**Quiz 4**

Question on use of synchronized

**Question # 1**

***What is the synchronized keyword?***

Java provides a built-in mechanism to provide atomicity called the **synchronized** block. A synchronized method is a shorthand for a synchronized block that spans an entire method body and whose lock is the object on which the method is being invoked.

A synchronized block consists of a reference to an object that serves as the lock and a block of code that will be guarded by the lock.

Synchronized blocks guarded by the same lock will execute one at a time. These blocks can be thought of as being executed atomically. Locks provide serialized access to the code paths they guard.

Below is an example of a class with a synchronized method.

class ContactBook {  
   
    Collection<String> contacts = new ArrayList<>();  
   
    synchronized void addName(String name) {  
        contacts.add(name);  
    }  
}

Note the synchronized method above is equivalent to the following rewrite:

    void addName(String name) {  
        synchronized(this) {  
            contacts.add(name);  
        }  
    }

**Question # 2**

***Is the print statement in the below code reachable?***

    void doubleSynchronization() {  
   
        synchronized (this) {  
            synchronized (this) {  
                System.out.println("Is this line unreachable ?");  
            }  
        }  
    }

**Q**

**A)**

Code isn’t reachabe because we try to synchronize twice on the same object

**Correct Answer**

**B)**

Code is reachable because we can synchronize multiple times on the same object

----

**You skipped the question. Would you like to try again?**

#### Hide Explanation

Synchronized is *reentrant* therefore if a thread is already synchroinzed on an object, it'll be able to resynchronize on it. Note in general, not all locks are reentrant and can cause a thread to deadlock itself.

**Question # 3**

***Consider the below class which has a synchronized method. Can you tell what object does the thread invoking the addName() method synchronize on?***

class ContactBook {  
   
    Collection<String> contacts = new ArrayList<>();  
   
    synchronized void addName(String name) {  
        contacts.add(name);  
    }  
}

***Class may be used as follows:***

        ContactBook contactBook = new ContactBook();  
        contactBook.addName("Trump");

**Q**

**A)**

The main thread

**B)**

Thread invoking the method

**Correct Answer**

**C)**

The object on which a thread invokes the method

**D)**

synchronize is a synchronization keyword and doesn’t require any objects for operation

----

**You skipped the question. Would you like to try again?**

#### Hide Explanation

The method is synchronized on the object on which a thread invokes the method. In the example usage, the method will be synchronized on the **contactsBook** object. Also note that prefixing the method signature with **synchronized** is equivalent of the defining the method in the following manner:

    void addName(String name) {  
        synchronized (this) {  
            contacts.add(name);  
        }  
    }

**Question # 4**

***An instance method synchronizes on the instance object, do you know what object do static methods synchronize on?***

**Hide Explanation**

A static synchronized method synchronizes on the **class object**.