CSCI 136 Programming Exam #0
Fundamentals of Computer Science II
Spring 2013

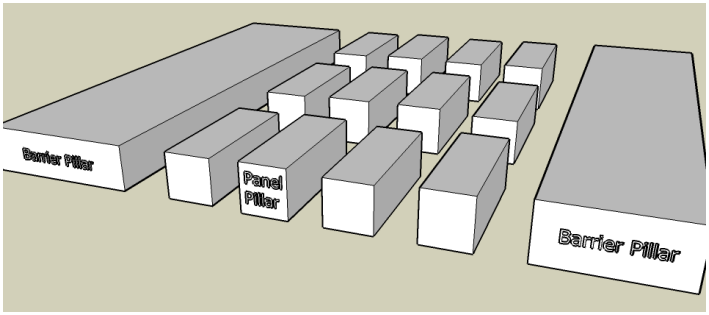
This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #0 dropbox. Please ***double check you have submitted all the required files***.

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

Overview. You are working for a coal mine in West Virginia. Your mine uses a continuous mining machine that digs up underground section of the mountain to obtain the high-grade coal used to make steel.

Unfortunately, if you dig up too much of the mountain, the mine will collapse. To avoid this, you are employing the "room and pillar" mining system. In this system, certain sections have been designated as "pillars". These pillars must not be mined under any circumstances.



The mine has been carefully surveyed, mapping it out as a rectangular grid of square sections. Each section has an estimated amount of coal in tons. We refer to each section by an (x, y) pair with (0, 0) being the southwest corner, (1, 0) is the section directly to the east of the southwest corner, (0, 1) is north of the corner, etc.

There are already some sections that have already been mined, these are the passageways from which you can mine further into the mountain. Your continuous mining machine can only mine straight north, south, east or west. Each "cut" made consists of a single section adjacent to an existing passageway.

You will be developing a software simulation of the mining operation. You will be developing two classes, `Section.java` and `Mine.java`. Partially completed versions of these classes as well as two mine survey files `mine10x5.txt` and `mine20x10.txt` can be downloaded from here:

http://katie.mtech.edu/classes/csci136/mine.zip

Part 1. First you should implement the data type `Section`. This basic class represents a square section of the mountain. It knows things like whether it is marked as a pillar, whether it has already been mined (thus forming a passageway), and its estimated amount of coal (in tons). It can construct itself based on a single letter code and a coal amount. It can also mark itself as mined. Here is the public API for `Section`:

```
public class Section
-------------------------------------------------------------------------------
               Section(String type, double tons)
      boolean  getMined()
      boolean  getPillar()
      double   getCoal()
      void     mine()
```

We have provided a test `main()` method. Here is our output:

```
% java Section
s1.getMined()  = false
s1.getPillar() = true
s1.getCoal()   = 10.5

s2.getMined()  = true
s2.getPillar() = false
s2.getCoal()   = 0.0
s2.mine()
s2.getMined()  = true
s2.getPillar() = false
s2.getCoal()   = 0.0

s3.getMined()  = false
s3.getPillar() = false
s3.getCoal()   = 42.2
s3.mine()
s3.getMined()  = true
s3.getPillar() = false
s3.getCoal()   = 0.0

s1.mine()
Exception in thread "main" java.lang.RuntimeException: Can't mine a pillar!
       at Section.mine(Section.java:46)
       at Section.main(Section.java:93)
```

Part 2. The Mine class is the brains of the operation. It knows things like the grid of `Section` objects that make up the mine. It can do things like construct itself based on a text file containing the mine survey, draw a graphical representation of the current state of the mine, calculate the most estimated coal in any section that is not a pillar, and mine the most promising section. Here is the public API for `Mine`:

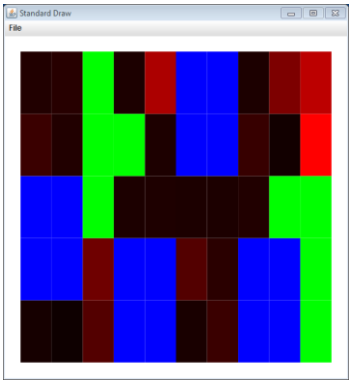
```
public class Mine
--------------------------------------------------------------------------------
               Mine(String filename)
      double   getMaxCoal()
      boolean  mineBestLocation()

      void     draw()                    // implemented for you (used by simulation)
      int      getWidth()                // implemented for you (for our testing)
      int      getHeight()               // implemented for you (for our testing)
      Section  getSection(int x, int y)  // implemented for you (for our testing)
```

Here is the mine survey file `mine10x5.txt`:

```
10 5
- 12.0 - 14.0 M  5.0 - 10.0 - 62.3 P 33.0 P  0.5 - 10.0 - 45.3 - 67.9
- 21.0 - 12.0 M  6.0 M  0.0 - 10.5 P 22.0 P  0.0 - 20.0 -  6.6 - 92.3
P 33.0 P  2.0 M  8.0 - 10.0 - 11.0 - 10.0 - 10.0 - 12.0 M  2.3 M  0.0
P 10.0 P 10.0 - 40.0 P 40.0 P 99.0 - 30.0 - 15.2 P 13.0 P  0.9 M  0.0
-  8.0 -  5.0 - 30.0 P 10.0 P 10.0 -  9.0 - 21.1 P 15.0 P  0.6 M  0.0
```

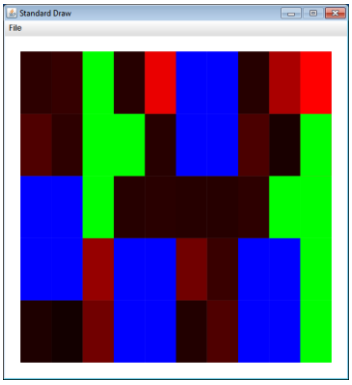
The first integer specifies the *width* of the mine in sections. The second integer specifies the *height* of the mine in sections. This is followed by pairs consisting of a letter code and a floating-point coal estimate value for each section. The file is arranged with *width* pairs on each line and *height* lines. The southwest corner of the mine, grid location (0, 0), is the first pair on the final line. For example "`- 9.0`" in the above survey file is the southwest corner.

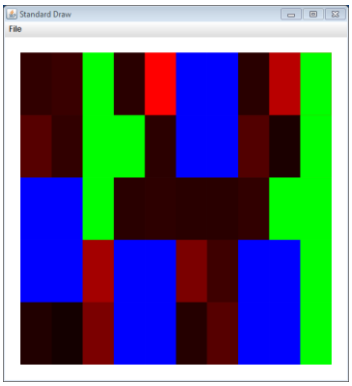
We have supplied a completed `main()` method for Mine. The `main()` method constructs a mine using the file specified as the first command-line argument. The program then simulates a greedy mining operation in which the next accessible section with the most estimated coal is mined. At each step of the simulation, a graphical representation of the mine is shown. The step number and the maximum coal in any non-pillar section is printed out to the console:

	`Step 0, max coal = 92.30`

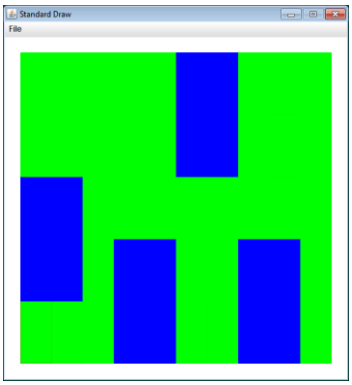
In the above diagram, green denotes mined sections, blue denotes pillars, and shades of red denote mineable sections. The more red a section, the higher the estimated coal in that section. The simulation then proceeds to choose the best section to mine next. The next section must be directly to the north, south, east, or west of an already mined section. It must also not be designated as a pillar.

Here is the output of the next few steps:

	`Step 0, max coal = 92.30` `Step 1, max coal = 67.90`

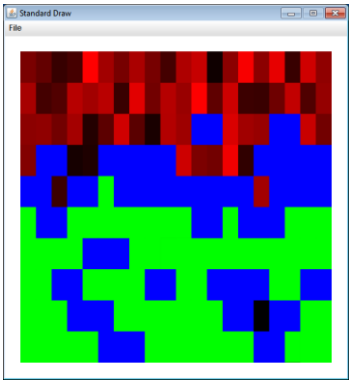
	`Step 0, max coal = 92.30`
	`Step 1, max coal = 67.90`
	`Step 2, max coal = 62.30`

. . .	. . .



```
...
Step 17, max coal = 30.00
Step 18, max coal = 21.10
Step 19, max coal = 21.10
Step 20, max coal = 10.00
Step 21, max coal = 10.00
Step 22, max coal = 9.00
Step 23, max coal = 8.00
Step 24, max coal = 8.00
Step 25, max coal = 8.00
Step 26, max coal = -Infinity
No more coal!
```

Note that for some mines, all the sections may not be mineable if they are cutoff from the passageways by pillar sections. Here is our final output for `mine20x10.txt`:



```
...
Step 50, max coal = 98.25
Step 51, max coal = 98.25
Step 52, max coal = 98.25
Step 53, max coal = 98.25
Step 54, max coal = 98.25
Step 55, max coal = 98.25
No more coal!
```