Threads and Non-blocking Reusable Controls

1. When and why we will use Threads in our programs?

We will use threads when the computer has multiple cpu's as is the case of most modern desktop computers. Threading is a way of subdividing specific operations within a single application into threads. Now the OS will not only divide processing time among different applications but also among each thread within that application. If we use multithreading with a computer that only has one CPU and we make a lot of threads the risk of getting overhead is higher because we only have one CPU to share its resources. If there were multiple CPU's or cores then we could start one process for each CPU and then subdivide each process into threads.
When we want to make an UI that's responsive so that it doesn't get blocked while performing an operation we would use 2 threads one for the operation and one for the UI itself.

2. Explain about the Race Condition Problem and ways to solve it in Java

A race condition happens when two or more threads try to access some shared data and tries to change the data simultaneously.The thread scheduler can do a context switch between threads so you don't know the order of which the threads will attempt to access this shared resource.
In order to solve the problem we can use Synchronized method which blocks the entire method (so only one thread can access at the time). We can use a lock like the ReentrantLock. We could use the .join which causes threads to wait for each other like the typical semaphore or monitor behaviour. We could use a semaphore and pass a token to a thread which gives this thread access to some critical region i.e. updating a shared counter. There are many ways to synchronize threads the important thing is that they are synchronized if your code has critical regions.

3. Explain how Threads can help us in making responsive User Interfaces

Since the UI is a thread itself it will block if that same thread is used to do a complicated operation like calculating the 100th fibonacci number. The solution is to make another thread which then could notify the GUI when it has finished its calculation in this way the GUI is still responsive and you could do other stuff on it while the calculation is performed.

4. Explain how we can write reusable non-blocking Java Controls using Threads and the observer Pattern

The observer (interface) can be implemented by those who wants to be observers. In this way you could have many subscribers observing the same publisher. These subscribers could be threads waiting for another thread (the subject) to finish it's critical region or whatever. When it has finished it's stuff it could call the update() function for all the observers

and tell them that they should update their label or text field etc. This makes the UI non blocking and responsive.