References

If the links for specific classes in Black Widow Chess do not work, navigate to the package src.com.chess.engine.classic from this link:

https://github.com/amir650/BlackWidow-Chess

Specific instances where I took help from this program, either directly copying code or getting inspiration for design that I ended up finishing by myself:

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/Piece.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/Move.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/Board.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/Table.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/FenUtilities.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/King.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/Player.java

https://github.com/amir650/BlackWidow-Chess/blob/master/src/com/chess/engine/classic/board/BoardUtils.java

https://www.youtube.com/watch?v=h8fSdSUKttk

https://www.youtube.com/watch?v=Ol2pAXgVE7c

(the above two links contain videos demonstrating an early version of this program that uses a `Tile` class which I based my own off)

https://www.youtube.com/watch?v=P-qGwTNBwdQ

https://mkyong.com/java/how-to-open-a-pdf-file-in-java/

https://jsfiddle.net/q76uzxwe/1/

https://stackoverflow.com/questions/8675038/increasing-decreasing-font-size-inside-textarea-using-jbutton

https://stackoverflow.com/questions/45558095/jscrollpane-not-scrolling-in-jtextarea

https://stackoverflow.com/questions/15771949/how-do-i-make-jfilechooser-only-accept-txt

https://stackoverflow.com/questions/14126975/joptionpane-without-button

https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html

https://docs.oracle.com/javase/8/docs/api/javax/swing/JDialog.html

https://docs.oracle.com/javase/7/docs/api/java/io/OutputStreamWriter.html

https://www.programiz.com/java-programming/inputstreamreader

https://stackoverflow.com/questions/54141716/java-second-last-occurrence-of-char-in-string

https://docs.oracle.com/javase/tutorial/2d/images/loadimage.html

https://stackoverflow.com/questions/4265159/javas-do-nothing-on-close

https://stackoverflow.com/questions/10346449/scrolling-a-jpanel

https://www.baeldung.com/java-count-chars

https://www.youtube.com/watch?v=CajXXmhIndI

https://stackoverflow.com/questions/4434894/java-swing-jmenu-mnemonic

https://stackoverflow.com/questions/9474121/i-want-to-get-year-month-day-etc-from-java-date-to-compare-with-gregorian-cal

Images of chess pieces downloaded as PNGs and renamed from files found here:
https://en.wikipedia.org/wiki/User:Cburnett/GFDL_images/Chess
I used black and white versions of every regular piece on a transparent background.

Code for the `Delta` class (com.DarkBlue.Move.Delta) was copied and slightly modified from another Ramapo College student's chess engine senior project.

The class hierarchy I used for chess pieces (a generic abstract piece deriving into a pawn, rook, knight, bishop, queen, and king) was my own idea that I wrote before I designed any other classes or got any help online. I did not base this hierarchy off any other chess engine, but chess engines I saw such as Black Widow use an identical hierarchy. I did, however, base certain fields and other designs choices of each specific subclass off things I saw in Black Widow.

Overall structure and certain fields of the `Player` base class were inspired by Black Widow, but subclasses and methods returning certain types of moves and numbers of captured pieces were my design.

The subclasses I used for the `Move` class (an abstract move deriving to a regular move, an attacking move, a castling move, and an en passant move) took inspiration from a few subclasses found in Black Widow, but my designs ended up being much simpler. I chose not to use as many specific subclasses for edge cases as Black Widow did as I felt it would be too cumbersome. Despite this, I did end up using certain design choices found in Black Widow, such as including a copy of the board in the `Move` object.

Directional and knight move king safety algorithms in `MoveEvaluation` were modified from Alpha Beta Chess by Jonathan Warkentin, but I wrote the section checking tiles directly next to the king myself.

Except for some of the king safety algorithms mentioned above, code for every move calculation method in the `MoveEvaluation` interface was completely designed and written by me. The original code for move calculation in Black Widow checked all 64 tiles one at a time for legality and it gave me the idea to look only at the tiles that were actually relevant to the way a piece moved. The author of Black Widow likely changed this code later on and it ended up looking similar to mine.

Any such similarities of move calculation algorithms are entirely coincidental.

Certain methods in `BoardUtilities` may look and function similarly to those found in Black Widow's `BoardUtils` class, but were written and tested by me.

The internal `GUITile` and `GUIBoard` classes, and methods such as drawing a tile and setting the internal board, were based on similar internal classes found in Black Widow's `Table` class.
Methods to build the GUI board based on the human player's color and place algebraic notation around its border, as well as turning tiles green if a piece is selected and back to their original color when moved or deselected were my own original design.

The `CapturedPiecePanel` and its associated methods in other classes was entirely designed by me, but is based off the `TakenPiecesPanel` in Black Widow.

The `DarkBlueMenuBar` was based off a similar structure in Black Widow but was entirely my own design. I coded the layout of the menu and the handling of `ActionEvent`s myself, but I got help from Stack Overflow in regards to setting mnemonics for each menu item.

The `MoveTextArea` was based off a move history panel in Black Widow, but was entirely designed by me.

The `GameWatcher` class and its associated methods were designed entirely by me as an alternative to using the now deprecated `Observer` class in Java.

The `GameState` enum and methods for determining it are entirely my own design.

Adjustments of castling rights made by the `Board` and observer are entirely my own design.

The structure of the mouseClicked() method in the `MouseAdapter` was directly based on the design used in Black Widow but I modified it significantly, added my own forms of error checking, and broke off sections of code into my own methods that were redesigned significantly from how they look in Black Widow. My program only allows for the primary mouse button to select, move, and deselect pieces. I took inspiration for the `MouseAdapter` from a video I found describing it on YouTube, completely unrelated to Black Widow.

Parsing of both files and custom FEN strings were my own design unless indicated otherwise. Determination of a valid FEN string took partial inspiration from Black Widow as described above, but I wrote and tested most of the code myself. Determining playability of valid FEN files was my own design.

The `BoardBuilder` class was directly based on the `Builder` class in Black Widow, but I added a constructor of my own, additional exception handling, and a method to determine tile color when constructing a board. I also removed certain fields that I found were unnecessary or cumbersome in Black Widow. All methods in the `Board` class that execute moves on a board were written by me, as were the methods that determine castling rights. The toString() method in the board parsing it into a partial FEN string without move clocks was also entirely my own design.

The `GameUtilities` interface was mostly designed by me, except for the replacements of numbers in FEN strings with that number of hyphens when parsing, and switch statements that parse the type of piece by what letter it is. I took both of these ideas directly from Black Widow, but I did not keep looking at Black Widow when I designed my own FEN parser. The usage of the `Builder` class when parsing a FEN string in Black Widow is something I ended up figuring out on my own while I designed mine, and code regarding string splitting and whitespace trimming also ended up looking similar but these methods were not directly copied. Certain similarities in basic methods such as determining whose turn it is, determining castling rights, and the names of other methods looking similar is entirely coincidental.

All code that handles pawn promotions is entirely my own design.

The AI was taken from Lauri Hartikka's guide to building a simple chess AI on JSFiddle, but I modified it by making the syntax compile in Java and adding edge cases to take promotions into account.

The "Thinking…" dialog was based on answers I found from Stack Overflow and modified slightly.

The "Help Me Move" algorithm and its usage of the `SwingWorker` class was based on the computer's own move algorithm used in Black Widow, but the minimax algorithm is called from the human's perspective. The algorithm that assembles a message to send the best move to the user is entirely my own design.

Popups with the FIDE laws of chess and the instruction manual were modified from code I found from MK Yong, as indicated on a URL above.