Stage 5.3 - Notifications vs Other Mechanisms

Scene 1: On-Set

Now that we've seen KVO and Notifications, we can discuss the strengths of weakness of the different mechanisms by which objects can interact in Cocoa.

<KEYNOTE SLIDE bold on slide>
The mechanisms we'll compare are:

1. **Ordinary message passing (method calls)**
2. **Delegation**
3. **KVO**
4. **Notifications**

Our comparison will be in terms of coupling and which party drives the timing of the interaction. That is: how much information must be shared between objects for the mechanism to operate, and which party initiates the communication.

1. Message passing, or conventional method calls require a reference to an allocated instance of an object, and access to the public interface for the instance class. A header file for the target class is typically imported into the scope of the the caller. Interaction is synchronously driven by the calling code.
2. Delegation requires both instances to have a reference to the other, and requires the delegate to conform to a delegate protocol. Interaction is driven by the delegating object. While this pattern requires a mutual sharing of references, the interaction is somewhat decoupled from the perspective of the delegating class as the delegate is typically optional and can be an instance of any class conforming to the delegate protocol. This makes it simple to modify or replace the delegate object. Delegation is best applied in cases where you have a 1:1 relationship between sender and receiver.
3. In KVO, the observer only needs to know that a given symbol exists in an observed object. This symbol information may be informally assumed, established using a protocol or even queried dynamically using the objective-C runtime system. Since the mechanics of notifying the observer of a change is handled "under-the-hood" by the runtime, user code in the observed object is unaware of the KVO relationship. Though the observing instance needs to be careful about adding and removing observation and properly handling callbacks, the overall level of coupling in a KVO relationship is minimal.
4. With notifications, no references are known and the code posting a notification has no need to know the number, if any, of listeners. Likewise, listeners have no idea how many, if any, posters for a given notification exist. Notifications are often the best solution when you have a 1:N relationship between sender and receivers.

</KEYNOTE>

Questions:

1. The Objective-C type used for notification names is? (ans. NSString)
2. When calling the method -addObserverForName:(1st arg) object:(2nd arg) queue:(3rd arg) usingBlock:(4th arg), which argument cannot be nil? (ans. 4th, the block argument)
3. True or False. All listeners receiving a notification will receive the same userInfo dictionary. (ans. True)
4. The class method used to get a reference to the default NSNotificationCenter is? (ans. +defaultCenter)
5. True or False. Once an object has added itself as an observer(registered for a notification), it cannot remove itself as an observer. (ans. false)