Lab 8

Catching Finch Exception

Try/Catch

```
try {…blockA…}
catch (SomeException e) {…blockB…}
```

- Any exceptions in the *try* block cause execution of the program to end and to go directly to the **matching** *catch*
- But what if the data structure needs to be cleaned?
  - e.g., a dialog box that should be closed

The third block: Finally

```
finally {…blockC… }
```

- The code in a *finally* always runs, no matter what the outcome for the try and catch blocks turn out to be
- Even a *return* within the try or catch blocks will result in the execution of the *finally* block

```
public class ExceptionTest {
   public static void main(String args[]) {
       try {
             int[] nums = {1,2,3,0,4,5};
             for (int i : nums)
                   System.out.println("Sixty divided by "
                         + i + " = " + (60/i));
       }
       catch (NullPointerException npe) {
             System.out.println("Shouldn't see this");
       }
       finally {
             System.out.println("Will we see this?");
       }
       System.out.println("How about this?");
   }
}
```

The output:

```
Sixty divided by 1 = 60
Sixty divided by 2 = 30
Sixty divided by 3 = 20
Will we see this?
Exception in thread "main"
  java.lang.ArithmeticException: / by zero
at ExceptionTest.main(ExceptionTest.java:9)
```

Lab 8

- Modification of project 3
- Instead of changing an incorrect value to a default value, catch them with your own FinchException class
- Specifically, you must catch:
  - negative durations
  - non-positive tones
  - (don’t worry about negative colors or incorrect orientations)

Submission

- Demo: Will only be to see the printing of an exception when it happens and a *display all* of what is left over
- Dropbox: Your exception class, the driver class, and the FinchAction subclasses that are edited to throw these exceptions
- Participation: Either hand in the handout or leave a comment on D2L submission of who worked on the lab