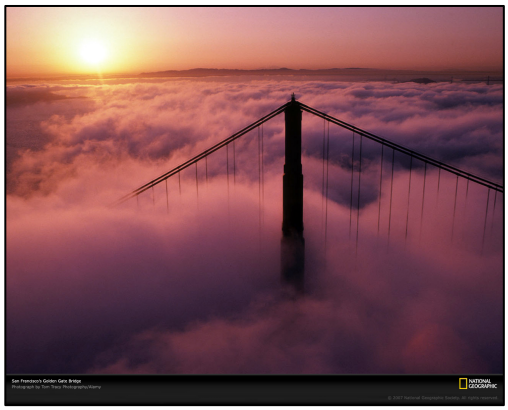
CS46B
Spring 2018
Homework 2

In this assignment you will model 3 kinds of cloud: stratus, cumulus, and cirrus.



Stratus clouds

Cumulus clouds

Cirrus clouds

Stratus and cumulus clouds can make rain; cirrus clouds cannot.

DUE DATE

This assignment is due on or before 11:59 PM on Saturday February 10. Late work will not be accepted except in cases of documented personal emergency.

GETTING STARTED

These instructions assume you've become familiar with Eclipse by doing Lab 1 / Homework 1. If any steps here are confusing, see the illustrations in the Lab 1 / HW1 instructions.

Create a directory called "HW2Workspace" wherever you choose. Start Eclipse. If it prompts you to "Select a Workspace", browse to HW2Workspace. Otherwise, it might come up in the last workspace you were in; in this case select File -> Switch Workspace and browse to HW2Workspace.

If you see the Welcome screen below, click the arrow circled in red.

Fig. 1

Use the menu item File -> New Java Project to create a new project. Caution: create a new Java Project, not a new Project. A wizard will appear. Type "hw2project" for the project name, and use defaults for all other settings. When you're finished, the Package Explorer will display your project.

Click on the tiny right-arrow next to the project name to expand the project. There isn't anything in the project yet, so the expansion won't be dramatic.

Right-click on the "src" icon and select the New->Package menu item; fill in the wizard form to create a package called "weather".

Create a class called "Sky" in the weather package as follows: Right-click on the icon next to "weather" in the Package Explorer and use the New -> Class menu item to create a class called "Sky" in the weather package.

Create 4 other classes in the weather package: Cloud, StratusCloud, CumulusCloud, and CirrusCloud. The class names, as well as the package name, must be spelled and capitalized exactly as shown here, or the automatic tester software will give you zero points.

The large panel in the center of Eclipse now lets you edit your source files. Note that the first line of each source says "package weather;". When a source file is in a package, this package declaration line must be the first thing in the file.

Now you're ready to write a polymorphic model. Fill in the bodies of all the classes as described below. When everything is running, you will create one last class, called Sky2, that will be a cleaner implementation of Sky.

Cloud.java

This class needs:
1) Private float instance variables: bottom and top.
2) A ctor that takes bottom and top as args and stores them in the instance variables. NOTE: "ctor" is an abbreviation for "constructor".
3) A public method getHeight() that returns a float, whose value is top minus bottom.
4) A public String method rain() that returns "It is raining"

StratusCloud.java and CumulusCloud.java

For these classes:
1) Change the class declarations so that they are subclasses of Cloud.
2) Fix the ctors so that they pass bottom and top to the superclass ctor.

CirrusCloud.java

Subclasses of Cloud inherit a rain() method that is not appropriate for cirrus clouds. Override rain() so that it returns "I cannot make rain"

Sky.java

This class holds lots of cloud instances of various types, and computes the average cloud height. It needs:
1) A private ArrayList<Cloud> called clouds.
2) A ctor that creates the clouds array list with an initial capacity of 100. Look up the javadoc API page for ArrayList to see what ArrayList ctor to call. If you don't know how to do this, ask on Piazza. DON'T ask Piazza for the answer, just ask how to find the javadoc API page.

3) A public method called add(Cloud c) that takes a cloud and adds it to the array list. The return type of this method should be boolean, and the method should always return true. That's a little weird but it will make more sense later.
4) A public float method called getMeanHeight(), which returns the average height of all the clouds in the array list.
5) A main() method. Paste in the following:

```
	public static void main(String[] args)
	{
		StratusCloud strat = new StratusCloud(100, 1000);
		if (!strat.rain().startsWith("It is raining"))
			System.out.println("Bad StratusCloud::rain");
		CumulusCloud cumu = new CumulusCloud(200, 2000);
		if (!cumu.rain().startsWith("It is raining"))
			System.out.println("Bad CumulusCloud::rain");
		CirrusCloud cirr = new CirrusCloud(300, 3000);
		if (!cirr.rain().startsWith("I cannot make"))
			System.out.println("Bad CirrusCloud::rain");
		Sky sky = new Sky();
		sky.add(strat);
		sky.add(cumu);
		sky.add(cirr);
		float mean = sky.getMeanHeight();
		if (mean < 1799  ||  mean > 1801)
			System.out.println("Bad mean height: expected 1800, saw " + mean);
		System.out.println("Everything (else) is ok");
	}
```

To test your work so far, run Sky as an application. Click the tiny "Run" button at the top of Eclipse (a white triangle pointing right, inside a green circle). The only output should be "Everything (else) is ok".

Sky2.java

Lastly, implement Sky in a somewhat different way. When one class holds many references to instances of another class, we say (for example) that Sky "aggregates" clouds. Sky aggregates by owning an array list. Sky2 will aggregate by *being* (in a sense) an array list. That is, Sky2 will *extend* ArrayList<Cloud>.

Create class Sky2 in the weather. The Sky2 ctor will need to explicitly call the correct superclass ctor, so that the initial capacity is 100.

Copy getMeanHeight() from Sky. You need to change it so that it traverses the correct list of clouds, which is the instance of Sky2 that is executing getMeanHeight(). Huge hint: its name is "this".

What about adding clouds to Sky2? You could just copy add() from Sky and then make a simple change. But there is an even simpler way.

Paste in the main() method from the Sky class. Change the line “Sky sky = new Sky()” to “Sky2 sky = new Sky2()” To test Sky2, run it as an application.

Testing your work

Import WeatherTester.java, which you downloaded along with this document. The easiest way to do this is to just drag-and-drop it into the “weather” package in the Package Explorer. Run WeatherTester as an app. For this assignment, nothing works unless almost everything works, so there isn’t any partial credit. You’ll either get 100 points or 0 points. If you get 0 points the tester will tell you exactly what the problem is, so you can fix your bug and get 100.

SUBMISSION

Export your work as a jar file called “CS46BHW2.jar”. Follow the same procedure that you used in Lab 1 (see Step 12 of the instructions) to list the contents of your jar. It must contain all the .java files (source files) that you developed. If you upload the wrong jar, or a jar that doesn’t contain all the java sources, you’ll get zero points for this homework. This rule is non-negotiable.

When you’re satisfied, upload CS46BHW2.jar to Canvas.