AppDynamics Coding Assignment

Many AppDynamics customers run websites which are popular all over the world. For these customers, AppDynamics provides capability to monitor traffic from different geographic regions, in real time. The user interface for this monitoring is shown below.

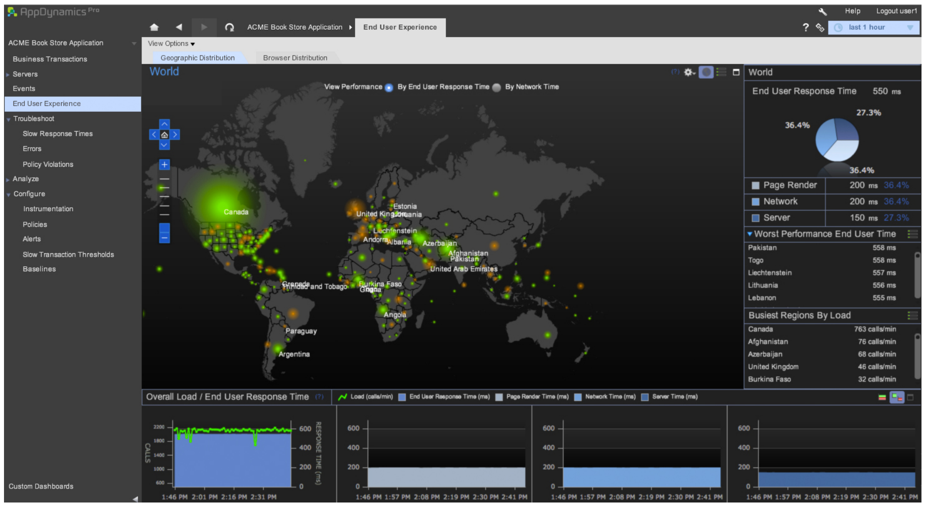
AppDynamics Pro
ACME Book Store Application
End User Experience
Geographic Distribution
Browser Distribution
World
View Performance By End User Response Time By Network Time
End User Response Time 550 ms
Page Render 200 ms 36.4%
Network 200 ms 36.4%
Server 150 ms 27.3%
Worst Performance End User Time
Busiest Regions By Load
Overall Load / End User Response Time

Figure 1: End User Monitoring Screen

As a customer's web application gets hits from different geographic regions, the corresponding area of the map lights up in the AppDynamics screen above. The color in which an area lights up represents the time the user had to wait to get a response from the customer website - normal response times in green, and slower ones in various shades of red.

This assignment is modeled around designing such a feature.

```
/**
 * A user interface for the map of the world. A map contains many regions.
 */
interface MapView {

    /**
     * Shade the given region of the map, with the appropriate color based on the
     * given UserExperience.
     */
    public void lightUp(Region region, UserExperience userExperience);
}
```

```
/**
 * Represents a region in a map.
 */
interface Region {
    String getName();
}

/**
 * Represents the responsiveness of the website.
 * This can be used to determine the color in which a hit has to be shaded.
 */
enum UserExperience { NORMAL, SLOW, VERY_SLOW, STALLED }

/**
 * An implementation we use for now; it simply prints to console.
 * A more realistic implementation will use some GUI toolkit to render the map
 * and shade the appropriate region with the appropriate color.
 */
class MapViewImpl implements MapView {
    public void lightUp(Region region, UserExperience userExperience) {
        System.out.println(new Date() + ": show " +
                           region.getName() + " in color for " + userExperience);
    }
}

/**
 * A controller which decides which regions of the map should light up when.
 */
public class MapController {

    /**
     * Represents a range of some ordered type.
     */
    public static class Range<T> {
        public final T from;
        public final T until;

        public Range(T from, T until) {
            this.from = from;
            this.until = until;
        }
    }

    /**
     * @param regions mapping of ip address ranges to regions on the map.
     *                The ranges include both the from and until ip addresses.
     *
     * @param userExperiences mapping of response time (in milliseconds) ranges
     *                        to user experience categories. The ranges include the
     *                        from time, but not the until time.
     *
     *
     * @param view the MapView which needs to be updated.
     */
    public MapController(Map<Range<Inet4Address>, Region> regions,
                         Map<Range<Long>, UserExperience> userExperiences,
                         MapView view) {
        // TODO
    }

    /**
     * Update the view to record a hit.
     *
     * @param ipAddress an ip address string
     * @param responseTime the response time, in milli second, experienced by the user
     *        with the given ip address
     */
    public void hit(String ipAddress, long responseTime) {
        // TODO
    }
}
```

Your assignment is to implement `MapController` along with appropriate unit tests. The above code is only intended to get you started on an implementation. You are free to split the code into multiple files, classes or methods - just as you would do in a professional setting. Since we expect to support fine-grained regions, your solution should scale well when the number of ip address ranges increases.

The input might look like the following:

```
10.20.10.1,10.20.10.255,CALIFORNIA
10.20.20.1,10.20.20.255,NEVADA
10.20.30.1,10.20.30.255,ILLINOIS
```

i.e. a comma-separated file where each line contains two ip addresses and the name of a region.

The input might also look like:

```
10.20.10.1,10.20.10.255,CALIFORNIA
10.20.20.1,10.20.20.255,NEVADA
```

```
10.20.30.1,10.20.30.255,ILLINOIS
10.20.40.1,10.20.40.20,WASHINGTON
10.20.50.1,10.20.50.255,NEW YORK
10.20.60.1,10.20.60.255,GEORGIA
10.20.70.1,10.20.70.255,ALASKA
10.20.40.10,10.20.40.255,SEATTLE
10.20.90.1,10.20.90.255,NEBRASKA
10.20.100.1,10.20.100.255,TEXAS
```

Please deliver your solution in an archive format of your choice, including all project files. Additional points awarded if we can simply unfold this archive into a working project in Eclipse or IntelliJ IDEA ;) Your solution will be reviewed by the engineers you would be working with if you joined AppDynamics. We are interested in seeing your real-world design, coding and testing skills.