Candidate A

Model

- Decent implementation of Tile
- Board class often makes use of int[] when Point would be a much better choice
- Messy and does not make good use of loops
- Difficult to understand - uses ArrayList<int[]> ??
- Board class has a ridiculous number of instance variables

GUI

- Pawn movement is unique and interesting
- “Size 2” button makes it too large for screen

Tests

- 24/70 tests fail
- Failing tests imply that Tiles are not constructed properly
- Tests have good code coverage

Javadoc

- There is little to no Javadoc present
- There are some in-line comments with basic information
- No detailed information about how the code works
- Code is difficult to understand, especially without Javadoc

Overall

- Does not check to ensure 2-4 players

Looking to the Future

- Stage 2: Code needs generous testing and cleaning
- Formula Cards: formula variables in place, not too hard to build off of that
- Magic Wands: wand variables already in place
- Save/Restore: toString would not be too difficult because the board has instance variables for pretty much everything you could imagine, code is convoluted, so restoring would be a nightmare to read through

- Bottom Line: While most of the guts seem functional, they are a tangled mess, it is definitely possible to restore without much typing, but with a lot of reading and reasoning

Candidate B

Model

- Have getters for almost everything necessary
- Pretty good implementation of Tile
- Paths are created separately from the Tile itself
- Forces player to pick up token when landed on
- Not very confusing code, separate methods to take care of things
- ToString methods are nicely formatted

GUI

- There are plus-shaped tiles that should not be present (this may be a Model problem)
- No instructions on how to play the game
- Easy to use (once you figure it out)
- Complicated GUI code for highlighting tiles, and basically everything else too
- Does not give any clue as to where is a valid movement

Tests

- Tests do not have good code coverage; if they did, the team would have caught the error in their checking for 2-4 players
- Tests are confusing and illogical

Javadoc

- Javadoc is clean and informative

Overall

- Checking for 2-4 players fails, but this is an easy fix
- Complicated GUI code makes it difficult to add things if needed

Looking to the Future:

Stage Two: Minor change to command line arguments, token retrieval, and tile shapes seem to be the most of it, plus testing
Formula: Cards are there, wouldn't be hard at all
Wand: Not too difficult

Save/Restore: ToString methods add a lot of convenience already, underlying model is sound, so it shouldn’t be too hard
Bottom Line: While the code is scary at first, most of the problems seem relatively minor and a solid model will minimize workload

Candidate C

Model

- Good implementation of Tile
- Throws ArrayIndexOutOfBoundsException when player tries to move off the Board (this should be an easy fix)
- Forces the player to pick up the next Token if they land on it - does not give the opportunity to move over it or choose not to pick it up (this should also be an easy fix)
- Board is clean and easy to follow

GUI

- Unappealing and difficult to see
- No instructions on how to play, but it is more intuitive than the others
- Arrow keys make movement easy
- Code is difficult to follow and should be rolled into loops

Tests

- 16/51 tests throw errors, and 2/51 tests fail
- Many of the Board tests throw NullPointerExceptions, which should resolve itself once Board boundary checking is implemented

Javadoc

- Clean and informative Javadoc

Overall

- No checking to ensure 2-4 players
- Good use of Observer pattern
- Some refactoring to do
    - Names do not follow convention
    - Many fields are public when they should be private and/or use a wrapper class instead of a primitive, which is unnecessary
    - The strings “North”, “South”, etc. should be factored out as constants of the Tile class
    - There are unnecessary fields that can be factored out

Looking to the Future:

Stage 2: Boundary checking and token retrieval are the brunt of the problems, minor tweaks here and there

Formula: Seem a little harder here than with others because the token is confusing

Wand: wcount variable is the only trace of existence

Save/Restore: Current model would be difficult, but refactoring could make it possible

Bottom Line: A little more work than others, but none of it is all that hard of work

Candidate D

Model

- Good design for the board, nice and simple
- There are many things that are missing, like they created classes that have nothing in them
- Scoring is not done correctly

GUI

- Does not tell the player what to do at all
- Intuitive to use, but unappealing to click on arrow keys for movement
- Does not let the player do anything past making the first move as it freezes up
- Multiple windows are unappealing

Tests

- 46 out of the 54 tests throw errors
- Many methods only get one test for them
- Some methods don't get any tests at all
- Some tests are just not even needed

Javadoc

- Starts with good descriptions of methods
- Later on, there is less and less Javadoc until eventually there is none

Overall

- There are many problems with this code but there are some good things as it seems the team first gave it their all but then soon gave up
- Needs to check for more errors like there not being two to four players

Looking Forward

Stage 2: Need to do some serious testing and javadocing, among other things
Wand: Basic structure of wandersty is already in place
Formula: Token list should make this relatively straightforward
Save/Restore: Variables in the tile will help a lot, not so sure about restoring
Overall: It seems like so much work just to get to stage 3

Candidate E

Model

- The code is well written but it is not the most efficient; the game runs slowly
- Decent implementation of Tile, but uses ints and assumes values will always be 0 or 1, when they should have used booleans

GUI

- The GUI has a nice clean design that lets the player know what to do
- The game proceeds slowly
- GUI code is extremely difficult to follow - there are too many fields with similar names and the use of the Image class is difficult to understand

Tests

- 6275 tests is too many. It is daunting to try and read all of these unnecessary tests.
- The tests create new GUIs, which causes them to run very slowly
- All the tests pass with no errors
- Many tests are checking integrated functionality with the GUI and Model - these are not unit tests.

Javadoc

- Every method gets a good well written javadoc
- The level of the javadoc is the same throughout the code
- Put in a lot of care so anyone could understand the code

Overall

- Everything is well organized and the code is the cleanest, can get lost very easily
- Improper use of the Observer pattern
- There is a check to see if there is two to four players
- The code is not the most efficient so it causes everything to run slowly
- Could get rid of much of the code that has been commented out as it just takes up space

For the Future:

Stage 2: Pretty good on that part
Wands: I'm not seeing any base code for this

Formula: Tokens list should make straightforward
Save/Restore: Everything knows everything, so that’s a lot to restore
The Bottom Line: While the base code is there, it is so complicated it might not be worth it to dive in