Write and test a program that uses threads to implement players for a simple card game. The game has 4 players and 4 piles of cards, one between each pair of players. The players draw cards from the **top** and discard cards to the **bottom** of the piles in the following pattern:

	Pile	
Player	Draw	Discard
1	1	2
2	2	3
3	3	4
4	4	1

4 Players
4 threads
4 piles of cards to draw & discard

The first player to collect 4 cards of the same value wins the game. The card deck has 32 cards: 4 of each value from 0 through 7. Initially each player is dealt 4 cards in round robin fashion starting from Player 1, and the remainder of the cards are added to the piles, starting from Pile 1, in round robin fashion (with the first card added to each pile the first one drawn, as in a queue).

Strategy

So that the same player (usually) wins for the same input deck the simple strategy you should use has each player preferring certain card denominations. In particular, player 1 prefers 0's and 1's, player 2 prefers 2's and 3's, player 3 prefers 4's and 5's, and player 4 prefers 6's and 7's. This means that (after drawing a card from the proper deck) a player discards any card from a non-preferred denomination, if possible. If that is not possible, then the player discards a card from the denomination with the fewest number of cards. To ensure progress in the game, a player may not indefinitely hold a card from a non-preferred denomination. This means that you must implement some aging algorithm for selecting a non-preferred denomination card that ensures that any particular non-preferred card is eventually selected for discard..

For example, if player 2 has (after drawing a card from the proper deck) one 2, one 4, one 5, and a pair of 3's, then the player would discard either the 4 or the 5. If player 3 had the same set of cards, the player would discard a 2 or a 3. An example of the other case, is that if player 2 had three 2's and a pair of 3's, then the player must discard a 3.

Implementation

The Main class must start the game application by creating the players and piles, distributing the cards and creating the various threads needed to run the game. For the various classes, some of your methods will need to prevent multiple threads from accessing shared data simultaneously. You may use the Queue class for the card piles.

methods

In addition, you will need to implement a procedure that handles stopping players who did not win the game. The winning player can simply exit the run procedure that started its thread execution. However, this player must notify the other player threads that they need to stop (e.g., by setting a member value in the player class via a synchronized method, that is subsequently checked by all player threads). Each thread must print a message when it finishes (e.g., "Player 1 wins and exits", "Player 2 exits", etc.). Also, drawing and discarding is an atomic action, so a player must discard after a draw, even when the player wins. This means a player always has 4 cards in the hand whenever printing the contents of a hand.

Input and Output

There is no user input for the game. Use the following declaration for the initial cards. Test your program by varying this definition.

int [] hands = {2, 3, 4, 0, 7, 1, 6, 5, 6, 5, 0, 2, 1, 7, 4, 3};
int [] pilecards = {3, 6, 1, 5, 4, 2, 0, 7, 1, 2, 3, 0, 5, 6, 7, 4};

The output must be written to four output text files (output streams), which are named player1.txt, player2.txt, player3.txt and player4.txt .

Each player must demonstrate card playing actions by printing the card drawn, the card discarded and the current hand for each action. The action must be labeled with the player number. Each player's actions should be written to a separate output file.

For example, intermediate moves should print something like the following:
player 1 draws a 4
player 1 discards 4
player 1 hand 5 5 5 6

The final move of the winner prints something like the following:
player 3 draws a 7
player 3 discards a 3
player 3 hand 7 7 7 7
player 3 wins
player 3 exits

As the final output for each player at the end of the game, the player should print three lines, appropriately customized for that player, to the player's output file. The first states whether the player won or lost, the second lists the player's hand at the end of the game, and the third lists the contents of the pile the player draws from at the end of the game. The lines look like this, with required keywords in bold:

WIN yes (or no)
HAND card1 card2 card3 card4
DRAWPILE contents of draw pile, separated by spaces (e.g., 2 5 3)