Exercise 6 (10 points) – can be done individually or in pair

- The first lines of all source files must be comment containing names & IDs of all members. Also create file readme.txt containing names & IDs of all members.
- Put all files (source, input, output) in folder Ex6_xxx where xxx = your full ID. That is, your source files must be in package Ex6_xxx and input/output files (if there is any) must be read from/write to this folder. From now on, you'll get point deduction for wrong package & folder structure.
- The group representative zips Ex6_xxx & submits it to Google Classroom. The other members submit only readme.txt. Email submission is not accepted.
- The exercise is graded only once, and after graded, members can't be added.

==========

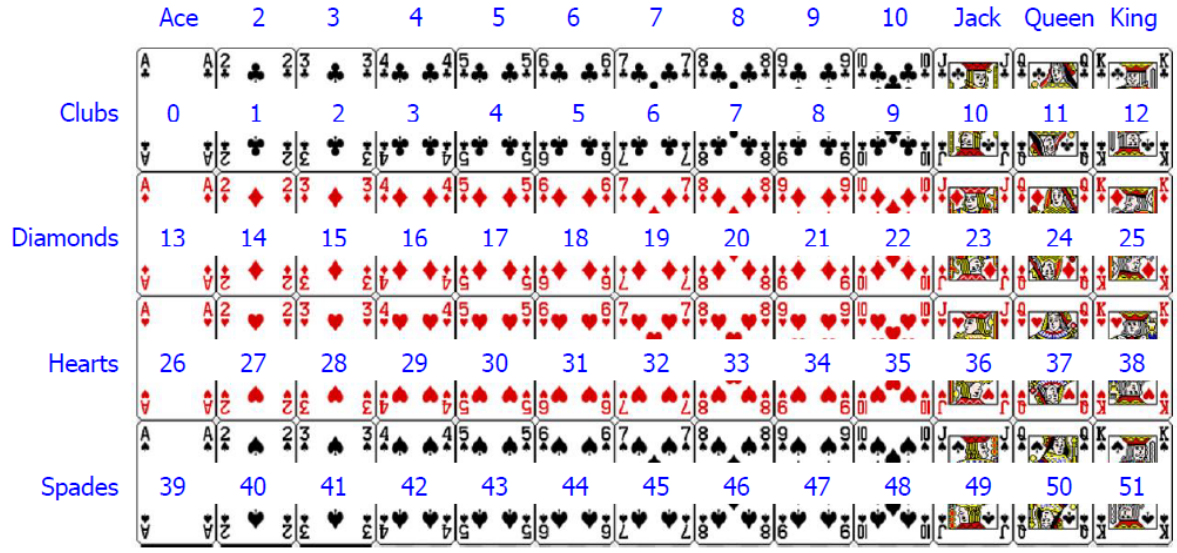
1. Complete classes **OneCard** and **CardThread**. Modify them as needed. You can add more variables & methods, but do not change the visibility of existing ones

```
class OneCard {
  private int score;        // 0-51
  private int suit;         // clubs, diamonds, hearts, spades
  private int rank;         // ace, 2-10, jack, queen, king

  public OneCard(int sc)    { score = sc; }
}

class CardThread extends Thread {
  private PrintWriter out;
  private ArrayList<OneCard> randomCards;

  public void run() {
    // Create PrintWriter object to write result to a separate file

    // Execute steps 1-3 in loop:
    //   1. Draw 4 cards from the same deck. The cards must not duplicate.
    //   2. Print round number and these 4 cards to output file.
    //   3. If all cards are from the same suit or have equal rank, stop the loop.
    //      Otherwise, clear randomCards & continue to the next round.

    // After the loop, print #rounds to the screeen
  }
}
```

** The output file must be placed in the same folder as your source file

Note – to use java.util.Random object

```
Random random = new Random();
Random.nextInt(11);            // an integer in [0, 11) range
Random.nextInt(5, 11);         // an integer in [5, 11) range
```



2. Write another class that acts as the main class. In its main method
   2.1 Ask user for #threads
   2.2 Create CardThreads to perform the tasks in (1)

```
--- exec:3.1.0:exec (default-cli) @
Number of threads =
3
Thread T1 finishes in 12 rounds
Thread T2 finishes in 73 rounds
Thread T0 finishes in 69 rounds
------------------------------------
BUILD SUCCESS
------------------------------------
```

In different runs, the finishing order between threads should be different. If it is always T0, T1, T2, …, then you may not do multithreaded program properly.

Threads also compete for System.out. So, if #rounds are close, the one who finishes first may get System.out later.

T0.txt

```
Round   1  [ 3 of Diamonds ,  9 of Hearts   ,  3 of Hearts   ,  Q of Hearts   ]
Round   2  [ 8 of Spades   ,  7 of Spades   ,  5 of Diamonds ,  A of Spades   ]
...
Round  68  [ 4 of Hearts   ,  2 of Diamonds ,  9 of Clubs    ,  2 of Spades   ]
Round  69  [ 6 of Hearts   ,  5 of Hearts   ,  4 of Hearts   ,  J of Hearts   ]
```

T1.txt

```
Round   1  [10 of Diamonds ,  2 of Spades   ,  J of Clubs    , 10 of Hearts   ]
Round   2  [ 8 of Clubs    ,  4 of Spades   ,  K of Diamonds ,  2 of Spades   ]
...
Round  11  [ 8 of Diamonds ,  J of Clubs    ,  K of Hearts   ,  4 of Hearts   ]
Round  12  [ 7 of Spades   ,  9 of Spades   ,  4 of Spades   , 10 of Spades   ]
```

T2.txt

```
Round   1  [ 8 of Diamonds ,  6 of Hearts   ,  3 of Hearts   ,  5 of Spades   ]
Round   2  [ 9 of Hearts   , 10 of Spades   ,  6 of Spades   ,  3 of Spades   ]
...
Round  72  [10 of Clubs    ,  K of Diamonds ,  2 of Clubs    ,  3 of Hearts   ]
Round  73  [ K of Diamonds ,  6 of Diamonds ,  3 of Diamonds ,  Q of Diamonds ]
```