**CS 145: Homework 2**

Was Here

# 1 Description

Your primary goals in this homework are to learn about conditional statements, loops, and file I/O. You will do this in context of writing a tool for converting a plain text geographical biography into a file that can explored with Google Earth or Google Maps.

You will write two new classes for this homework: `KeyholeIO` and `WasHere`. Place these two classes in a package named `hw2`.

# 2 Background

Google Earth marks locations on its virtual globe with *placemarks*. The simplest placemark—a pin placemark—points to an exact latitude-longitude. One can also define region placemarks, which cover broader areas, and line placemarks, which mark things like borders and really long walls. Placemarks are typically positioned manually or automatically placed by Google Earth itself, but you can open a file of placemarks. Creating these files is what you will do in this homework.

The file is structured using the Keyhole Markup Language (KML). Here's an example of a file containing exactly one of each kind of placemark:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Style id="seethrough"><PolyStyle><color>7fffffff</color></PolyStyle></Style>

<!-- A pin placemark -->
<Placemark>
<name>NAME</name>
<description>DESCRIPTION</description>
<Point>
<coordinates>LAT0,LON0</coordinates>
</Point>
</Placemark>

<!-- A line placemark -->
<Placemark>
<name>NAME</name>
<description>DESCRIPTION</description>
<LineString>
<tessellate>1</tessellate>
<coordinates>LAT0,LON0 LAT1,LON1 LAT2,LON2 ...</coordinates>
```

```
</LineString>
</Placemark>

<!-- A region placemark -->
<Placemark>
<name>NAME</name>
<description>DESCRIPTION</description>
<styleUrl>#seethrough</styleUrl>
<Polygon>
<outerBoundaryIs>
<LinearRing>
<coordinates>LAT0,LON0 LAT1,LON1 LAT2,LON2 ...</coordinates>
</LinearRing>
</outerBoundaryIs>
</Polygon>
</Placemark>

</Document>
</kml>
```

Words in all capital letters need to be replaced with real content. For a fully working example, see your instructor's geographical biography, which demonstrates all three placemarks.

# 3 Requirements

In this homework, you will convert a non-KML geographic file into a KML file. Your solution will be broken down in two classes: `KeyholeIO` and `WasHere`.

## 3.1 KeyholeIO

Your `KeyholeIO` class handles the output of placemarks in KML format. It must meet this specification:

- Has a method `openKML` that accepts a `PrintWriter` as its sole parameter. It writes the first four lines of the above example KML file to the `PrintWriter` object. Test this method before moving on.
- Has a method `closeKML` that accepts a `PrintWriter` as its sole parameter. It writes the last two lines of the above example KML file to the `PrintWriter` object. Test this method before moving on.
- Has a method `writePin` that takes the following parameters, in this order: a `PrintWriter`, a `String` for the place name, a `String` for the place description, and a `String` for the

comma-separated longitude-latitude. It writes the pin placemark to the `PrintWriter` as seen in the example. Test this method before moving on.

- Has a method `writeLine` that takes the following parameters, in this order: a `PrintWriter`, a `String` for the place name, a `String` for the place description, and a `String` for the space-separated sequence of comma-separated longitude-latitude pairs. It writes the line placemark to the `PrintWriter` as seen in the example. Test this method before moving on.

- Has a method `writeRegion` that takes the following parameters, in this order: a `PrintWriter`, a `String` for the place name, a `String` for the place description, and a `String` for the space-separated sequence of comma-separated longitude-latitude pairs. It writes the region placemark to the `PrintWriter` as seen in the example. Test this method before moving on.

- Has a method `writePlacemark` that takes the following parameters, in this order: a `PrintWriter`, a `String` for the place name, a `String` for the place description, and a `String` containing one or more comma-separated longitude-latitude pairs. If only one pair is present, it writes a pin placemark to the `PrintWriter` object. If the last pair is identical to the first, it writes a region placemark, but does not write the repeated longitude-latitude to the `PrintWriter`. Otherwise it writes a line placemark. (Call upon your existing methods.) Test this method before moving on.

## 3.2 WasHere

Your `WasHere` class handles the conversion of the geographical biography from an unstructured format to the KML format. It must meet this specification:

- Has a method `convert` that takes in two parameters. The first is a `String` parameter for the path to a geographic biography file containing placemark information in a plain-text, non-KML format. The file has this format:

  ```
  name1
  description1
  coords1
  name2
  description2
  coords2
  ...
  ```

  See your instructor's biography for a complete example. You may assume the file is well-formed and has a number of lines that is a multiple of three. The coordinate lines may have one longitude-latitude pair or they may have several separated by spaces. The second parameter is a `String` path to the KML file to write.

This method opens the biography file and writes it out to the KML file. See the KML version of your instructor's biography for reference. Opening files is an operation that may fail, remember. Mark your method with `throws FileNotFoundException`, as discussed in lecture.

- Has a `main` method that calls `convert` and passes it the paths to your biography file and KML file. If you'd like to include your biography in your package and you biography is named "mybio.txt", pass the path "src/hw2/mybio.txt".

Please share testing code on Piazza. You are encouraged to write your own biography and resulting KML files on Piazza. You may vote exactly once for your favorite biography (not your own, however) with a "+1" reply in the followup discussion. The anthem of the student with the most votes will be trumpeted across many nations.

# 4 Files

- SpecChecker: twodee.org/teaching/cs145/2012A/homework/speccheck_hw2.jar
- Example input: twodee.org/teaching/cs145/2012A/homework/washere_eg.txt
- Example output: twodee.org/teaching/cs145/2012A/homework/washere_eg.kml

# 5 Other expectations and submission

See homework 1 and preassignment 1 to see what else you are graded on and how to submit your work.