CMSC 420:Fall 2019 Dave Mount

Programming Assignment 0: MeeshQuest Basics (Preliminary)

Handed out: Tue, Sep 24. Due: **Thu, Oct 3, 11:59pm**. (See submission instructions below for late policy.)

In this assignment you will input an XML document containing a sequence of commands, execute these commands, and generate an XML document that summarizes the results. We recommend that you refer to the document "MeeshQuest Getting Started," the "MeeshQuest-Skeleton" code, and the document "Processing XML Files for the Programming Project," all of which can be found on the 420 class handouts page:

http://www.cs.umd.edu/class/fall2019/cmsc420-0201/handouts.html

For this initial part of the project, you will not need to implement any data structures. The purpose is mostly to verify that you can process input commands from an XML file, create internal objects (which you can store using Java's built-in data structures), and generating a sorted output presented in an XML file
The output consists of an XML document whose root element is "`<results>`". Each processed command will add an element to the output document, as described below. Your program will process the following commands:

Create City: This command has the following form:

```
<createCity name="Annapolis" y="14" x="12" radius="15" color="red"/>
```

The city's name is a string the x and y coordinates are integers, the radius is a nonnegative integer, and the color is among the allowed colors (e.g., `red`, `green`, `blue`, ...) listed in the schema file `part1in.xsd`, which is provided with the MeeshQuest skeleton code.

This command creates a city object with the specified name, coordinates, radius, and color. All of these attributes (`name`, `x`, `y`, `radius`, and `color`) will be given, but the order in which they appear may vary. Also note that all commands and attributes are case-sensitive.

Such an object can be successfully created if (1) its name is unique (i.e., there is not already another city with the same name), and (2) its coordinates are unique (i.e., there is not already another city with the same (x, y) coordinates). For this initial assignment, you may assume that there the input is valid, and so *no error checking is needed.* (This will be changed in the next version of the project.)

Given this command, your program will generate an XML element in your output document that echoes back all the pertinent information. It has the following form:

```
<success>
    <command name="createCity"/>
    <parameters>
        <name value="Annapolis"/>
        <x value="12"/>
        <y value="14"/>
```

```
            <radius value="15"/>
            <color value="red"/>
        </parameters>
        <output/>
    </success>
```

The output parameters must be given *in the order shown above* (that is, `name`, `x`, `y`, `radius`, and `color`), irrespective of the input order.

List Cities: This command has the following form:

```
    <listCities sortBy="name"/>
```

where the "`sortBy`" attribute can be either "`name`" or "`coordnate`". This lists the cities that exist so far. In the former case, cities are listed in ascending alphabetical order by the city name (e.g., using `java.lang.String.compareTo()`), and in the latter case they are sorted lexicographically according to their (x, y)-coordinates. (That is, cities are first sorted by their x-coordinates, and cities having the same x-coordinates are then sorted by their y-coordinates.) Each city is listed with all of its attributes. The order of the attributes is not significant.

```
    <success>
        <command name="listCities"/>
        <parameters>
            <sortBy value="name"/>
        </parameters>
        <output>
            <cityList>
                <city name="Annapolis" x="12" y="14" color="red" radius="15"/>
                <city name="Derwood" x="19" y="20" color="green" radius="40"/>
            </cityList>
        </output>
    </success>
```

Sample Input/Output: Here is a sample input. It will be input and parsed by the function `XmlUtility.validateNoNamespace()`, which we have provided you in `cmsc420util.jar`:

```
    <commands
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="part1in.xsd"
      spatialWidth="512"
      spatialHeight="512">
     <createCity name="Baltimore" y="39" x="76" radius="10" color="green"/>
     <createCity name="Chicago" x="87" y="41" radius="15" color="blue"/>
     <listCities sortBy="coordinate"/>
    </commands>
```

And the resulting sample output as generated by the utility `XmlUtility.print()`, which we have provided to you in `cmsc420util.jar`:

```
    <?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <results>
      <success>
        <command name="createCity"/>
        <parameters>
          <name value="Baltimore"/>
          <x value="76"/>
          <y value="39"/>
          <radius value="10"/>
          <color value="green"/>
        </parameters>
        <output/>
      </success>
      <success>
        <command name="createCity"/>
        <parameters>
          <name value="Chicago"/>
          <x value="87"/>
          <y value="41"/>
          <radius value="15"/>
          <color value="blue"/>
        </parameters>
        <output/>
      </success>
      <success>
        <command name="listCities"/>
        <parameters>
          <sortBy value="coordinate"/>
        </parameters>
        <output>
          <cityList>
            <city color="green" name="Baltimore" radius="10" x="76" y="39"/>
            <city color="blue" name="Chicago" radius="15" x="87" y="41"/>
          </cityList>
        </output>
      </success>
    </results>
```

Submission Instructions and Late Policy: Submit your program through the submit server

https://submit.cs.umd.edu/fall2019/

Here is the late policy:

Up to 6 hours late:	5% of total
Up to 24 hours late:	10% of the total
For each additional 24 hours late:	20% of the total