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

November 11, 2010

Due: Friday, November 12th, 2010, 11:29am

Introduction

In many contexts, a method can be faced with an unexpected situation that prevents it from meeting its stated postconditions. Such situations can arise due to an incorrect set of parameters (specifically, not meeting the stated preconditions), problems in communication with a disk or another computer, and even bugs in the implementation of the method. *Exceptions* provide the means for a method to halt its execution and to alert its caller that a problem has occurred.

In your Finch class implementations to date, you have addressed the problems that arise in the classes by detecting errors and then taking steps to repair the offending variable values (e.g., the *duration* instance variable). In this lab, you will instead raise exceptions when these problems arise.

Objectives

By the end of this laboratory exercise, you will be able to:

1. create exception classes,

2. throw exceptions to indicate errors, and

3. catch exceptions to yield robust behavior.

Getting Started

Create a "lab 8" project. From your project 3 implementation, copy to your new project the implementation of **FinchAction** and all of its child classes. In addition, copy the **FinchActionList**, **FinchOrient** and **FinchObstacle** classes. (depending on your implementation, you may need to copy other classes).

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 **FinchAction** and its children, the FinchException must be thrown in response to an illegal constructor parameter.

Note: if the constructor implementation for **FinchOrientationGuarded** and **FinchObstacleGuarded** requires a string that specifies the type of guarding, then you should throw an exception if the string is not recognized.

Note 2: 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

Create a Driver class that tests your new exception capabilities by catching any exceptions and printing out a warning. Specifically, your main() method should attempt to create a set of **FinchActions**. Those that are created successfully should be added to a **FinchActionList**. Those that throw an exception should simply be reported.

At the end of your main() method, display the list of valid **FinchActions**.

What to Hand In

All materials are due: Friday, November 12th, 2010, 11:29am.

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.