CSE1322L - Assignment 3 (Spring 2026)

Introduction:

These days, there are several different ways to get a weather forecast. You could use the oldest method available (looking outside and guessing), but you could also check any news channel on TV, dozens of internet websites, apps on your phone, or through some sort of widget on your computer. Besides the first method, all the other methods listed rely in part or in whole on measurements from one or more Weather Forecast Offices spread across the US, which feed their data to the National Weather Service.

In this assignment, we will write a simplified version of this system, in part using classifications and metrics from the National Oceanic and Atmospheric Administration (e.g.: Types of Weather Phenomena), and in part with information we'll make up.

Requirements

The features described below must be in your program.

- A total of **eight** classes: the driver, WeatherEvent, Precipitation, Obscuration, Rain, Snow, Fog, and Particle.
- WeatherEvent class, the superclass of all other classes (except the driver):
  - Must have 4 fields:
    - A string called "location"
    - A static integer called "nextId", initialized at 0
    - An integer called "id"
    - A boolean called "active"
  - **WeatherEvent(String, boolean)**: This overloaded constructor sets id with the value of nextId, and then increments nextId by 1. It then assigns the arguments to the appropriate fields
  - All fields except for "nextId" must have a getter
  - "location" and "active" must have setters
  - **String toString()**: This override must return a string in the following format:

```
"Weather Event Location: {location}
 id: {id}
 active: {active}"
```

- Precipitation class, which is a superclass of Rain and Snow
  - Must have 1 field: a double called "rateOfFall"
    - This field must have a getter and a setter. Do not allow this field to be set at a value below 0.
    - This field is keeping track of how fast the Precipitation is happening, measured in inches/h.
  - **Precipitation(String, boolean, double)**: the first 2 arguments must be passed to the superclass constructor while the last argument is used to set the object's own

field. Remember to not allow the field to be set to a negative value: if the passed argument is negative, set the field to 0 instead.

- **String toString()**: Overrides its superclass method by adding information to it. The information added is the Precipitation's rateOfFall and, in parenthesis, if the rate is light, medium or heavy:
    - Light: less than 0.5 inches/h
    - Medium: Between 0.5 and 1.5 inches/h (inclusive)
    - Heavy: Above 1.5 inches/h

```
“Weather Event Location: {location}
 id: {id}
 active: {active}
 Rate of Fall: {rateOfFall} in/h ({Light/Medium/Heavy})”
```

- Obscuration class, a superclass of Fog and Particle:
    - Must have 1 field: an integer called “visibility”
        - This field must have a getter and a setter. Do not allow this field to be set at a value below 0.
        - This field is keeping track of how far away a person can see through the Obscuration, measured in eights of a mile.
    - **Obscuration(String, boolean, int)**: the first 2 arguments must be passed to the superclass constructor while the last argument is used to set the object's own field. Remember to not allow the field to be set to a negative value: if the passed argument is negative, set it to 0 instead.
    - **String toString()**: Overrides its superclass method by adding information to it. The information added is the Obscuration's visibility. If the visibility is at or above 56, the visibility is “Normal”. Otherwise, the visibility should be displayed in eights of a mile.

```
“Weather Event Location: {location}
 id: {id}
 active: {active}
 Visibility: {visibility}/8 mi”
```

OR

```
“Weather Event Location: {location}
 id: {id}
 active: {active}
 Visibility: Normal”
```

- Rain class
    - Must have 1 field: a double called “dropSize”
        - This field must have a getter and a setter. Do not allow this field to be set at a value below 0.02.

        - This field is keeping track of the diameter of the rain drops, measured in inches.
    - **Rain(String, boolean, double, double)**: The first 3 arguments must be passed to the superclass constructor while the last argument is used to set the object's own field. Remember to not allow this field to be set to a value below 0.02. If the argument is less than 0.02, set the field to 0.02.
    - **String toString()**: Overrides its superclass method by adding information to it. The information added is the Rain's drop size, as well as their classification. Drop sizes as classified as:
        - Small: Less than 0.066 inches
        - Medium: Between 0.066 and 0.112 inches (inclusive)
        - Large: Greater than 0.112 inches

```
"Weather Event Location: {location}
 id: {id}
 active: {active}
 Rate of Fall: {rateOfFall} in/h ({Light/Medium/Heavy})
 Drop size: {dropSize} ({Small/Medium/Large})"
```

- Snow class
    - Must have 1 field: a double called "temperature"
        - This field must have a getter and a setter. Do not allow this field to be set at a value below -459.67 or above 32.
        - This field is keeping track of the temperature of the snowfall, measured in Fahrenheit.
    - **Snow(String, boolean, double, double)**: The first 3 arguments must be passed to the superclass constructor while the last argument is used to set the object's own field. Remember cap the field's value in the range described above.
    - **String toString()**: Overrides its superclass method by adding information to it. The information added is the temperature.

```
"Weather Event Location: {location}
 id: {id}
 active: {active}
 Rate of Fall: {rateOfFall} in/h ({Light/Medium/Heavy})
 Temperature: {temperature} F"
```

- `Fog class`
    - Must have 1 field: a boolean called "freezingFog"
        - This field must have a getter and a setter.
        - This field is keeping track of if the fog is freezing or not.
    - **Fog(String, boolean, int, boolean)**: The first 3 arguments must be passed to the superclass constructor while the last argument is used to set the object's own field.
    - An override to setVisibility(): A Fog's visibility must be greater than 0 but less than 5.

        - If the user tries to set to a value outside of that range, set the value to be the nearest valid value (either 1 or 4).
        - Fogs are characterized by their severe lack of visibility, hence the heavy restriction on visibility.
    - **String toString()**: Overrides its superclass method by adding information to it. If the Fog is freezing, add “ALERT! FREEZING FOG!”. Otherwise, add nothing.

      ```
      “Weather Event Location: {location}
       id: {id}
       active: {active}
       visibility: {visibility}/8 mi
       ALERT! FREEZING FOG!”
      ```

      OR

      ```
      “Weather Event Location: {location}
       id: {id}
       active: {active}
       visibility: {visibility}/8 mi”
      ```

- `Particle class`
    - Must have 1 field: a string called “particleType”
        - This field must have a getter and a setter. <u>This field must have only one of the following values</u>:
            - Dust
            - Sand
            - Ash
            - Any value that’s not one of the above should be set as “Other”.
        - This field is keeping track of what the Obscurant is made of
    - **Particle(String, boolean, int, String)**: The first 3 arguments must be passed to the superclass constructor while the last argument is used to set the object's own field. <u>Remember that the object's field can only have of 4 different values</u>; refer to them above.
    - **String toString()**: Overrides its superclass method by adding information to it. The information added is the Obscuration’s particle type.

      ```
      “Weather Event Location: {location}
       id: {id}
       active: {active}
       visibility: {visibility}/8 mi
       Particle type: {particleType}”
      ```

      OR

      ```
      “Weather Event Location: {location}
       id: {id}
      ```

```
active: {active}
visibility: Normal
Particle type: {particleType}”
```

- In the Driver:
  - Create an Arraylist of WeatherEvents
  - In a loop, prompt the user for the following options:
    - **Add weather event:** Prompt the user for what type of WeatherEvent they wish to create (Rain, Snow, Fog, Particle), then prompt them for the necessary information to create said event, adding it to the arraylist. Print an error message if the user picks a type of WeatherEvent that doesn’t exist.
    - **Update location:** Prompt the user for the ID of a WeatherEvent. If said ID exists, prompt the user for the new location of said WeatherEvent and update it. Otherwise, print an error message that no such WeatherEvent exists.
    - **Update active:** Prompt the user for the ID of a WeatherEvent. If said ID exists, invert the activity status of that weather event (from true to false and vice-versa). Otherwise, print an error message that no such WeatherEvent exists.
    - **View all events:** Calls and prints the toString() of all WeatherEvents in the arraylist.
    - **Quit:** Terminate the program.
- You must generate a UML diagram of your classes. Check Lab 5 on instructions and expectations of UML diagrams.

Deliverables

- Assignment3.java (driver)
- WeatherEvent.java
- Precipitation.java
- Obscuration.java
- Rain.java
- Snow.java
- Fog.java
- Particule.java
- UML.pdf (UML diagram)

Considerations

- This assignment may seem intimidating, but that’s just because of the number of things you have to do; the assignment itself isn’t very hard, so don’t be discouraged.
- Remember that you will get partial credit for partial work. Try to deliver as much of the assignment as you can.
- Despite what the deliverables say, you can submit all your classes in a single file.

- Remember that a subclass inherits all the public members of its superclass. As such, even if your Rain class doesn't have a method called "getId()", it can still call this method because:
    - Rain is a subclass of WeatherEvent (which does have a getId())
    - getId() is a public method in WeatherEvent
- You may add any other helper methods you believe are necessary, but they won't count towards your grade.

Sample Output (user input in red)

```
[Weather Tracking System]
1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 6
Invalid option!

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 1

1. Rain
2. Snow
3. Fog
4. Particle
What type of weather event is being added? 1
Where is the event happening? Clayton
What is the rate of fall? (in/h) 0.6
What is the diameter of the drops? (in) 0.04
Rain event added

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
```

```
Enter your option: 1

1. Rain
2. Snow
3. Fog
4. Particle
What type of weather event is being added? 2
Where is the event happening? Cobb
What is the rate of fall? (in/h) 1.7
What is the temperature? (F) 50
Snow event added

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 1

1. Rain
2. Snow
3. Fog
4. Particle
What type of weather event is being added? 3
Where is the event happening? Douglas
What is the visibility? (1/8 mi) 0
Is the fog freezing? (y/n) y
Fog event added

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 1

1. Rain
2. Snow
3. Fog
4. Particle
```

```
What type of weather event is being added? 4
Where is the event happening? Hawaii
What is the visibility? (1/8 mi) 60
What is the obscuration made of? (Sand/Dust/Ash/Other) Sulfur
Particle event added

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 4

Weather Event Location: Clayton
id: 0
Active: true
Rate of fall: 0.60 in/h (Medium)
Drop size: 0.04 (Small)

Weather Event Location: Cobb
id: 1
Active: true
Rate of fall: 1.70 in/h (Heavy)
Temperature: 32.00 F

Weather Event Location: Douglas
id: 2
Active: true
Visibility: 1/8 mi
ALERT! FREEZING FOG!

Weather Event Location: Hawaii
id: 3
Active: true
Visibility: Normal
Particle type: Other


1. Add weather event
2. Update location
```

```
3. Update active
4. View all events
5. Quit
Enter your option: 3

Enter id of weather event: 4
No event with that id.

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 3

Enter id of weather event: 3
Event set to "inactive"

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 2

Enter id of weather event: 2
Enter the new location of the weather event (currently “Douglas”):
Floyd
Location updated

1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 4

Weather Event Location: Clayton
id: 0
Active: true
```

```
Rate of fall: 0.60 in/h(Medium)
Drop size: 0.04 (Small)

Weather Event Location: Cobb
id: 1
Active: true
Rate of fall: 1.70 in/h(Heavy)
Temperature: 32.00 F

Weather Event Location: Floyd
id: 2
Active: true
Visibility: 1/8 mi
ALERT! FREEZING FOG!

Weather Event Location: Hawaii
id: 3
Active: false
Visibility: Normal
Particle type: Other


1. Add weather event
2. Update location
3. Update active
4. View all events
5. Quit
Enter your option: 5
Shutting off...
```