CSCI-101 Introduction to Programming

Lab 10 - Tuesday

Log onto cs.bridgewater.edu. Change your working directory to the **labs** directory in your repository. Create a subdirector (in labs) named **lab10**. Change your working directory to **lab10**. When complete with this lab, please push your code to GitHub and update your **README.md** file to provide the path to the **lab10** directory and your Java source code.

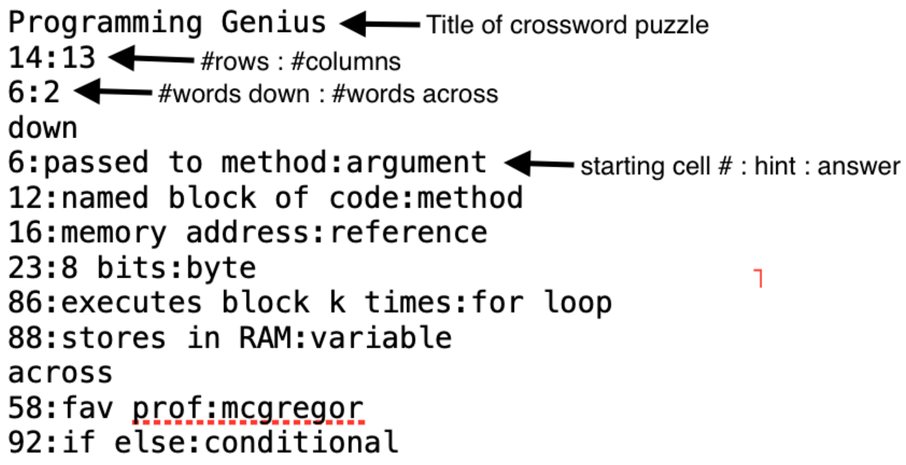
---

puzzle1.txt

Copy the following lines of text into the file named puzzle1.txt

```
Programming Genius
14:13
6:2
down
6:passed to method:argument
12:named block of code:method
16:memory address:reference
23:8 bits:byte
86:executes block k times:for loop
88:stores in RAM:variable
across
58:fav prof:mcgregor
92:if else:conditional
```

The data in puzzle1.txt describes a crossword puzzle. Below is an explanation of each line of data.



```
Programming Genius
14:13
6:2
down
6:passed to method:argument
12:named block of code:method
16:memory address:reference
23:8 bits:byte
86:executes block k times:for loop
88:stores in RAM:variable
across
58:fav prof:mcgregor
92:if else:conditional
```

Title of crossword puzzle
#rows : #columns
#words down : #words across
starting cell # : hint : answer

Please note that the starting cell # specified in the file assumes the numbering starts at 1, not 0.

Crossword.java

Write a program in a file named **Crossword.java** that takes a filename as an argument on the command line. The program reads the file and prints the crossword puzzle board described in the file.

Note that your program should work with any crossword puzzle file that has the same format as the one described above.

After you've initialized a Scanner reads the data from the file into the following data structures.

String title	// holds the title of the crossword puzzle
char[][] board	// holds the crossword puzzle board
String[][] down	// holds the starting location, hint, and answer for each down word
String[][] across	// holds the starting location, hint, and answer for each across word

Initialize the 2D board so that all cells are set to the Unicode character '\u25A0' (filled box).

Write methods named **insertWordDown()** and **insertWordAcross()** and use them to set the characters where the words are positioned on the board to the Unicode character '\u25A1' (border box).

Create a method named **printBoard()** that takes the title of the puzzle and the board as arguments and prints the title followed by the contents of the board to the screen similar to the board shown below.

```
Programming Genius
┌───────────────────────────┐
│ ■ ■ ■ ■ ■ □ ■ ■ ■ ■ ■ □ ■ │
│ ■ ■ □ ■ ■ □ ■ ■ ■ □ ■ □ ■ │
│ ■ ■ □ ■ ■ □ ■ ■ ■ □ ■ □ ■ │
│ ■ ■ □ ■ ■ □ ■ ■ ■ □ ■ □ ■ │
│ ■ ■ □ ■ ■ □ □ □ □ □ □ □ □ │
│ ■ ■ □ ■ ■ □ ■ ■ ■ ■ ■ □ ■ │
│ ■ ■ □ ■ ■ □ ■ □ ■ □ ■ ■ ■ │
│ □ □ □ □ □ □ □ □ □ □ □ ■ ■ │
│ ■ ■ □ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
│ ■ ■ □ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
│ ■ ■ ■ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
│ ■ ■ ■ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
│ ■ ■ ■ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
│ ■ ■ ■ ■ ■ ■ ■ □ ■ □ ■ ■ ■ │
└───────────────────────────┘
```

Note that your colors will be inverted if your terminal has a black screen.

That's all for now. Just get the board to print nicely on the screen. We'll add the game mechanics later.

Bonus: Use the box drawing Unicode characters to draw a box around the board. https://en.wikipedia.org/wiki/Box-drawing_character