Programming Structures and Abstractions (CS 2334)
Lab 8: Exceptions

November 9, 2009

Due: Friday, November 13th, 2009, 11:29 am

Group members (same as for your project):

Objectives

The objectives of this lab are to:

1. create exception classes,
2. throw exceptions to indicate errors, and
3. catch exceptions to yield robust behavior.

Problem Context

In our implementation of the FinchAction classes to date, we have dealt with the incorrect specification of constructor parameters (e.g., the duration) by simply ignoring the problem or by resetting the instance variables so that they are within legal bounds. In this lab, we will augment our **project 3** implementation with a FinchException class that will be thrown any time an illegal parameter is specified during execution of the constructor method. This will be caught during the text file loading step in our driver class and dealt with robustly.

Specifically, you will:

- create a class that extends the **Exception** class,
- throw this exception when FinchAction parameters are incorrectly specified to the constructors, and
- catch this exception in the driver and robustly respond to the exception.

For example, if the following input is given to your program:

```
seek          19 JOG 1000 -10.0 -10.0
seek          17 JOG -10 10.0 10.0
seek          15 RGB 100 200 10 10 0
seek          14 RGB -25 100 100 10 0
seek          21 SOUND 100 250
seek          20 SOUND -100 250
seek          23 SOUND 100 -250
```

your program will respond with output similar to:

```
FinchException: Illegal duration (-10)
    seek          17 JOG -10 10.0 10.0
FinchException: Illegal duration (-25)
    seek          14 RGB -25 100 100 10 0
FinchException: Illegal duration (-100)
    seek          20 SOUND -100 250
FinchException: Illegal frequency (-250)
    seek          23 SOUND 100 -250
> show all
seek: priority=15: duration=100, Beak: darken = false , color = 200, 10, 10
seek: priority=19: duration=1000, Jog: -10.0, -10.0
seek: priority=21: duration=100, Tone: 250
>
```

Milestones

Milestone 1: Create a FinchException Class

Create a **FinchException** class that extends **Exception**. This class should:

- contain (at the very least) an instance variable that stores an error string. This string will be used to describe the nature of the error, and
- implement a **toString()** method.

Milestone 2: Throw FinchExceptions

For each **FinchAction**, the FinchException must be thrown in response to an illegal constructor parameter. Specifically, the exception should be thrown when:

- duration is negative, or
- frequency is non-positive.

Note: the changes required to take this step should be small and straight forward. If this is not the case, then you are probably on the wrong track and should talk to one of us.

Milestone 3: Catch FinchExceptions

Modify the text loading step of your driver class to catch FinchException on creation of a new FinchAction instance. Once this exception is caught, you must print an error message that includes a description of the exception and the line from the file that resulted in the exception.

Note: this implementation should only be a few lines of code. If it is more, then you need to re-evaluate.

What to Hand In

All materials are due: Friday, November 13th, 2009, 5:00pm.

Hand in the following:

- an electronic copy of your modified code (to D2L), and
- include a note at time of hand-in as to which group members participated in the lab.

NOTE: ONLY HAND IN ONE COPY PER GROUP.

In addition to handing in a copy of the code, you must do a short demonstration of your working code for the TA or the instructor. Ideally you will do this before the end of the lab period. Otherwise, please make an appointment before the deadline. All group members should be in attendance during the demonstration.