Types of Thread Application

- In general there are two type of thread applications

  1. Single Threaded Application
  2. Multi Threaded Application

Single Threaded Application:

- When we invoke java application, JVM by default creates a thread is called “main thread”.
- In single threaded application, execution starts at main thread and end at the same thread.
- All the methods of single thread executes sequentially.

Multi threaded Application:

- Creating a user thread from main thread referred as multi threaded application.
- Multi threaded application execution starts at main thread only.
- Program execution completes, when all the running threads moved to dead state.

Understanding join() method

- The join method allows the current executing thread to wait for the completion of another thread.
- Every join() method throws InterruptedException, hence compulsory we should handle either by try catch or by throws keyword. Otherwise we will get compile time error.

Understanding sleep() method:

- If we want a thread to pause performing any actions for a given amount of time then we should use sleep() method.
- This is an efficient means of making processor time available to the other threads of an application.
- we can pause the execution of a thread by using '2'predefined methods.

1)Thread.sleep() //specified time in milliseconds.

2)Thread.sleep(long millisecs, int nanosec) //specified milliseconds and nanoseconds. The allowed nano second value is between 0 and 999999

- However, these sleep times are not guaranteed to be precise, because they are limited by the facilities provided by the underlying OS.

Understanding interrupt() method

- An interrupt is an indication to a thread that it should stop what it is doing and do something else.
- For the interrupt mechanism to work correctly, the interrupted thread must be in either sleep state or wait state.
- If the selected Thread is not in sleep mode then interrupt() will wait until it went in to sleep mode, and then it will cause interruption for that thread.

Example:

```
ClassA a=new ClassA();
Thread t=new Thread(a);
t.start();
t. interrupt();
```

Understanding yield() method

- yield() provides a mechanism to inform the "thread scheduler" that the current thread is willing to hand over its current use of processor, but it'd like to be scheduled back soon as possible.
- If we are using the yield method then the selected thread will give a chance for other threads with **same priority** to execute.
- If there are several waiting Threads with same priority, then we can't expect exactly which Thread will get chance for its execution.
- We can't guess again when the yielded thread will resume its execution.

Getting and setting name of a Thread:

- Every Thread in java has some name it may be provided explicitly by the programmer or automatically generated by JVM.
- Thread class defines the following methods to get and set name of a Thread.
  - ✓ public final String getName()
  - ✓ public final void setName(String name)

Understanding Thread Priorities

- In the Java programming language, every thread has a priority.
- We can increase or decrease the priority of any thread by using setPriority(int newPriority) method.
- We can get the priority of the thread by using getPriority() method
- Priority can either be given by JVM (5) while creating the thread or it can be given by programmer explicitly.
- Accepted value of priority for a thread is in range of 1 to 10.
- Thread priorities are highly system-dependent we should always keep in mind that underlying platform should provide support for scheduling based on thread priority.

- There are 3 static variables defined in Thread class for priority.

public static int MIN_PRIORITY --->1

public static int NORM_PRIORITY --->5

public static int MAX_PRIORITY --->10