or less."
      },
      {
        "title": "The -inl.h Files",
        "message": "You may use file names with a -inl.h suffix to define complex inline functions when needed
      },
      {
        "title": "Function Parameter Ordering",
        "message": "When defining a function, parameter order is: inputs, then outputs."
      },
      {
        "title": "Names and Order of Includes",
        "message": "Use standard order for readability and to avoid hidden dependencies: C library, C++ librar
      }
    ]
  },
```

YAML

```yaml
- category: true
  title: C++ (Google Style Guide)
  body:
    - category: true
      title: Header Files
      body:
        - title: 'The ''#define'' Guard'
          message: >-
            All header files should have '#define' guards to prevent multiple
            inclusion. The format of the symbol name should be
            <PROJECT>_<PATH>_<FILE>_H_.
        - title: Header File Dependencies
          message: 'Don''t use an #include when a forward declaration would suffice.'
        - title: Inline Functions
          message: >-
            Define functions inline only when they are small, say, 10 lines or
            less.
        - title: The -inl.h Files
          message: >-
            You may use file names with a -inl.h suffix to define complex
            inline functions when needed.
        - title: Function Parameter Ordering
          message: >-
            When defining a function, parameter order is: inputs, then
            outputs.
        - title: Names and Order of Includes
          message: >-
            Use standard order for readability and to avoid hidden
            dependencies: C library, C++ library, other libraries' .h, your
            project's .h.
```

Implementation of Code-Glosser

The “Interesting” Design and Implementation details

Syntax Highlighting

- Code-Glosser originally a NetBeans plugin
  - NetBeans only allows Java Swing
    - Java Swing has extremely minimal support for HTML and CSS, no JavaScript
- Need to support any language for it to become truly useful
  - Found ‘highlight.js’ library
    - Supports 169 languages and 77 styles
    - Works by executing a JavaScript function on page load to add CSS Tags around code for syntax highlighting
      - Remember: Java Swing does not support JavaScript
  - Need a way to execute JavaScript, and extract the changed HTML after page load
    - JavaScript is available in JavaFX, and so switched to an independent platform
      - Needs to be a headless browser so as not to disturb user.
  - Set the current style.css and extracted HTML to display
- Need to be usable offline
  - Library downloaded and coupled with Code-Glosser
- Exported HTML needs to be independent
  - Cannot rely on a relative CSS file
  - Since JavaScript is already executed in JavaFX WebView, can embed CSS for both Markup and Style

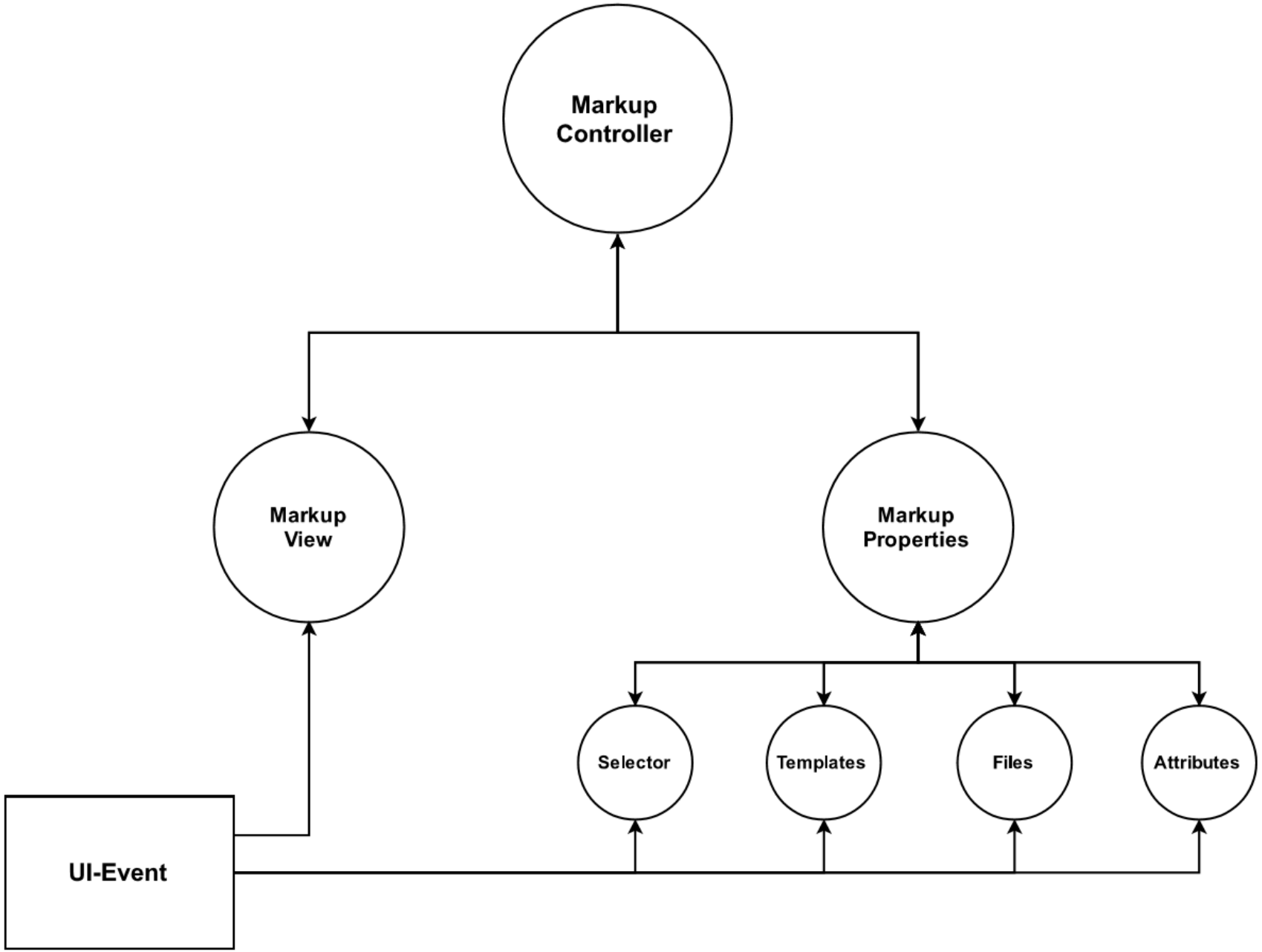
Java Swing and Concurrency – Background

- Java Swing has one UI Thread
  - Called the “AWT Dispatch Thread”
  - In charge of updating the UI
  - Thread used by default
- Problem
  - When performing long-running tasks, UI will not update and will ‘freeze’
    - UI will no longer be responsive as it cannot return until it finishes the task.
- Naïve Solution
  - “Throw more threads at the problem until it goes away”
  - Only the UI Thread is allowed to manipulate the UI
    - GUI components are not ‘thread-safe’
  - Having multiple threads process different UI events simultaneously can result in undefined behavior without proper synchronization or communication
    - Synchronization is a huge bottleneck and can end up making the code perform even worse than before
      - Non-Blocking synchronization is over-complicated (and **VERY** difficult) for plain old GUI work
    - Communication through Message-Passing is viable, but how can we do so?

Handling UI Events – The Problem

- UI Events can occur at anytime, even while processing another event
- UI Events may need to notify multiple components
  - These components may need to **react** and notify other components based on certain conditions
    - Including the component that notified it
      - Events need to flow **both ways**
- What about **IO** Events
  - IO Event – A network call or reading from hard drive
  - **All** IO Bound tasks should **not** be performed on the UI Thread
    - IO Bound – “Time needed is based on speed of the IO Device (I.E: Hard Drive), not CPU.”
    - CPU Bound – “Time needed is based on speed of CPU”
  - Need to perform these on a background thread…
- How do we keep this extensible
  - How do we not tightly couple components together
  - How do we keep the control flow understandable
    - How do we model it
  - How can it be improved and extended upon with ease

Handling UI Events – The Model



Handling UI Events – The Solution

- Push-Based Event Notifications
  - $EventBus$
    - Uses RxJava's $Observable$ and $Observer$ implementations
      - $Observable$ – "A 'publisher' that emits items that can be observed."
      - $Observer$ – "A 'subscriber' that observes the items emitted by the 'publisher' $Observable$."
    - Can "connect" to other $EventBus$' that send/receive to/from us
      - Each component maintains an $EventBus$
  - $EventProcessor$
    - Processes $Event$s received over an $EventBus$
    - Sends $Event$s over the $EventBus$
- Responsiveness
  - All processing is performed in a single background thread
    - A single background thread greatly reduces complexity of multithreading
  - UI updates are performed on the Swing UI Thread
    - Keeps the UI from "freezing"
- Extensibility
  - Very easy to add, modify, or even remove an $EventProcessor$
    - Simple as registering/unregistering on the $EventBus$

$Event$

- Each $Event$ is made of a $sender$, a $recipient$, a $descriptor$, and an opaque $data$ reference
  - $sender$ – Who sent this $Event$
  - $recipient$ – Who will receive this $Event$
  - $descriptor$ – What this $Event$ is
    - The meaning is up to the sender and recipient to find out.
  - $data$ – Data associated with this $Event$
    - The meaning is coupled to the $descriptor$
- Optimization (Assuming HotSpot JVM)
  - Size of 4 + 4 + 4 + 4 + 8 = 24 bytes
    - 4 References + 8 Byte Object Header
    - Compressed oops
      - Pointers (references) are 32-bits (4 bytes) in size, even on a 64-bit system
      - How – Objects are allocated on an 8-byte alignment; these 32-bit pointers are scaled to a factor of 8 and added to a 64-bit base address
  - Garbage Collection
    - Events are immutable and short-lived
      - Only used by a single $EventProcessor$
    - Contained in the “young” generation

```java
// Fields to determine WHO sent it, WHO this is for, and WHAT this event is.
public final String sender;
public final String recipient;
public final String descriptor;

// The data that is associated with this event. The actual type and how it is
// used is up to the sender and intended recipient to determine.
public final Object data;

/**
 * Create an instance of an Event. This is used over the constructor in case
 * we make some further changed and/or optimizations.
 * @param from Who are we?
 * @param to Who should see this?
 * @param what What kind of event is this?
 * @param data What is the event data?
 * @return Event encoded.
 */
public static Event of(String from, String to, String what, Object data) {
    return new Event(from, to, what, data);
}

/**
 * Ignores the passed parameter in favor of returning an empty Observable. This
 * is needed for when handling events that do not need propagation without needing
 * to interrupt the control flow with explicit checks.
 * @param ignored
 * @return
 */
public static Observable<Event> empty(Object ignored) {
    return Observable.empty();
}

private Event(String sender, String recipient, String descriptor, Object data) {
    this.sender = sender;
    this.recipient = recipient;
    this.descriptor = descriptor;
    this.data = data;
}

@Override
public String toString() {
    return "Event: {Sender: " + sender + ", Recipient: " + recipient + ", Descriptor: \"" + descriptor + "\"}";
}

public String toString(Function<Event, String> asString) {
    return "Event: {Sender: " + sender + ", Recipient: " + recipient + ", Descriptor: \"" + descriptor + "\"}";
}
```

$EventBus$

- Handling Incoming Events
  - Filter out any $Event$ not meant for us
  - Event Logging
  - Defer processing to $EventProcessor$
    - $flatMap$ allows them to return **zero or more** $Event$s through an $Observable$
    - The $Observable$ can have associated with it it's own processing
      - I.E: IO Processing, Interval or delayed $Event$s
  - Handled (by default) on the background worker thread
    - Although the $EventProcessor$ may switch back and forth between threads and schedulers
  - Error Handling
    - When an error occurs in any stage of the pipeline, such as an uncaught Exception, the handler will be invoked and processing will halt
  - Broadcast any outgoing $Event$s
    - If the $EventProcessor$ returned any, we broadcast them

```
public EventBus(EventProcessor processor, String id) {
    this.id = id;

    // Handle receiving events
    ingoingEvent
            // Only accept events if they are addressed to use
            .filter(e -> e.recipient.equals(id))
            // Log any and all events
            .doOnNext(e -> LOG.info(e.toString()))
            // Defer processing. The processor is free to go between schedulers
            // all they like, and may return zero or more Events.
            .flatMap(processor::process)
            // All events received are processed by the background worker thread.
            .subscribeOn(Globals.WORKER_THREAD)
            // If it emits any events, send them as outgoing. Any errors will cause
            // a runtime exception and termination! In the future, there will be
            // a way to recover from errors that are recoverable, but that will be
            // implemented based on need.
            .subscribe(outgoingEvent::onNext, this::onError);
}

private void onError(Throwable ex) {
    // Dump stack frame
    LOG.log(Level.SEVERE, "Error while processing event: {0}", ex.getMessage());
    LOG.severe(Stream
            .of(ex.getStackTrace())
            .map(Object::toString)
            .collect(Collectors.joining("\n")));

    // User informed of critical error before exiting.
    JOptionPane.showMessageDialog(null, "Error while processing an event!!!\n"
            + "A stacktrace has been written to the log file, 'log.txt'.\n"
            + "Please Submit this to the developer: LouisJenkinsCS@hotmail.com\n"
            + "Error Message: " + ex.getMessage(), "Critical Error", JOptionPane.ERROR_MESSAGE);
    System.exit(1);
}
```

$EventProcessor$

- Processing Events
  - Handle the $Event$ based on the $sender$
    - We already know that this $Event$ is meant for us, but now we can determine WHO sent it.
  - Handle how to process the $Event$ based on $descriptor$
    - Since we know WHO sent it, we can use the sender's constants to determine WHAT it is.
  - Process the $Event$
    - Since we know precisely WHO and WHAT, we know HOW we can process it.
  - Reactively send $Event$s
    - If it warrants it, we can send our own $Event$s, or none
      - Can return zero element with $Observable.empty()$.
- What about UI Events
  - **Note:** Only the UI Thread may interact with the UI
  - An $Event$ that requires updating the UI may be enqueued to the AWT Dispatch Thread's Event Queue
    - This is handled asynchronously with respect to our worker thread
  - UI Events are dispatched **sequentially** and in **encounter order**
    - We do not need to wait for the UI Thread to finish it's work, as they will be performed in the order they are added
      - Not a race condition

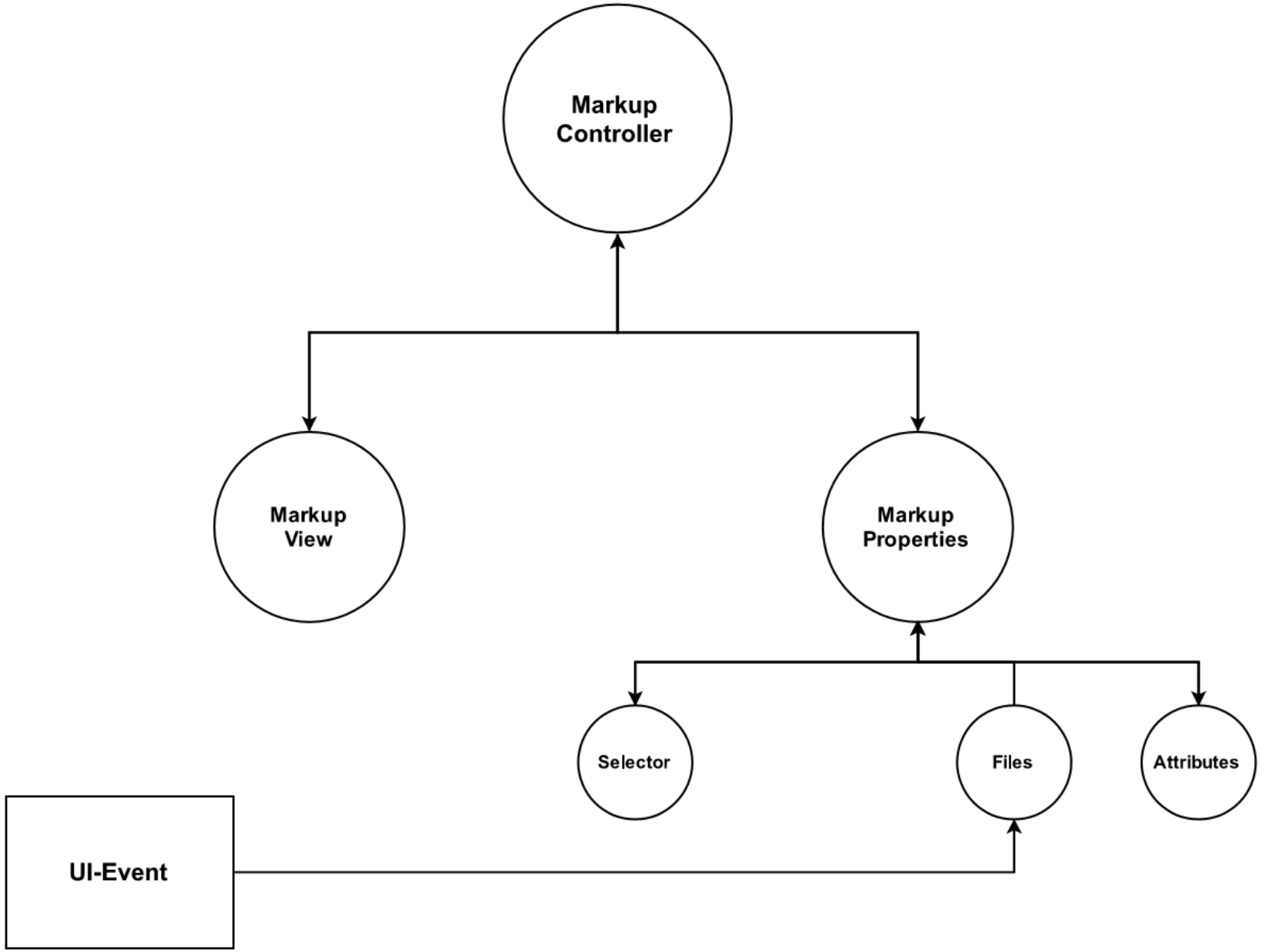
```
@Override
public Observable<Event> process(Event e) {
    switch (e.sender) {
        case Event.MARKUP_CONTROLLER:
            switch (e.descriptor) {
                case MarkupController.NEW_MARKUP:
                    return newMarkup((Markup) e.data);
                case MarkupController.DISPLAY_MARKUP:
                    return displayMarkup((Markup) e.data);
                case MarkupController.RESTORE_MARKUPS:
                    return restoreMarkups((List<Markup>) e.data);
                case MarkupController.REMOVE_MARKUP:
                    return removeMarkup((Markup) e.data);
                case MarkupController.SELECTED_ID_RESPONSE:
                    return selectedIdResponse((Markup) e.data);
                default:
                    throw new RuntimeException("Bad Custom Tag from MarkupController!");
            }
        case Event.PROPERTY_ATTRIBUTES:
            switch (e.descriptor) {
                case PropertyAttributes.TEXT_CHANGE:
                    return textChange((String) e.data);
                case PropertyAttributes.COLOR_CHANGE:
                    return colorChange((Color) e.data);
                default:
                    throw new RuntimeException("Bad Custom Tag from PropertyAttributes!");
            }
        case Event.PROPERTY_FILES:
            switch (e.descriptor) {
                case PropertyFiles.FILE_SELECTED:
                    return fileSelected((Path) e.data);
                default:
                    throw new RuntimeException("Bad Custom Tag from PropertyFiles!");
            }
        case Event.PROPERTY_SELECTOR:
            switch (e.descriptor) {
                case PropertySelector.SELECTED_ID:
                    return selectedId((String) e.data);
                default:
                    throw new RuntimeException("Bad Custom Tag from PropertySelector!");
            }
        case Event.PROPERTY_TEMPLATES:
            switch (e.descriptor) {
                case PropertyTemplates.APPLY_TEMPLATE:
                    return applyTemplate((Markup) e.data);
            }
        default:
            throw new RuntimeException("Bad Sender!");
    }
}

@Override
public EventBus getEventEngine() {
    return engine;
}
```

Event Sourcing

- Logging
  - Log files contain the exact sequence of events that occur in the system
    - Describes WHEN the event occurs, WHO sent it, WHO is the intended receiver, and WHAT the event is.
  - Makes it easier to debug problems that potential clients may face
    - Easy for me to determine where and what caused the issue by examining the log file

Example: Selecting a File

1. User Selects a new file in PropertyFiles
2. PropertyFiles notifies MarkupProperties of file name
3. MarkupProperties notifies PropertyAttributes to clear back to default state.
4. MarkupProperties notifies PropertySelector to clear it's adapter of all entries (as they are no longer valid)
5. MarkupProperties notifies MarkupController of file name
6. MarkupController checks if a previous session for this File exists.
   - If yes, goto 7, else goto 9
7. MarkupController notifies MarkupProperties of a list of Markups to restore the state of
8. MarkupProperties notifies Selector to add all entries to it's adapter (in lexicographical order)
9. MarkupController notifies MarkupView with file contents and markup highlight bounds if applicable
10. MarkupView passes a syntax highlighted version of the file contents to it's view-model and adds highlighting for all sent highlight offsets.



Conclusion

Final Words and Thoughts

Software Engineering

- A simple idea becomes that much more when production is on the line
  - Bugs, error handling, arch-specific issues, etc.
    - “That doesn’t happen when I run it on my machine”
  - Maintenance
    - The job isn’t over once it is released
      - New features need to be added
      - Bugs need fixing
- It isn’t always “fun”
  - Even if there is nothing new to learn, the obligations still remain
  - Spent 250+ hours on this
    - Lost interest about 1/5 of the way
      - Still had to do it
- Not something I want to do again
  - At least not unpaid…
  - Solidified my resolve to go to graduate school for my PhD
    - Prefer theory anyway

Final Thoughts

- Code-Glosser took a long time to make, but if people use it, then it was worth it
  - A lot of thought was put into its design
  - A lot of time was put into its implementation
- Distribution
  - https://github.com/LouisJenkinsCS/Code-Glosser/releases
- Bug Reporting and Feature Requests
  - File an issue using GitHub's bug tracker
  - https://github.com/LouisJenkinsCS/Code-Glosser/issues