Java Concurrency Collections Exercises

1. Write two java thread programs with **Callable** implementation with ***Future*** return types to read and write from common array list of Integer objects. Use AtomicInteger objects in the list and define read and write operations on the list without synchronized keywords.
2. Implement the above read and write operations with Semaphore to lock on normal integer values (Don't use AtomicInteger) and check the performance with locked and atomic values.
3. Implement a Hashmap with keys always returns a constant value and check the results.
4. Define Person class with id, name and salary variables. Define ArrayList to contain multiple duplicate Person objects. Design program with two threads as Producer to verify the duplicate and other thread Consumer to remove these duplicate and add to a Set collection. Define Producer and Consumer with BlockingQuueue shared object for signaling.
5. Implement the above with semaphore for signaling between the Producer and Consumer threads.

6. Define two threads with Callable one for writing even numbers and one for writing odd numbers at interval of 2 seconds on console. Use AtomicInteger list as shared object to synchronize the access between the threads to check odd and even numbers **without** the use of synchronized keyword or synchronized blocks.

7. Implement the above with SynchronousQueue implementation for signaling between the two threads.
8. Define a List containing employee information (name, age, salary, address) and share this across two threads with **Callable** and **Future** as one to read with iterator and other thread to add/remove records by intervals. Use **Collections synchronized** wrapper methods to synchronize the access.

9. Define a User class with operations for updateUser and readUser methods. Inside updateUser write to user list and in second to wtite to list. Provide console print messages with intervals of 2 sec to display progress. Implement locking with Collections synchronized methods.

****