CS212: Paradigms

Dr. Gross, Spring 2016

Homework 4 - Inherit the Wind

1. Due Thursday, Feb 4 by the beginning of lab (push with comment "submission for hw3")
2. Open your lab project's src folder firstname.lastname
3. Create a new UML file wind.uml and save to lastname.firstname\Documents
4. Click on that file in NetBeans and go to Team->Add
5. Go to Team->Commit… and enter "created UML file"
6. Start your design
7. Create a new package called edu.blackburn.cs.cs212sp16.lastname.wind for your code

We're going to make a weather system. It will give us a ten-day report:

```
Welcome to Weather 3000! Here's your report in degrees Celsius
Day 1:
Morning Temperature: 18 C
Clouds: None
Midday Temperature: 24 C
Precipitation: None
Wind: 18 kph N

Day 2:
Morning Temperature: 12 C
Clouds: Light
Midday Temperature: 15 C
Precipitation: 6 cm Rain
Wind: 5 kph N

Day 3:
Morning temperature: 6 C
Clouds: Medium
Midday Temperature: 3 C
Precipitation: 10 cm Rain
Wind: 12 kph N

Day: 4
Morning temperature: -14 C 13.75
Clouds: None
Midday Temperature: -8 C
Precipitation: 20 cm Snow
Wind: 20 kph S 14

Day 5:
Morning temperature: -1 C
Clouds: None
Midday temperature: 5 C
Precipitation: 2 cm Rain
Wind: 8 kph S
```

…

Some rules:

- All classes should override the toString() method and return the string above
- Create a new set of objects for each day
- For Day 1, set the initlal temperature randomly to a reasonable number that is not whole
- Clouds
    - Only clouds affect the change from morning to midday temperature
    - Cloud level is determined randomly
    - Cloud level can be heavy, medium, light, or there can be no clouds
    - No clouds will raise the temperature by 6 degrees
    - Light clouds will raise the temperature by 3 degrees
    - Medium clouds will lower the temperature by 3 degrees
    - Heavy clouds will lower the temperature by 14 degrees
- Wind
    - Wind speed and direction are determined randomly
    - Wind from the south will raise tomorrow morning's temperature by 0.5 degrees per kph
    - Wind from the north will lower tomorrow morning's temperature by 0.65 degrees per kph
- Precipitation
    - Precipitation amount is determined randomly and is the same for rain and snow
    - However, the amount should be multiplied by 10 for snow (a rough approximation)
    - You will need to know the midday temp to determine if it will be snow or rain
    - Rain will lower tomorrow morning's temperature by .9 degrees per cm
    - Snow will lower tomorrow morning's temperature by .15 degrees per cm
- Measurement
    - You'll need the Measurement class from Thursday lab, and you will need to subclass it
    - Think of the number and kinds of measurements
    - Measurement and its subclasses should store values as doubles
    - They should show rounded whole numbers when converted to a string
    - ROUNDED, not truncated
- Inheritance
    - Is Snow a subclass or Rain? Rain a subclass of Snow? Something else?
    - Where else can you use inheritance?
    - Where *should* you use inheritance? Where *shouldn't* you use inheritance?
- Overall
    - All classes should override public String toString() and return the string (or part of the string) printed out above
    - Print out 20 days
    - You can use static once, for your main method

Some hints:

1. Don't worry about a command interface; I know you can use Scanner
2. Write test code and comment what it should do and what it proves if it works

3. Design a bit, code a bit, test a bit
4. If a method is longer than 10 lines, create a new method (hint: use Refactor->Extract Method)
5. How do you know if your calculations are correct? (hint: print a log file)
6. j 2

To save your work:

1. When you add new files (or think you might have) go to Team->Add
    a. THIS OFTEN FAILS!
2. Every few minutes, when the code compiles, go to Team->Commit
3. When you are ready to upload your code (at least every 30 mins), go to Team->Remote->Push to Upstream
4. You can always check the Github website to verify that everything is there and current

Grading Rubric

	Issue	**Points**
1	Coding standards and comments	10
2	Time estimate and final accounting	5
3	Properly using the repository	5
4	Class diagram syntactically correct	10
5	Class diagram coherent & consistent	5
6	Implementation is object-oriented (no static)	10
7	Measurement & subclasses designed/implemented correctly	10
8	Objects for different weather types designed/implemented correctly	10
9	Correct calculations of temperature changes	10
10	Correct reporting of temperature	10
11	At least one other correct use of inheritance beyond Measurement	10
12	Proper use of visibility	5