**Blocking Queue**

* Is A Queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
* A BlockingQueue may be capacity bounded. At any given time it may have a remainingCapacity beyond which no additional elements can be put without blocking.

**What is Bounded and un-Bounded Queue**

* Bounded Queues are queues which are bounded by capacity that means we need to provide the max size of the queue at the time of creation.
* Unbounded Queues are queues which are NOT bounded by capacity that means we should not provide the size of the queue.
* All Queues which are available in java.util package are Unbounded Queues
* Queues which are available in java.util.concurrent package are Bounded Queues.

**Executor FrameWork**

* + Java executor framework ([java.util.concurrent.Executor](https://www.geeksforgeeks.org/java-util-concurrent-executor-interface-with-examples/)), released with the JDK 5 is used to run the Runnable objects without creating new threads every time and mostly re-using the already created threads.
  + Thread pools overcome this issue by keeping the threads alive and reusing the threads

**Future**

* + A Java *Future*, java.util.concurrent.Future, represents the result of an asynchronous computation. When the asynchronous task is created, a Java Future object is returned. This Future object functions as a handle to the result of the asynchronous task. Once the asynchronous task completes, the result can be accessed via the Future object returned when the task was started.

**ThreadLocal**

* + This class provides thread-local variables. These variables differ from their normal counterparts in that each thread that accesses one (via its get or set method) has its own, independently initialized copy of the variable.
  + ThreadLocal instances are typically private static fields in classes that wish to associate state with a thread
* Basically it is an another way to achieve thread safety apart from writing immutable classes.
* Since Object is no more shared there is no requirement of Synchronization which can improve scalability and performance of application.
* It extends class Object.
* ThreadLocal provides thread restriction which is extension of local variable. ThreadLocal are visible only in single thread. No two thread can see each others thread local variable.
* These variable are generally private static field in classes and maintain its state inside thread.

**volatile keyword in Java**

 If one thread modifies its value the change might not reflect in the original one in the main memory instantly.

 Now the other thread is not aware of the modified value which leads to data inconsistency

volatile keyword here makes sure that the changes made in one thread are immediately reflect in other thread

**What is Stream:(core java pg 333)**

Stream carries data just as a waterpipe carries water from one place to another placeit can classified into two

1. Inputstream(receive or read data)
2. Output stream(send or write data)

This concept is used to receive data form input device and send data to output device

**Transient Variables(scjp pg no 50)**

* If you mark an instance variable as transient, you're telling the JVM to skip (ignore) this variable when you attempt to serialize the object containing i**t.**

**Jan (12-1-2022) Assignments**

**What Happens to Static variables while serialization(scjp pg472)**

* Static it isn't part of the instance's state at all. Remember, you should think of static variables purely as CLASS variables. They have nothing to do with individual instances. But serialization applies only to OBJECT

**Serial Version UID**

* It is a private static final long variable
* Created by jvm(op,jdk) for every .class file to recognize the .class file uniquely from one to another
* In process of serialization and de-serialization

**Native Method**

* Native is a modifier in java applicable only for methods
* The methods which are implemented in non java are called native method
* Why – to improve system performance
* For native methods implementations are already given so it must end with semicolon(;)