# Project #3 Map Interface

*Computer Science 2334*

*Fall 2007*

## *Note:*

As discussed in class on Monday, Oct. 22, there has been a change in the mapping components for Project #3. We will be using Google Maps, rather than Yahoo! Maps for our map generation. This change was necessitated for two reasons: 1. It appears to us that the mapping service provided by Yahoo! does not match Yahoo!'s own specification. 2. The debugging information provided by Yahoo! was not very helpful (e.g., the same error message, complaining about a particular error in the XML, was returned regardless of the XML sent to Yahoo!).

While this change has resulted in significantly more work for our TA Mark Woehrer who is providing the primary mapping components for your project, it should be no more work for you to integrate your code with the code provided to interface with Google than it would have been for you to integrate your code with code provided to interface with Yahoo! However, while there should be no **more** work in the present scenario, it will be slightly **different** work. In particular, your code will be generating JavaScript rather than XML and will be popping up an external web browser to display the map, rather than using placing the map within a Java GUI window. Because of these changes, slightly revised versions of the Project #3 and project #3 GUI documents have been placed on the class web pages.

## *Background:*

The Google Maps Application Programming Interface (API) uses JavaScript to create useful, interactive maps within graphical web browser windows. (See: http://www.google.com/apis/maps/index.html and related pages.) You are undoubtedly already familiar with at least one web browser – how else would you have accessed this document from the class web pages? Most web browsers (e.g., Firefox, Opera, Epiphany, Konqueror, OmniWeb, Safari, etc.) are graphical, though some (e.g., Lynx) are text-based. For this assignment, you will need to use a graphical browser to display the map and have the data plotted on it using JavaScript.

JavaScript is a programming language used primarily for client-side scripting on the web. This means that a web browser will load a piece of JavaScript source code – typically from a web server, though in this assignment from a file – which it can then interpret and run. Because the JavaScript is normally coming from a web server to your browser (which is said to be the client which the server is serving), we call this "client-side." Because it is the source code that is loaded by the browser, we call it "scripting."

Note that this is different than the way browsers handle Java. With Java, the source code is compiled into bytecode before being handed to the browser. The browser then uses a Java Runtime Environment as a virtual machine to run the Java bytecode.

Java and JavaScript share some syntax but are not closely-related languages. They are both used in browsers to add dynamic content, which is why Sun (which developed Java) and Netscape (which developed JavaScript) saw a marketing opportunity and agreed on the similar names.

## *Description:*

To interact with the Google Maps API, we'll embed our JavaScript in a small HTML wrapper. This is because the web browser is expecting the file it loads to be a web page. Inside the HTML will be information to tell the browser that we are using JavaScript (“`<script ... >`”), to fetch base JavaScript from Google (src='`http://maps.google.com/maps?file=api&amp;v=2`'), and to implement our own JavaScript functions. Our JavaScript will tell the browser to create a new map to go in the canvas, create a new center point for the map and center the map there, put a marker at the latitude/longitude coordinates of the zip code selected, and put a *polyline* (a connected series of line segments) on the map for each storm to be plotted, where each point on the polyline is a data point for that storm.

A template for this JavaScript has been provided to you in Mark's code. In this code, the JavaScript is a single long string he called “`content`” that will be written to a file, which the browser will then open. When the browser opens the file, it will run the JavaScript (assuming the browsers is JavaScript capable and has JavaScript processing turned on).

Mark has also provided simple examples for how to substitute into the JavaScript `content` string and how to start the browser. The substitution examples are rather like stub-code in that they allow you to see how parts of the code will function but do not implement all the functionality that you'll want in your code. In order to complete your assignment, your will need to create methods that can substitute into the `content` string the actual data selected by the user from the GUI you are creating. For the center point, you may use the latitude/longitude coordinates of the zip code selected. In addition, you'll need to modify Mark's code such that multiple polylines can be overlaid on the map, rather than just one.

## *Extra Credit:*

There are many possibilities for extra credit on the mapping side of the assignment. The key is to make them useful, explain why they are useful, and implement them well. For example, the center point probably shouldn't be the latitude/longitude coordinates of the zip code selected. A better solution would probably to make the center point the midpoint in both latitude and longitude between the extreme values in the data (including the zip code coordinates). As another example, it would be good to use different colors for the different polylines in a single map. Alternately or in addition, you could add different colored markers for different intensities of storms or associate additional data relating to each point with its marker so that it appears when the user “mouses over” the point. How many points you may get for any addition depends on how useful it is and how well it is explained and implemented.