* Interrupting thread: When run method return/exception/stop method( deprecate)
* Interrupt method: Set the boolean flag ( interrupt status)

+ Check: Thread.currentThread.isInterrupted()

+ Cant check the interrupt status when thread is block -> throw interruptException

+ interrupted() : static method, current thread; isInterrupted(): method for any object.

* Thread states in java:

+ New: A thread that has not yet started

+ Runnable: A thread executing in the java virtual machine

+ Blocked: Blocked , wating for a monitor lock. Example: waitting lock to enter synchronized method

+ Waiting: Waiting indefinitely for another thread to perform a particular action

+ Time waiting: Waiting for another thread to perform an action for up to a specified waiting time is in this state

+ Terminated: exited.

* Join method: This method puts the current thread on wait until the thread on which it’s called is dead.
* Race condition: problem when two or more thread access shared data
* Solution: using Sysnchronozation.
* Lock: One thread locks the lock object, no other thread can get pass the lock statement.
* Synchronized: ~ using lock.
* Synchronized block: is used to lock specific object.
* Dead lock: Two or more thread is waiting each other.
* The stop method: is inherently unsafe, it terminate all pending method.
* the suspend method: frequently leads to deadlocks, the lock is own by the suspended thread.
* Callable: is similar to a runnable but it return a value.
* Thread pool: use when we need to limit the number of threads running in our application at the same time. There is a performance overhead associated with starting a new thread, and each thread is also allocated some memory for its stack.