The University of Jordan
Faculty of Engineering and Technology
Department of Computer Engineering

Programming of Networks Protocol
Assignment-1
Eng. Asma Abdel Karim

** Write a Java program that allows the user to resolve host names and IP addresses by creating *InetAddress* objects in different ways, and saving them and restoring them from a file on the disk as follows:

- The program uses a file named *ip.out* to serialize and deserialize *InetAddress* objects. When the program is first started, serialized *InetAddress* objects must be restored from the file for the program to start its operations.
- The program must then continuously ask the user what to be performed as follows:

1. Enter a new host name to be resolved.
2. Enter a new IP to be resolved.
3. Read host names to be resolved from file.
4. Display all IP addresses.
5. Save and quit.

Note-1: Use an in-memory data structure (e.g. *ArrayList*) to add resolved *InetAddress* objects during the program execution.
Note-2: All created *InetAddress* objects must be logged in a log file named "*log.txt*" with the date and time of their creation. For each *InetAddress* object, a line is written to the file that contains the following info in order:
<Date/time_of_creation> <Hostname/IPAddress>
Note-3: Even if it's the first time the program is executed and hence there are no *InetAddress* objects to be restored from the file *ip.out*, the program must execute normally and start creating *InetAddress* objects as required.
Note-4: After performing the selected operation, whether it was executed successfully or not, and until the user chooses to save and quit, the program must get back to this main menu to take the user's next choice.

Choice-1: if the user selects choice-1, the program must ask him to enter a host name to be resolved. The program must create an *InetAddress* object that represents this host name and thus resolve its IP address.

- If the DNS lookup succeeds, the *InetAddress* object must be added to the group of created *InetAddress* objects and logged as required in the *log.txt* file.
- If the DNS lookup fails, the program must perform two more trials. If all three trials fail, it must print the message: "*Sorry could not resolve host_name*" and log the line: *<Date/time_of_creation> <Hostname>: Could not be resolved!*

Choice-2: if the user selects choice-2, the program must read an IP address from use, create an *InetAddress* object that represents this IP address, and try to perform reverse DNS lookup to retrieve the host name. The program must ask the user to specify the form in which IP is to be entered:

1. Dotted Decimal Notation.
2. Numbers

a. For choice-1, the program must read the IP in dotted decimal notation. For choice-2, the program must read the numbers that represent the four octets of the address individually.
b. If the IP address is reversely resolved successfully, the *InetAddress* object must be added to the group of created *InetAddress* objects and logged as required in the *log.txt* file.
c. If the reverse lookup fails, it must print the message: "*Sorry could not resolve IP address*" and log the line:
   <Date/time_of_creation> <IP>: Could not be resolved!
d. If the user does not enter a correct IP in the dotted decimal notation (four octets separated by dots) or if the octets values are not numbers between 0-255, your program must print the message "Wrong IP" to the console. Then, return to the main menu.

Choice-3: if the user selects choice-3, the program must ask the user to enter the name of the txt file he wants to read host names from. Host names are assumed to be written on separate lines on the file. The program must read these host names, create *InetAddress* objects for each of them, and operate as specified in choice-1.

Choice-4: if the user selects choice-4, the program must display all *InetAddress* objects that include resolved hostnames/IP addresses to the console by invoking the *toString* method.

Choice-5: if the user selects choice-5, the program must serialize *InetAddress* objects to the file *ip.out*. Objects must be serialized individually, such that the program writes first the count of objects then the objects themselves. Then quit.