Jamarcus Coulter

1. What are loaders and how do we implement loaders?
Loaders are an API that loads data from a content provider or data source into a fragment or fragment activity. Loaders run on separate threads and make thread management easier with their own callback methods. Loaders persist and cache data onRestore from configuration changes that prevents duplicate results and can implement observers to monitor changes of a data source. Loaders are created and managed by implementing a LoaderManager.

To Implement a loader, first an activity or fragment must implement LoaderManager and call getSupportLoaderManager(). Then onCreateLoader, onLoadFinished, and onLoaderReset methods are overridden. To create a loader, a class can either extend AsyncTaskLoader, CursorLoader, or Loader class.

2. What is an AsyncTaskLoader?
An AsyncTaskLoader is the loader equivalent to a AsyncTask. It is a loader that can perform tasks on worker (non-main) threads. Unlike AsyncTask, AsyncLoader can persist data through configuration changes and within the lifecycle of an activity/fragment.

3. What is a Handler Thread for?
A Looper is a class that runs a message loop/queue for a Handler Thread and passes the message from the queue to the handler. A Handler sends messages to other looper handlers and and adds message to the message queue. The Handler Thread is the thread that processes the messages in the queue.

4. What are some common threading restrictions in android?
Some common threading restrictions are that almost all UI interactions, including rendering, are processed on the main thread. All long running process should be processed on a non-main thread. Asynchronous threading is necessary to keep the main thread for the UI responsive. Only the main thread can update UI objects.

5. What are thread pools and thread pool executors?
Thread pools are a collection of threads that are managed so that previous created threads are used for processes and only creates a new thread when necessary or specified. Thread pools increase performance when executing large numbers of async tasks by reducing the amount of overhead that would incur from the creation and deletion of threads. Thread Pool Executors, subclass of ExecutorService, manages the thread pool. The executor creates the thread pool, set parameters for the pool like initial size, max size, idle time of threads before termination, and other attributes. The executor opens and closes the thread pool.