CS 1101 – Introduction to Computer Science

Spring 2022

Lab 11 – Classes and Objects

Due Date: Monday, April 25, end of the day (11:59pm).

Objective: The goal of this assignment is to help you get familiar with **classes** and **object.**

Assignment:

Create a City class that stores information about a city. This class should store the following information:

Name - name of the city
State- name of the state where the city is
Size – Total area/size of the city (in square mile, can have a decimal value)
Population – population of the city
Elevation – The elevation of the city in feet (no decimal points)

The City class should include the following methods:

- *Constructors*
  - A no-arg constructor
  - City (String na) //a constructor for the class that takes the name of the city and sets the class variable called name with na.
  - City (String na, String st, double si, int pop, int elev) //another constructor that takes the name of the city (na), the state of the city (st), the population of the city (pop), the size of the city (si), and the elevation (elev). The constructor should set the proper class variables with these values.
- void getType() which prints the type of the city. Type is determined based on the population:

  City type – Condition
  - *City* – population between 500K to 1 million
  - *Medium City* – population between 1 million to 5 million
  - *Large City* – population between 5 million to 10 million
  - *Mega City* – population more than 10 million
- Getter/accessor method for each class variable
- Setter/mutator method for each class variable

Create a separate **Runner** class.

- **This class will contain the main method**.
- Within the main method, create an array of City objects called *listOfCities*.
  - The size of the array is 20.
- The class should have the following methods:

- **Print**: a method that takes the array (*listOfCities*) as parameter and prints the name, state, size, population, and elevation of each city in the array. The method does not return anything.
- **Search by city**: a method that takes the city array (*listOfCities*), and a name of a city (to search for) as parameter. The method prints all the information of the city if the city is in the *listOfCities* array and calls the method getType from class City to print the type of city (city, medium city, large city, or mega city). If not, print "Given city is not in the array".
- **Search by state**: a method that takes the city array (*listOfCities*), and a name of a state (to search for) as parameter. The method prints all the information of the cities in that from the *listOfCities* array. If the array does not contain any city from a given state, print "Given city is not in the array".
- **Max population**: a method that takes the city array (*listOfCities*) as parameter. The method prints all the information of the city with the largest population. The method does not return anything.
- **Highest elevation**: a method that takes the city array (*listOfCities*) as parameter. The method prints all the information of the city with the highest elevation. The method does not return anything.

- This class should have a menu with the following functionalities that uses the above-mentioned methods. The menu may look like this:

```
Welcome to the list of cities!
Select an option:
1. Show all the cities in the list
2. Look up a city by name
3. Look up cities by a state
4. Most populated city
5. Highest city
6. Exit
```

Note: You are encouraged to hardcode the information of the city objects in the main method of the **Runner** class, so that the program is not taking City's information from the user. A set of sample cities are given below **(feel free to add more to this list)**:

```
listOfCities [0] = new City ("El Paso", "Texas", 259.3, 680000, 3740);
listOfCities [1] = new City ("New York", "New York State", 302.6, 8400000, 33);
listOfCities [2] = new City ("Los Angeles", "California", 503, 3900000, 305);
listOfCities [3] = new City ("Chicago", "Illinois", 234, 2700000, 597);
listOfCities [4] = new City ("Houston", "Texas", 665, 2300000, 79);
listOfCities [5] = new City ("Phoenix", "Arizona", 517, 1600000, 1086);
listOfCities [6] = new City ("Philadelphia", "Pennsylvania", 141.7, 1500000,
39);
listOfCities [7] = new City ("San Antonio", "Texas", 505, 1500000, 650);
listOfCities [8] = new City ("San Diego", "California", 372.4, 1400000, 62);
listOfCities [9] = new City ("Dallas", "Texas", 385.8, 1300000, 430);
```

Note: Please practice proper access to information: keep the class variables private, use methods to access the variables. Please create other methods, if you need to.

Example:

*Welcome message/Option 1

```
C:\Users\bsalvarez\Documents\CS 1101\Labs\Lab11>java Runner
Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
1
Name            State           Size            Population Elevation
El Paso         Texas           259.300000      680000     3740
New York        New York State  302.600000      8400000    33
Los Angeles     California      503.000000      3900000    305
Chicago         Illinois        234.000000      2700000    597
Houston         Texas           665.000000      2300000    79
Phoenix         Arizona         517.000000      1600000    1086
Philadelphia    Pennsylvania    141.700000      1500000    39
San Antonio     Texas           505.000000      1500000    650
San Diego       California      372.400000      1400000    62
Dallas          Texas           385.800000      1300000    430

Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
```

*Option 2

```
2
What is the city that you are looking for?
El Paso
Name            State           Size            Population Elevation
El Paso         Texas           259.300000      680000     3740

The type of city is: City

Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
```

*Option 3

```
3
what is the state that you are looking for?
Texas
Name            State           Size            Population Elevation
El Paso         Texas           259.300000      680000     3740
Houston         Texas           665.000000      2300000    79
San Antonio     Texas           505.000000      1500000    650
Dallas          Texas           385.800000      1300000    430

Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
```

*Option 4

```
4
Name            State           Size            Population Elevation
New York        New York State  302.600000      8400000    33

Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
```

*Option 5

```
5
Name            State           Size            Population Elevation
El Paso         Texas           259.300000      680000     3740

Welcome to the list of cities!
Select an option:
 1.Show all the cities in the list
 2.Look up a city by name
 3.Look up cities by a state
 4.Most populated city
 5.Highest city
 6.Exit
```

*Option 6

```
6
Have a good day. Bye!

C:\Users\bsalvarez\Documents\CS 1101\Labs\Lab11>
```

Deliverables: Submit two files in Blackboard:

(i) Lab11_Lastname.doc --- contains the algorithm of your program,
(ii) City.java --- the java file of your program,
(iii) Runner_Lab 11_Lastname.java --- the java file of your program.

Grading Criteria:

- [10 points] Algorithm or pseudocode for Runner class.
  - Show the flow of the program.

- [30 points] Java program. → City Class
  - [10 points] The program uses meaningful variable names, and meaningful comments. Has proper access to information: class variables private
  - [10 points] Has the required constructors specified in the instructions.
  - [10 points] Has the appropriate setter and getter methods.

- [60 points] Java program that is similar to the algorithm. → Runner Class
  - [20 points] The program compiles and runs.
  - [10 points] The program generates correct output.
  - [20 points] The program uses methods and an array of objects from the City class to access the information required.
  - [5 points] The program uses meaningful variable names, and meaningful comments.
  - [5 points] The program is indented properly.

- Late submission [**-10**] points for every 24 hours after the deadline.

If you need clarification, ask the TA. Your TA will instruct you with further details.