More on Command Line Processing

```
import java.util.Scanner;

public class fToC
{
        public static void main(String[] args)
        {
        if (args.length > 0)
                System.out.println(f_to_c(Double.parseDouble(args[0])));
        else
                {
                Scanner console = new Scanner(System.in);

                while (console.hasNext())
                        System.out.println(f_to_c(Double.parseDouble(console.nextLine())));
                }
        }

        public static double f_to_c(double temp_in_fahrenheit)
        {
        return (temp_in_fahrenheit - 32) * (5.0/9.0);
        }
}



// Program generates temperatures in degrees Fahrenheit from -125 - +135

import java.util.Random;

public class genValues
{
        public static void main(String[] args)
        {
        int numberOfValues = 1000;

        if (args.length > 0)
                numberOfValues = Integer.parseInt(args[0]);  // if a command line argument specified, use it

        Random rnd = new Random();

        for (int i = 0; i < numberOfValues; i++)
                System.out.println(rnd.nextInt(261) - 125);  // temperature ranges -125 - +135

        }
}
```

Sample run:

```
$ java genValues 10 | java fToC
18.88888888888889
-81.66666666666667
51.111111111111114
-72.77777777777779
40.55555555555556
56.111111111111114
23.88888888888889
30.0
-68.88888888888889
-61.111111111111114
```

Notes:

- Observe how the fToC program can handles either a command line argument OR values from STDIN
  - The number of generated values has a default of 1000, unless a command line argument is present
- Notice how conversion from text to int or double is performed using parseInt() or parseDouble()
- The fToC program should be enhanced to catch errors, including... what????
- Outputting temperature values to 10^{-14} implies a very high-level of accuracy, how can this be remedied?