This page last changed on Jan 10, 2007 by dswood.

Graph - Path

Session Objectives

By the end of this session students should ...

- Understand the NullObject pattern
- Be able to refactor to the NullObject pattern

Session Overview

Activity
• Lab - Min Path • Lab - All Paths

Session Notes

Lab - Min Path

!Graph - Cost^graph with costs.png!

- Find the minimum path according to either of the criteria introduced so far (hop count or cost).
- The questions costTo and hopsTo should be answered by the path object.

Things to look for:

- Start by copying costTo() to pathTo() and introducing a Path class as the return.
- Will need to implement equals() and hashCode() for Path and Link.
- How to model the concept of no path.
- Path should get pulled out of Node into its own class. It is returned by a public method on Node and has significant functionality.
- NO_PATH null object should be introduced.

Discuss the *null object* pattern. Put the NO_PATH example on the screen and talk through:

- Ask the students when it is commonly used? When you would consider returning null and having null checks all over the place. This is common when you have a class and need to represent it not existing.
  - What is the null object in the Path example? NO_PATH
  - Ask for some more examples of when null object might be used.
- Problems with 'null' that make null object better:
  - Nulls tend to hide errors and make their source harder to find.
  - Nulls tend to cascade so if you are passing nulls around you have to do null checks everywhere.
- Describe to the students the common usage patterns of a null object:

    - Typically declared as constants
    - Typically implemented as anonymous inner subclasses
    - Override methods of the parent with rational defaults that do not have side effects. For example typically they don't throw exceptions, they don't change their own state.
    - Do not escape from module/package boundaries.
    - Beware of excessive checks for null objects. Checks should only occur at the module boundary.

Lab - All Paths

Things to look for:

- Start by copying pathTo() to allPathsTo() and introducing a Paths class as the return.

- How to model the concept of no path.
- Path should get pulled out of Node into its own class. It is returned by a public method on Node and has significant functionality.
- NO_PATH null object should be replaced by exceptions. It is no longer necessary and this prevents it from getting out of the package with public Paths.
    - Point out to students that when you remove all usages of a class you delete the class completely. If it needs to be resuscitated use source control.
- Strategies should be replaced with Comparators.

Discuss:

- Finally found the right question to ask. Can go into YAGNI discussion here, when would it have been right to see this solution.
- Code was constantly deleted.
- Path went from Concrete Class --> Interface --> Abstract Class --> Concrete Class (One of the many reasons not to name interfaces starting with I).

Ask the students how the various solutions to Path differ from what they would have expected prior to OBC.