Lab5: Applying Undo/Redo Design Pattern to Chess

EECS3311-W20. *March 18, 202*0. Work on your own. **Academic integrity rules apply**.

Important: A submission that does not compile and satisfy basic acceptance tests to check design feasibility and correctness will not receive a passing grade. To pass there are also other criteria. See below.

On March 06, we released some documentation to help you apply the undo/redo design pattern [3111-W20-Public](#) (github). In that documentation, moves were by a King and a Bishop. This Lab handles undo/redo of moves by a King and a Knight.

Undo/redo design pattern

The undo/redo design pattern is discussed in OOSC2 chapter 21, with the following goals:

- The mechanism should be applicable to a wide class of interactive applications, regardless of the application domain.
- The mechanism should not require redesign for each new input command.
- **It should make reasonable use of storage**. (E.g. in this example, you must not store a history of moves where each move is stored with the whole board accompanied by the location of all the pieces. You must store just **minimal** information needed to undo or redo a specific move. Note that in this example the history is unbounded and thus efficient storage is essential).
- It should be applicable to arbitrary-levels of undo/redo.

~sel/retrieve/3311/lab5

The `retrieve` provides you with the following:

- `docs` folder has the grammar file `chess.ui.grammar.txt`, from which you can generate an ETF project.
- `regression-testing` folder has an oracle `oracle.exe`. Your submission must match the Oracle character-for-character. In this folder you will also find two acceptance tests `at01.txt` and `at02.txt` to get you started. You will want to write many more to test your design correctness.
- Your design must apply the undo/redo design pattern.

Submission

Submission instructions are at the course wiki.

You must also produce a clear PDF document top level clear BON class diagram `bon-draw.io.pdf`, which is a selective version of the top-level architecture that documents your top-level design. It is selective because it cannot include everything, just enough to document the top-level design so that we can check that you have actually applied the undo/redo design pattern.

Grading

- 50% for Design Correctness (provided you have applied the undo/redo design pattern) using acceptances tests.
- 50% for Design Architecture matching the undo/redo pattern. You must follow BON class diagram design conventions as documented on the course wiki.

To achieve a passing grade:

- Your submission must compile and satisfy basic acceptance tests to check design feasibility and correctness. We will apply additional grading tests to test your design feasibility and correctness.
- and, you must implement and document the efficient undo/redo design pattern.