Check:

* Comments
* Constructors(no-argument)
* Annotations
* Efficiency of program and output
* Why is IOException thrown?(Something bad happened or interrupted is fine.

Testing: (JUNIT is white-box testing)

* If user enters an integer less than or equal to 0 or a non-Integer value for k or n then this value is not accepted and an error message is output.
* If user enters doesn’t enter either 1 or 2 for strategy number then this value is not accepted and an error message is output.
* If user enters a filename which cannot be found then an error message must be output. It shouldn’t throw an exception if a non-String value is entered.
* If the file read doesn’t contain at least 2\*k\*n lines of non-negative integers (therefore no String objects or null values) then an error message should be output. If there is a null value for a line then this may be the end of the file or a blank line this needs to be checked if this is the case.(Check what happens when file contains blank lines(Ask Jonathan maybe)
* Only one winner should be displayed to screen or in files.
* When the game has been paused or resumed by the user then a message detailing this should be output to the user and the game state at pausing should be correctly represented in the player output files.
* Check that a player number cannot be 0 but a card denomination can.
* The saveToFile method in the CardDeck class should save information corresponding to the CardDeck instance upon which this method is called.
* The methods addCard() and popCard() defined in the CardDeck class add the Card argument to the bottom of the deck and take off the card argument from the top of the deck, respectively.
* The player files should represent a game correctly, for example if a Player wins it should have a final hand with one card denomination repeated.
* If a deck becomes empty then the player drawing from it must let another thread process instead.
* Maybe but input for k and n into one loop to save space.