This time all input is from the command line.

1. (2 points) Write a program that calculates either the exact division of two integers or the division of two floating-point numbers. You’ll recognize the former as that of Homework 2: Display a rational number in format: whole part fractional part, e.g., \(11 / 4 = 2 3/4\). The latter requires no extra formatting. Make sure that every case (including errors such as dividing by zero) is handled intelligently.

Your program should recognize two flags: `-i` and `-f` to indicate the type of the arguments.

- `-i` Indicates that the following two arguments are integers: numerator, denominator.
- `-f` Indicates that the following two arguments are floating-point numbers: numerator, denominator.

The following two commands should therefore be legal and produce no error messages:

```java
java SomeName -i 45 3
java SomeName -f 45.01 3.03
```

The following errors should be detected:

- too many arguments,
- wrong flag,
- division by zero.

The wrapper classes will automatically detect if you are converting a float stored in a string to an int. We have not discussed how to deal with these cases.

2. (4 points) Write a program that simulates a \(20 \times 32\) display. Think of it as a PDA screen capable of showing 20 lines of 32 characters with the \((0,0)\) coordinate at the upper left hand corner.

From the command line the user can issue requests of the form:

```java
java SomeName -at 8 9 "String to be displayed without inverted commas."
```

where `-at i j` indicates the coordinate starting from which the message should be printed. Your display should be able to wrap text automatically. Assume that there can be multiple `-at i j` commands within the arguments.

All text which is out of bounds is silently clipped.

**Always** print a full \(20 \times 32\) display.