

265P Week 3 Homework: UML Diagram

ScreenMarkers - [Templates](#)

Screen markers are intended to allow users to draw their own specified boxes on top of the Runescape client. They can edit the border and fill color of the boxes, the transparency, and they can move the boxes by holding the 'alt' key and dragging. This helps users know the location of items on the screen to reduce the time of intended actions.

We begin our search by using the find usages command on IntelliJ with the query "ScreenMarker". The reason for this is because we referenced the Runelite wiki to learn that the plugin we are interested in is called ScreenMarker. We see that this class specifies properties of what makes up a **ScreenMarker** object, but it ends there. Because it stops there, we go back to the find usages feature and visit the **ScreenMarkerPanel** class, and further investigation tells us that this class is a JPanel representation of the marker. We use cmd-f and input "screenmarker" and see that there is a property called plugin that is of type **ScreenMarkerPlugin** as well as a property called marker of type **ScreenMarkerOverlay**. Both classes are then examined. The ScreenMarkerOverlay class contains the **ScreenMarkerRenderable** object to render the graphics. The ScreenMarkerPlugin looks to be the centerpoint of the feature we are looking at because it references properties from classes **ScreenMarkerCreationOverlay**, **ScreenMarkerMouseListener**, **ScreenMarkerPluginPanel** (which in turn uses **ScreenMarkerCreationPanel**), the ScreenMarker object itself, and it also keeps a list of objects of type ScreenMarkerOverlay.

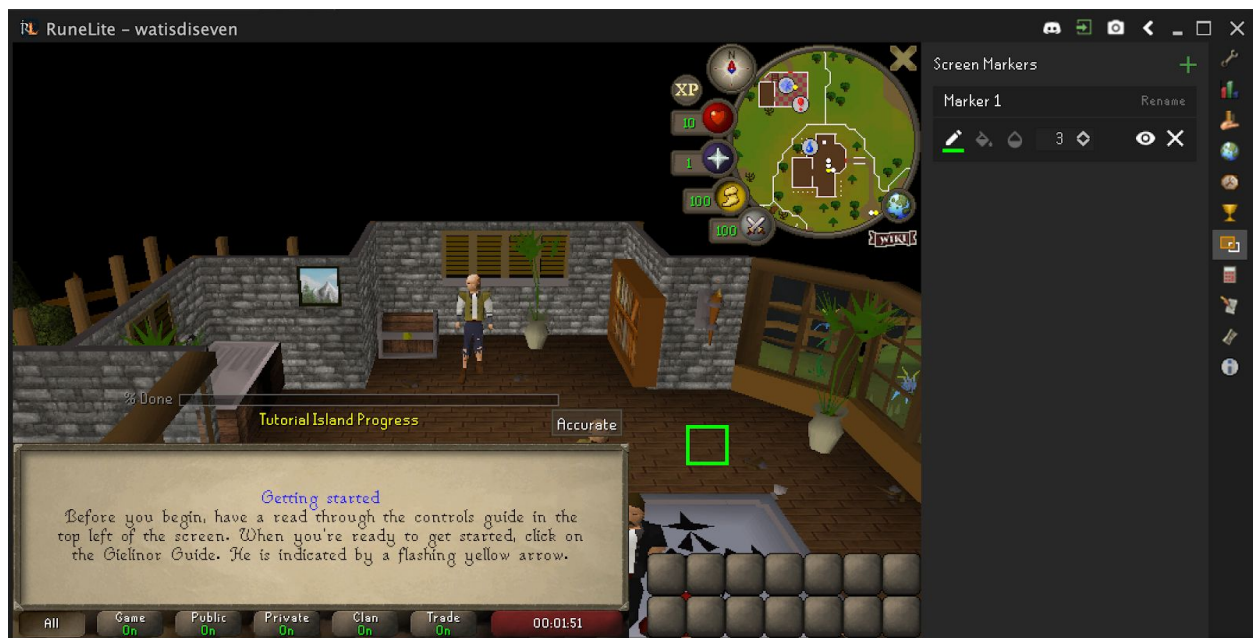
ScreenMarkerPanel.java:

```
private final ScreenMarkerPlugin plugin;  
private final ScreenMarkerOverlay marker;
```

ScreenMarkerPlugin.java

```
@Getter  
private final List<ScreenMarkerOverlay> screenMarkers = new ArrayList<>();  
  
@Inject  
private ConfigManager configManager;  
  
@Inject  
private MouseManager mouseManager;  
  
@Inject  
private ClientToolbar clientToolbar;  
  
@Inject  
private OverlayManager overlayManager;  
  
@Inject  
private ScreenMarkerCreationOverlay overlay;  
  
@Getter  
@Inject  
private ColorPickerManager colorPickerManager;  
  
private ScreenMarkerMouseListener mouseListener;  
private ScreenMarkerPluginPanel pluginPanel;  
private NavigationButton navigationButton;  
  
@Getter(AccessLevel.PACKAGE)  
private ScreenMarker currentMarker;
```

With this, we have traversed all the classes that is associated with the Screenmarker. Each of these classes supports the creation of a customizable (via width and color) box that the user can control on the client in order for them to mark specific positions on their screen. ScreenMarkerPlugin.java in particular is the entry point to which the plugin can be activated and shutdown, specifically in the startUp() and shutDown() methods. The mouse listener is also instantiated in startUp() so that click events the user makes when the plugin has been activated can be observed. The click event of the mouse triggers a call to startCreation() while a click on the confirm label prepared in ScreenMarkerCreationPanel.java triggers a call to finishCreation(). Looking at these classes, we get a glimpse of the basic flow of how a screen marker can be made by the user in code, but the rest of the classes support this feature, providing a mechanism to which the screen marker can be adjusted by position, color, deletion, and so on. The following shows a simple screen marker in action.



Metronome - [Templates](#)

The in-game metronome is a ticking mechanism. For a player to be efficient with gameplay, the game collects all inputs within 0.6 seconds then processes them all at once. The metronome will make a sound every ticking period to alert the player. The player can change the ticking period and enable/disable ticking. The player can exploit this idea to optimize his/her gameplay strategy as necessary.

By using **grep**, we located all case-insensitive usages of the word **metronome**. Most usages reside in MetronomePlugin.java and MetronomePluginConfiguration.java. Usually, remaining usage for a Java project is in git files or in binary files. Maybe the other files have something to say.

Below are the usages of the word **metronome** in the program.

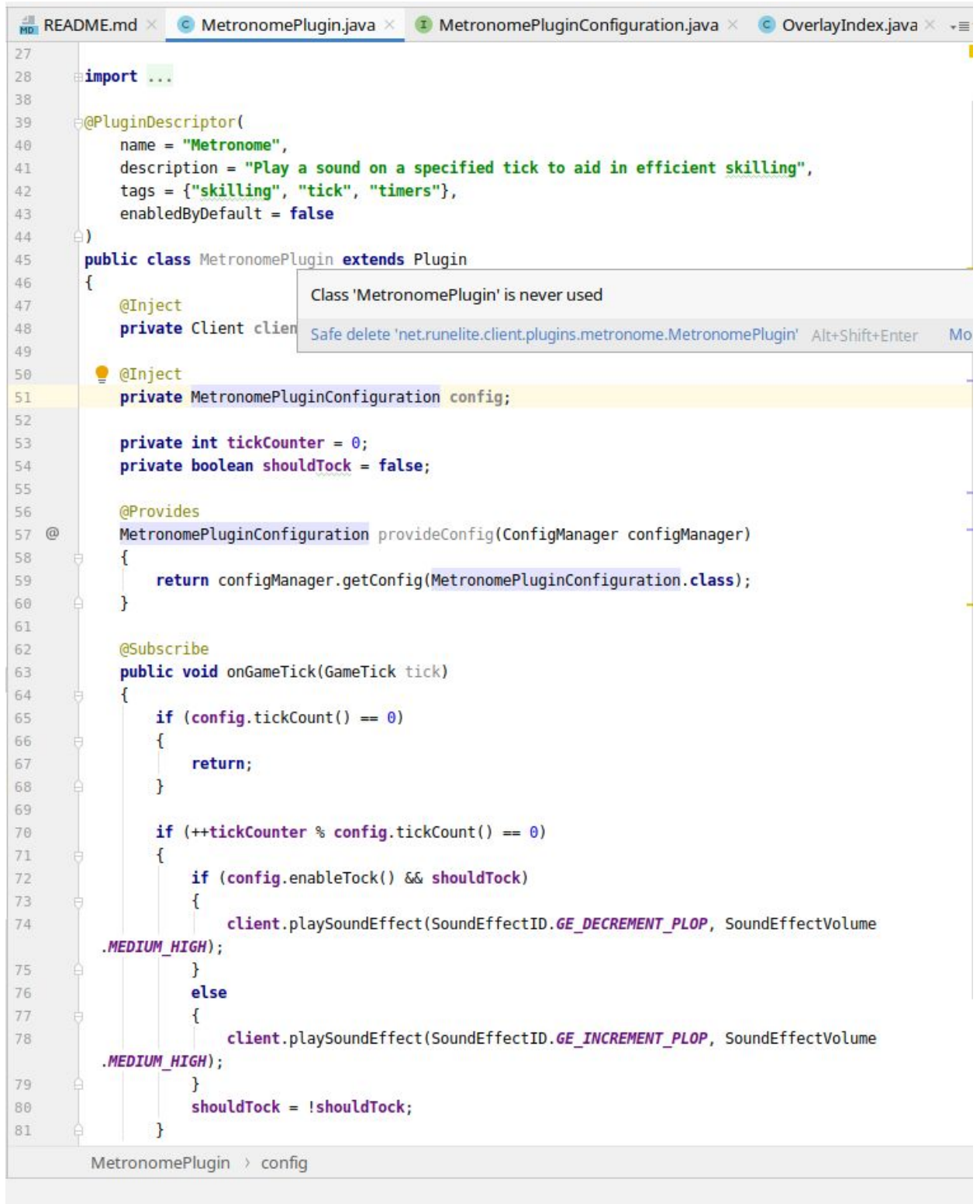
```
(base) debba@dieWeisseMaschine:~/Downloads/git/runelite$ grep -Pir metronome . --color=always | grep -v "Binary"
./runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst:net/runelite/client/plugins/metronome/MetronomePluginConfiguration.class
./runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst:net/runelite/client/plugins/metronome/MetronomePlugin.class
./runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst:/home/debba/Downloads/git/runelite/runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePluginConfiguration.java
./runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst:/home/debba/Downloads/git/runelite/runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePluginConfiguration.java:package net.runelite.client.plugins.metronome;
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePluginConfiguration.java:@ConfigGroup("metronome")
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePluginConfiguration.java:public interface MetronomePluginConfiguration extends Config
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:package net.runelite.client.plugins.metronome;
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:    name = "Metronome",
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:public class MetronomePlugin extends Plugin
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:    private MetronomePluginConfiguration
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:    MetronomePluginConfiguration provideC
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:    onfig(ConfigManager configManager)
./runelite-client/src/main/java/net/runelite/client/plugins/metronome/MetronomePlugin.java:        return configManager.getConfig(MetronomePluginConfiguration.class);
(base) debba@dieWeisseMaschine:~/Downloads/git/runelite$
```

Strangely enough `MetronomePlugin` uses `MetronomePluginConfiguration` while nothing else uses either class. Yet the plugin appears in the program settings.

Below is a screenshot of the Metronome option in the program.



IntelliJ itself indicates that nothing uses MetronomePlugin.
Below IntelliJ greys out the class name.



```
27
28 import ...
38
39 @PluginDescriptor(
40     name = "Metronome",
41     description = "Play a sound on a specified tick to aid in efficient skilling",
42     tags = {"skilling", "tick", "timers"},
43     enabledByDefault = false
44 )
45 public class MetronomePlugin extends Plugin
46 {
47     @Inject
48     private Client client
49
50     @Inject
51     private MetronomePluginConfiguration config;
52
53     private int tickCounter = 0;
54     private boolean shouldTock = false;
55
56     @Provides
57     MetronomePluginConfiguration provideConfig(ConfigManager configManager)
58     {
59         return configManager.getConfig(MetronomePluginConfiguration.class);
60     }
61
62     @Subscribe
63     public void onGameTick(GameTick tick)
64     {
65         if (config.tickCount() == 0)
66         {
67             return;
68         }
69
70         if (++tickCounter % config.tickCount() == 0)
71         {
72             if (config.enableTock() && shouldTock)
73             {
74                 client.playSoundEffect(SoundEffectID.GE_DECREMENT_PLOP, SoundEffectVolume
75                     .MEDIUM_HIGH);
76             }
77             else
78             {
79                 client.playSoundEffect(SoundEffectID.GE_INCREMENT_PLOP, SoundEffectVolume
80                     .MEDIUM_HIGH);
81             }
82             shouldTock = !shouldTock;
83         }
84     }
85 }
```

Class 'MetronomePlugin' is never used
Safe delete 'net.runelite.client.plugins.metronome.MetronomePlugin' Alt+Shift+Enter Mo

MetronomePlugin > config

Fortunately, the files `inputFiles.lst` and `createdFiles.lst` contain entries for both Metronome classes, the former having `.java` files and the latter having `.class` files. This is the most likely way to reference the `MetronomePlugin`; rather than hard code it, have a list of plugins and other things to centralize references and make life easier. This may be a deliberate design choice by the developers, or it may be because the project is a Maven project and part of these files' pathnames have maven in their names. It's likely a developers' choice because IntelliJ has Maven support built-in and doesn't detect the `MetronomePlugin` class.

The `simpleUML` plugin works wonders - apparently it sorts the classes based on some metric; all of the plugins are in one row. So after some pixel peeping, we found the `MetronomePlugin` and traced the corresponding `MetronomePluginConfiguration`. This sorting scheme is the only clue we have so far to trace the right classes.

Template - ScreenMarker (Where we've been)

Folder	File	Method	Relevant?	Relevant how?	Confidence
net/runelite/client/plugins/screenmarkers/	ScreenMarker.java	properties/fields	yes	describes properties that make up a screen marker and how it will appear on the ui	medium
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerPanel.java	ScreenMarkerPanel(ScreenMarkerPlugin, ScreenMarkerOverlay) [constructor]	yes	initializes the screen marker plugin and the screen marker overlay	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerOverlay.java	render()	yes	render() method references ScreenMarkerRenderable object and calls methods from that class	medium
net/runelite/client/plugins/screenmarkers/	ScreenMarkerRenderable	class declaration	yes	render() method deals with rendering the graphics for the screen marker	medium
net/runelite/client/plugins/screenmarkers/	ScreenMarkerPlugin.java	properties/fields	yes	entire class contains all relevant fields that references other associated classes to the screen marker	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerCreationOverlay.java	ScreenMarkerCreationOverlay(ScreenMarkerPlugin) [constructor] render()	yes	The field references the screen marker plugin and the render() method references the current screen marker to render its graphics	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerMouseListener.java	ScreenMarkerMouseListener(ScreenMarkerPlugin) [constructor] mousePressed(), mouseReleased(), mouseDragged()	yes	constructor ties the plugin reference to its field, and the mousePressed(), mouseReleased(), and mouseDragged() methods calls methods from the ScreenMarkerPlugin class	medium
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerPluginPanel.java	ScreenMarkerPluginPanel(ScreenMarkerPlugin) [constructor]	yes	constructor ties the plugin reference to its field	medium
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerCreationPanel.java	ScreenMarkerCreationPanel(ScreenMarkerPlugin) [constructor]	yes	Object was a field in ScreenMarkerPluginPanel. Deals with ui components and icons in support of what is shown to the user when they want to use the screen marker.	medium

Template - ScreenMarker (Where to go)

Folder	File	Method	Why?	Priority
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerPanel.java	ScreenMarkerPanel(ScreenMarkerPlugin, ScreenMarkerOverlay) [constructor]	Looked at both the constructor, saw that it initialized relevant properties with the values passed in from the parameters of the constructor.	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerOverlay.java	render()	render() references the field of type ScreenMarkerRenderable, and calls methods from that class	medium
net/runelite/client/plugins/screenmarkers/	ScreenMarkerRenderable.java	class declaration	Deals with rendering the graphics for the screen marker	medium
net/runelite/client/plugins/screenmarkers/	ScreenMarkerPlugin.java	properties/fields	the entire class contains all relevant fields that references other associated classes.	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerCreationOverlay.java	ScreenMarkerCreationOverlay(ScreenMarkerPlugin) [constructor] render()	The field references the screen marker plugin and the render() method references the current screen marker to render its graphics	high
net/runelite/client/plugins/screenmarkers/	ScreenMarkerMouseListener.java	ScreenMarkerMouseListener(ScreenMarkerPlugin) [constructor] mousePressed(), mouseReleased(), mouseDragged()	constructor ties the plugin reference to its field, and the mousePressed(), mouseReleased(), and mouseDragged() methods calls methods from the ScreenMarkerPlugin class	medium
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerPluginPanel.java	ScreenMarkerPluginPanel(ScreenMarkerPlugin) [constructor]	constructor ties the plugin reference to its field	medium
net/runelite/client/plugins/screenmarkers/ui/	ScreenMarkerCreationPanel.java	ScreenMarkerCreationPanel(ScreenMarkerPlugin) [constructor]	Object was a field in ScreenMarkerPluginPanel. Deals with ui components and icons in support of what is shown to the user when they want to use the screen marker.	medium

Template - Metronome (Where we've been)

Folder	File	Method	Relevant?	Relevant how?	Confidence	Notes
runelite-client/src/main/java/net/runelite/client/plugins/metronome/	MetronomePlugin.java	provideConfig()	Yes	Class definition; shows up in program despite not being in Java code; some other type of file references it	Sure	project doesn't directly use it
runelite-client/src/main/java/net/runelite/client/plugins/metronome/	MetronomePluginConfiguration.java	Multiple	Yes	Linked with MetronomePlugin; otherwise no other usage	Sure	besides MetronomePlugin, nothing else uses it directly (as of this query)
runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile	createdFiles.lst	None	Yes	Found references to Metronome*.class	Very sure	
runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile	inputFiles.lst	None	Yes	Found references to Metronome*.java	Very sure	Likely this is the list of Java files from which createdFiles.lst derives

Template - Metronome (Where to go)

Folder	File	Method	Why?	Priority	Notes
runelite-client/src/main/java/net/runelite/client/plugins/metronome/	MetronomePlugin.java	Any method regarding Metronomes	Most obvious candidate for Metronome class	High	No meaningful data here, except for the fact that nothing uses this class
runelite-client/src/main/java/net/runelite/client/plugins/metronome/	MetronomePluginConfiguration.java	None	Class name suggests relevance to MetronomePlugin, which uses this class	High	Back to square one, nothing useful
runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile	createdFiles.lst	None	grep showed that this file contained something about metronomes	Medium	Lots of .class files, Metronome*.class included
runelite-client/target/maven-status/maven-compiler-plugin/compile/default-compile	inputFiles.lst	None	grep showed that this file contained something about metronomes	Medium	Lots of .java files, Metronome*.java included. This seems to be the answer we seek