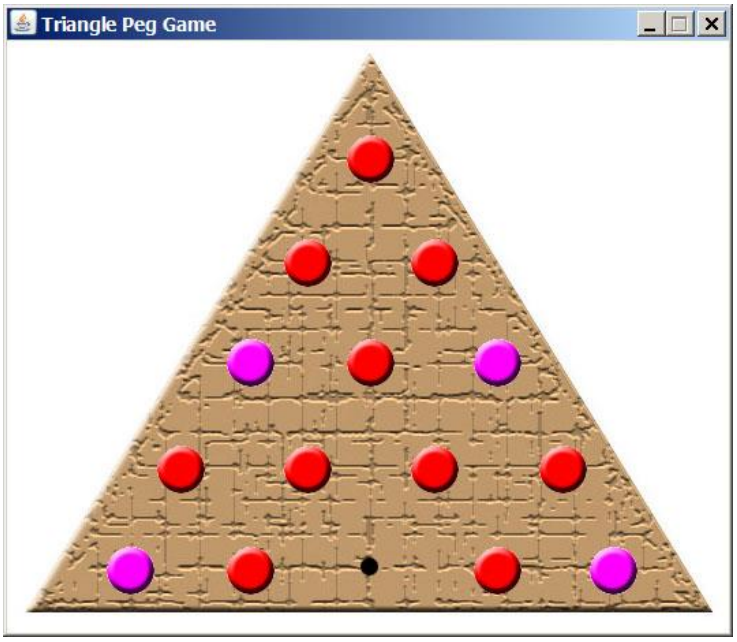
Laboratory 10: Triangle Peg Solitaire



Triangle Peg Solitaire is a game where you start with several pegs at fixed locations on a triangular board. There is one location that does not have a peg in it. This location can be anywhere on the board, but it is usually the bottom-middle spot. You remove pegs one at a time by jumping over other pegs to empty slots, removing the jumped pegs from the board. You can only jump over the nearest neighbor pegs. The goal is to have only one peg remaining at the end.

- Red pegs are pegs that currently cannot jump over any other pegs.
- Purple pegs are pegs that currently can jump over at least one other peg.
- The green peg (after the first jump) is a peg that can be used to find the optimal solution (fewest pegs left at the end) from the current board configuration.

Starting Files:

- TrianglePegSolitaire.zip
- PegState.java
- Pegs.java
- PegFirstClickState.java
- PegSecondClickState.java

Lab:

- Interfaces
- Dynamic Changeability
- Two-Way Has-A
- State Design Pattern
- Executable Jar Files

Part I: Peg State

Triangle Peg Solitaire has three distinct states.

- **First Click:** A clicked on peg has only one possible jump, so take that jump.
- **Second Click:** A clicked on peg has two possible jumps, so two mouse clicks are required to execute the jump.
- **Game Over:** There are no more legal jumps (the game is over). The next click will start a new game with the clicked-on location as the open spot.

What the states have in common is that *they are all activated on a mouse click*. Take a look at **PegState.java**. The only method in this interface is mouseClicked. The states that you will be writing will implement this interface.

Part II: Pegs

Use the state design pattern to complete **Pegs.java**. Remember that this design pattern requires a two-way has-a relationship. The states that you will need are **PegFirstClickState**, **PegSecondClickState**, and **PegGameOverState**. *Note that Pegs has methods to determine if a peg has been clicked on or if an open slot has been clicked on*. You will need both of these!

- public int findSelectedPeg(int x, int y)
  - used to find out which peg was clicked on
- public int findSelectedSlot(int x, int y)
  - used to find out which slot was clicked on
- public ArrayList<Jump> getPossibleJumps(int select)
  - returns an ArrayList of the possible jumps from pegs

Part III: The States

Complete the three states. You will need to call methods on Pegs.java. For PegGameOverState the mouseClicked method looks for the clicked-on slot. This slot becomes the open space for the new game. An invalid click stays in PegGameOverState.

The main method is in the TrianglePegSolitaire class.

Now try creating an executable jar file that you can double-click to run (done in make.bat):

1. Compile your java code, generating all of the program's class files.

2. Create a *manifest file* containing the following 2 lines: (done for you)

```
Manifest-Version: 1.0
Main-Class: name of class containing main (without .class extension)
```

The name of the file should end with the `.mf` suffix. It is important that the file ends with a blank line.

3. To create the JAR, type the following command:

   ```
   jar cmf manifest-file jar-file input-files
   ```

   The *input-files* must include any class files, images, sounds, etc. that your program uses.

4. To view the contents of the JAR, type:

   ```
   jar tf jar-file
   ```

5. Execute the application from the command line by typing:

   ```
   java -jar jar-file
   ```

   If the application is GUI-based, you can also launch it by double-clicking the JAR file.

Only one submission per team is necessary, but please make sure to include both names at the top of your source code, as well as in the comments section when submitting the lab, so both people can get credit.