Learning Activity: Game of Life, Various Enhancements

Back

Educational Objectives

- Gain familiarity with precondition checking
- Update the Game of Life implementation to be in a named package
- Implement independent constants in an interface.

Instructions

This learning activity consists of three updates the Game of Life implementation. While it centers on the files of PA: Game of Life, Next Generation, since the Game of Life is a cumulative exercise / project, these changes will take place using your PA: Game of Life: Graphical Interface files. The updates for this LA will be use as the basis of future programming assignments on the Game of Life.

The enhancements are:

- Board constants for Game of Life
- Named package for Game of Life
- Input validation for GameOfLife
- Adaptive implementation for GameOfLife

The first one in obligatory. The remaining three enhancements can be completed in any order. Completing two of the enhancements successfully constitutes the Minus Level; three, the Check Level; and all four, the Plus Level.

Board Constants

The input validation code needs to check argument values against known bounds. These constants could potentially change. For example, our implementation for this class assumes a 19x19 grid for the Game of Life board. The constraint establishes the boundary values that are used in input validation. It would be just as reasonable to have a board of different dimensions.

These values are typically represented as Java constants, that is, public static final fields. The names of these constants are typically written in all capital letters with underscore characters used to separate words, for example: EXIT_ON_CLOSE. Even though this value is clearly used by the javax.swing.JFrame class, it is not declared there. Instead, it is declared in javax.swing.WindowConstants. This allows multiple classes to use exactly the same constants. This would be particularly important if the constants were reference values, rather than primitive.

Here is a simple interface that declares constants for the 19x19 Go-board Game of Life, GoBoard.Constants.java.

Update MyGameOfLife to inherit these constants. They do not have to be used within the code at this time.

Named Package

Update the Game of Life files to exist in the named package **csc143.gol**. As you saw in the video demo, this is a relatively minor change to the code: the package declaration is added to the source code and the appropriate folder structure is needed. Since the folder structure is a required aspect of named packages, this entails submission of a Java archive (.jar) file. Both source code (.java) and bytecode (.class) files shall be included in the Java archive for the submission to be considered complete.

This entails moving the following to the named package:

- GameOfLife
- MyGameOfLife
- GameOfLifeBoard
- GoBoardConstants
- AltGameOfLife (if used)

The TestGameOfLife test file should be in a different package. We want to prevent test code from accessing package private resources. This class shall remain in the anonymous package.

Successful Completion

The submitted Java archive includes the above files in the indicated packages. The source code shall reflect the package. The folder structure entailed by the packages appears in the archive.

Input Validation

The GameOfLife interface includes only four methods: getCellState, setCellState, nextGeneration, and toString. Add input validation to the implementation of these methods, verifying that the argument values passed to the method are acceptable. (Some of you may have already implemented this work.) If the any of the argument values are not acceptable, notify the client code by raising the unchecked exception, java.lang.IllegalArgumentException. The exception object shall include a meaningful string message that relates specifically to the error(s) in the argument(s). The input validation code shall use the named constants, rather than hand-coded values.

Since IllegalArgumentException is an unchecked exception, no changes are required in the interface. For credit for this enhancement, the JavaDoc comments for the associated methods shall use the @throws tag to notify writers of client code about the possibility of the exception being raised.

Successful Completion

Input validation code appears in the MyGameOfLife implementation of the GameOfLife methods to validate method argument values. Validation code uses named constants rather than hard-coded values. If argument validation fails, the client is notified using an IllegalArgumentException object with a meaningful message string.

Adaptive Implementation

An additional reason to have constants like these in an interface like GoBoardConstants is to allow them to be "swapped out" by a different set of values, if the implementation / configuration details change.

Update the Game of Life files to use this values in GoBoardConstants in the implementation of the GameOfLife interface methods.

The goal is that the constants could be changed by using a different interface (with different values for the same names) and the Game of Life code will adjust appropriately. As an example, Game of Life could be set up to run on an 8x8 checker board, with row and column values ranging from 0 - 7. In fact, there is no reason for the board to be square. We can assume it will be rectangular for simplicity.

Here is a simple interface that declares constants for the 8x8 checker-board Game of Life, CheckerBoard.Constants.java. This interface shall be placed in the csc143.gol package.

Here is an updated test file for the text-based Game of Life, that has been adapted to use the 8x8 checker-board constraints, TestCheckerBoard.java. This class shall remain in the anonymous package.

Here is a video showing the behavior of MyGameOfLife when it has an "adaptive implementation", changing its behavior based on whether it implements GoBoardConstants or CheckerBoardConstants.

For this enhancement, it is not necessary to update GameOfLifeBoard to use the constants. The change to MyGameOfLife is sufficient as a proof of concept. This is a bit tricky. Make sure you keep a working copy of MyGameOfLife somewhere before you start this work.

Successful Completion

The implementation of MyGameOfLife is adaptive, the code for toString and nextGeneration using the named constants from the interface. Testing code is provided as demonstrated in the video above.

Submission

One (1) Java archive with the update Game of Life implementation.

If Named Package has been implemented, the appropriate folder structure needs to be present in the Java archive.

If Adaptive Implementation has been completed, TestCheckerBoard can be set as the entry point in the archive.

Evaluation Criterion

Poor
Only Board Constants correctly implemented

Minus
Board Constants and any one of Named Package, Input Validation, Adaptive Implementation correctly implemented

Check
Board Constants and any two of Named Package, Input Validation, Adaptive Implementation correctly implemented

Plus
Board Constants and all three of Named Package, Input Validation, Adaptive Implementation correctly implemented