Input and Output

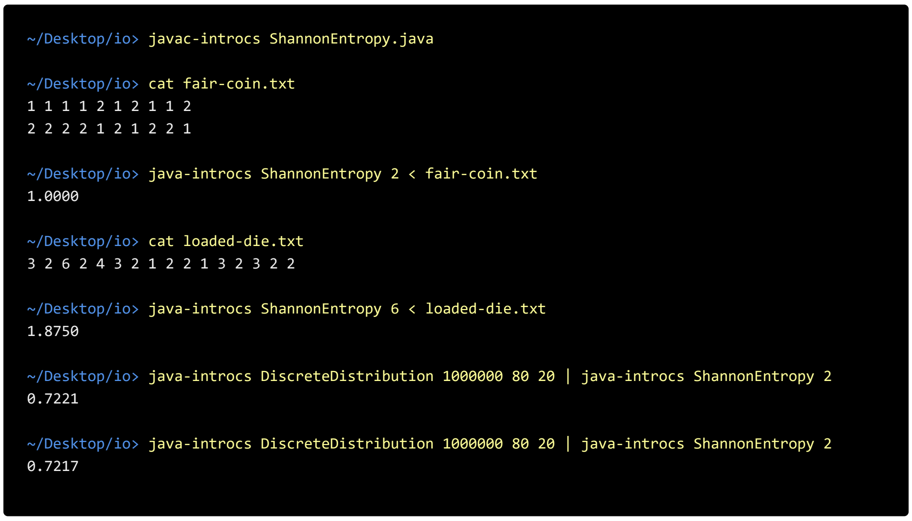
Spec FAQ Project Submit

The purpose of this assignment is to give you practice writing programs with input and output, including standard output, standard input, and standard drawing.

1. **Shannon entropy.** Write a program `ShannonEntropy.java` that takes a command-line integer *m*; reads a sequence of integers between 1 and *m* from standard input; and prints the Shannon entropy to standard output, with 4 digits after the decimal point. The *Shannon entropy* of a sequence of integers is given by the formula:

$$H \;=\; -\,(\,p_1 \log_2 p_1 \;+\; p_2 \log_2 p_2 \;+\; \ldots \;+\; p_m \log_2 p_m\,)$$

where p_i denotes the proportion of integers whose value is i. If $p_i = 0$, then treat $p_i \log_2 p_i$ as 0.



```
~/Desktop/io> javac-introcs ShannonEntropy.java

~/Desktop/io> cat fair-coin.txt
1 1 1 1 2 1 2 1 1 2
2 2 2 2 1 2 1 2 2 1

~/Desktop/io> java-introcs ShannonEntropy 2 < fair-coin.txt
1.0000

~/Desktop/io> cat loaded-die.txt
3 2 6 2 4 3 2 1 2 2 1 3 2 3 2 2

~/Desktop/io> java-introcs ShannonEntropy 6 < loaded-die.txt
1.8750

~/Desktop/io> java-introcs DiscreteDistribution 1000000 80 20 | java-introcs ShannonEntropy 2
0.7221

~/Desktop/io> java-introcs DiscreteDistribution 1000000 80 20 | java-introcs ShannonEntropy 2
0.7217
```

Step-by-step calculation. Consider the following sequence of 16 integers generated from a loaded die.

3 2 6 2 4 3 2 1 2 2 1 3 2 3 2 2

This table shows the frequencies x_i, the proportions p_i, and the $-p_i \log_2 p_i$ terms:

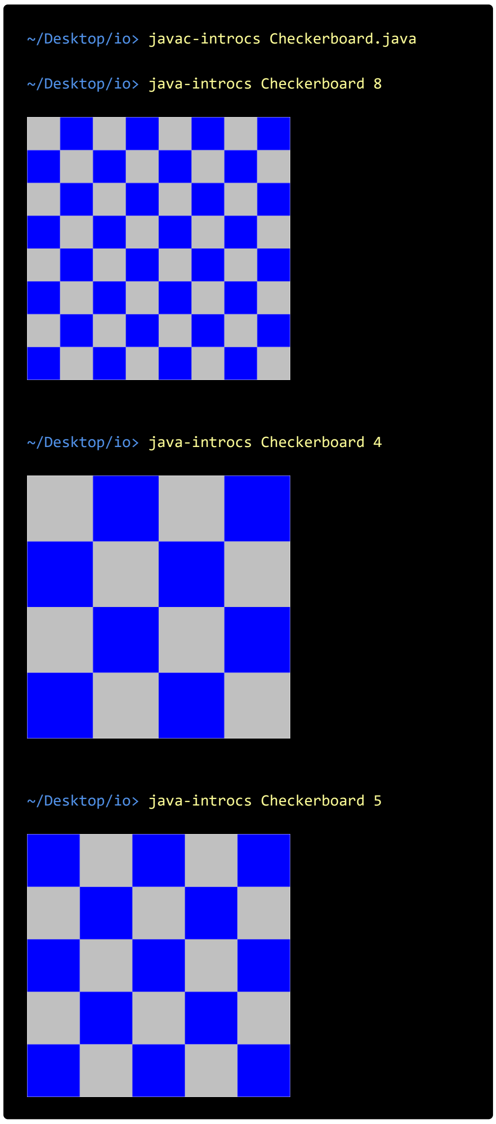
i	x_i	p_i	$-p_i \log_2 p_i$
1	2	1/8	3/8
2	8	1/2	1/2
3	4	1/4	1/2
4	1	1/16	1/4
5	0	0	0
6	1	1/16	1/4
	16	1	15/8

The Shannon entropy is 1.875 = 15/8.

The Shannon entropy is a measure of the rate of information produced by a random source, such as the outcomes of flipping a fair coin or rolling a loaded die. It is a fundamental concept in information theory and data compression.

2. **Checkerboard.** Write a program `Checkerboard.java` that takes a command-line integer *n* and plots an *n*-by-*n* checkerboard pattern to standard drawing. Color the squares blue and light gray, with the bottom-left square blue. To draw,

   - Call `StdDraw.setScale(0, n)` so that *x*- and *y*-coordinates of the canvas range from 0 and *n*.
   - Call either `StdDraw.filledSquare()` or `StdDraw.filledPolygon()` to draw each of the n^2 squares.
   - Make sure that the squares fit snugly in the standard drawing window.
   - Do not change the canvas size.

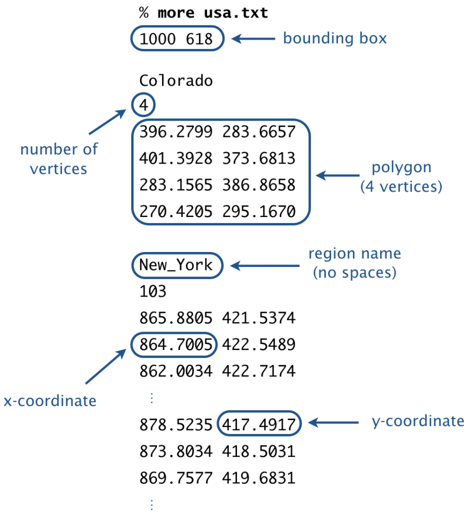


```
~/Desktop/io> javac-introcs Checkerboard.java

~/Desktop/io> java-introcs Checkerboard 8

~/Desktop/io> java-introcs Checkerboard 4

~/Desktop/io> java-introcs Checkerboard 5
```

3. **World maps.** Write a program `WorldMap.java` that reads boundary information of a country (or other geographic entity) from standard input and plots the results to standard drawing. A country consists of a set of regions (e.g., states, provinces, or other administrative divisions), each of which is described by a polygon.

   Input format. The first line contains two integers: *width* and *height*. The remaining part of the input is divided into regions.

   - The first entry in each region is the name of the region. For simplicity, names will not contain spaces.
   - The next entry is an integer specifying the number of vertices in the polygon describing the region.
   - Finally, the region contains the *x*- and *y*-coordinates of the vertices of the polygon.



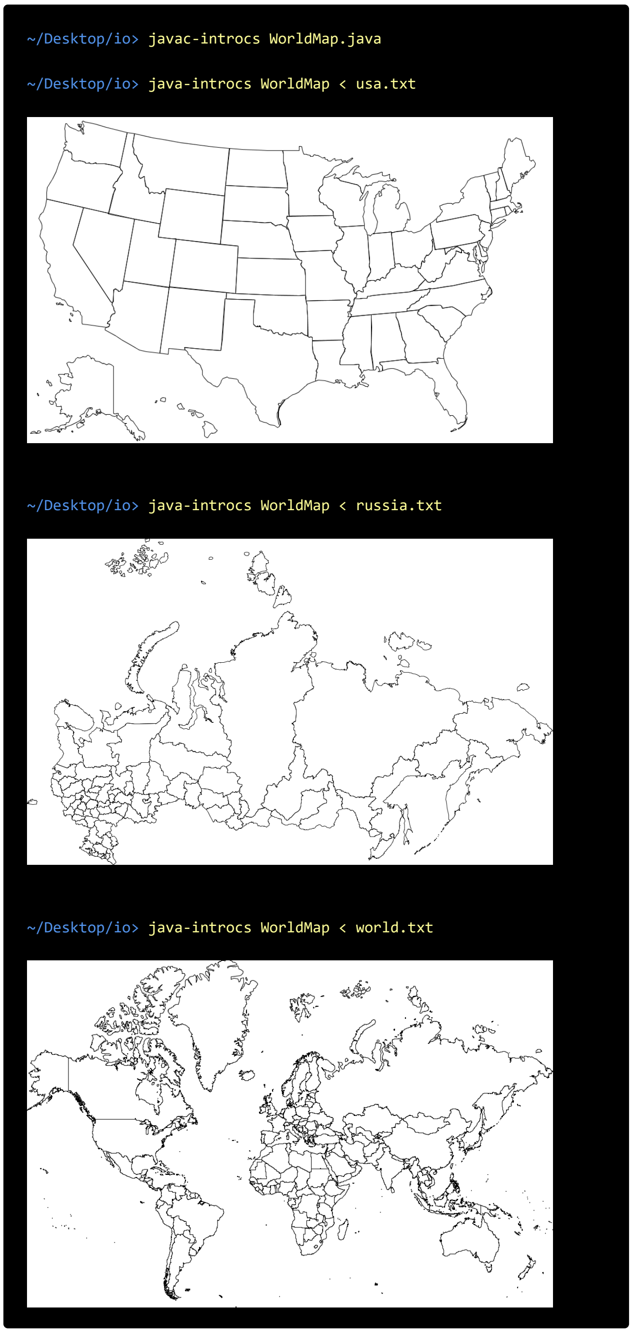
```
% more usa.txt
1000 618

Colorado
4
396.2799 283.6657
401.3928 373.6813
283.1565 386.8658
270.4205 295.1670

New_York
103
865.8805 421.5374
864.7005 422.5489
862.0034 422.7174
   ⋮
878.5235 417.4917
873.8034 418.5031
869.7577 419.6831
   ⋮
```

   For simplicity, if a region requires more than one polygon to describe its boundary, we treat it as multiple regions, with one polygon per region.

Output format. Draw the polygons to standard drawing, using the following guidelines:

- Call `StdDraw.setCanvasSize()` to set the size of the canvas to be *width*-by-*height* pixels.
- Call `StdDraw.setXscale()` and `StdDraw.setYscale()` so that *x*-coordinates of the canvas range from 0 to *width* and the *y*-coordinates range from 0 to *height*.
- Call `StdDraw.polygon()` to draw each polygon.

Here are some sample executions for the input files `usa.txt`, `russia.txt`, and `world.txt`. Additional input files are available for 100+ countries and all 50 U.S. states.



```
~/Desktop/io> javac-introcs WorldMap.java

~/Desktop/io> java-introcs WorldMap < usa.txt
```

```
~/Desktop/io> java-introcs WorldMap < russia.txt
```

```
~/Desktop/io> java-introcs WorldMap < world.txt
```

Submission. Submit a `.zip` file containing `ShannonEntropy.java`, `Checkerboard.java`, and `WorldMap.java`. You may not call library functions except those in the `java.lang` (such as `Integer.parseInt()` and `Math.sqrt()`) and `stdlib.jar` (such as `StdIn.readInt()` and `StdDraw.polygon()`). Use only Java features that have already been introduced in this course (e.g., loops and arrays, but not functions).

This assignment was developed by Kevin Wayne.